Agent-based Short-Term Load and Price Forecasting Using a Parallel Implementation of an Adaptive PSO-Trained Local Linear Wavelet Neural Network

Athanassios M. Kintsakis
Electrical and Computer Engineering Dept.
Aristotle University of Thessaloniki
Thessaloniki, Greece
akintsakis@issel.ee.auth.gr

Antonios Chrysopoulos
Electrical and Computer Engineering Dept.
Aristotle University of Thessaloniki
Thessaloniki, Greece
achryso@issel.ee.auth.gr

Pericles A. Mitkas
Electrical and Computer Engineering Dept.
Aristotle University of Thessaloniki
Thessaloniki, Greece
mitkas@auth.gr

***Abstract*— Short-Term Load and Price forecasting are crucial to the stability of electricity markets and to the profitability of the involved parties. The work presented here makes use of a Local Linear Wavelet Neural Network (LLWNN) trained by a special adaptive version of the Particle Swarm Optimization algorithm and implemented as parallel process in CUDA. Experiments for short term load and price forecasting, up to 24 hours ahead, were conducted for energy market datasets from Greece and the USA. In addition, the fast response time of the system enabled its encapsulation in a PowerTAC agent, competing in a real time environment. The system displayed robust all-around performance in a plethora of real and simulated energy markets, each characterized by unique patterns and deviations. The low forecasting error, real time performance and the significant increase in the profitability of an energy market agent show that our approach is a powerful prediction tool, with multiple expansion possibilities.**

***Index Terms*—Load Forecasting, Neural Networks, Parallel architectures, Particle swarm optimization, Price Forecasting, Wavelet Neural Networks.**

I. INTRODUCTION

The liberalization of Energy Markets has created a new competitive environment, beneficial for the participating entities (energy providers, consumers etc.), as well as for the grid stability. This restructuring of electric industry has motivated the appearance of new entities in the traditional energy markets. A key entity in this case is the concept of retailers who act as intermediaries between producers and consumers of electrical energy. Retailers have to manage their customer portfolio and deal with the incurring imbalances in their customer needs at any given time. In this context, accurate energy demand (load) and electricity price forecasting models are imperative to the operation and strategic planning of retailing companies. In particular, short-term forecasting (1 to 24 hours ahead), is very important, since the involved parties must optimize their bidding strategy based on future knowledge of hourly prices and production costs. Different techniques have been applied to load and day ahead Market Clearing Price (MCP) forecasting such as stationary and non-stationary time series models and models based on computational intelligence. A comprehensive review about electricity price forecasting can be found in [1].Due to the use of linear modeling, time series techniques are having a hard time predicting the nonlinear and rapidly changing signals of load and price.

Computational intelligence models such as support vector regression and neural networks have showed better results. The Wavelet Neural Network (WNN) is a special case of neural network that has been proposed as a better alternative to the classic feed forward neural network for approximating arbitrary non-linear functions [2]. As the number of the WNN's hidden layers needed increases exponentially for high dimension problems, an alternative type of WNN has been proposed, namely the Local Linear Wavelet Neural Network (LLWNN), in which the connection weights between the hidden layer units and output units are replaced by a local linear model. The standard training algorithm for WNNs is the gradient descent. An alternative training algorithm is the Particle Swarm Optimization [3] which has been successfully applied in neural network training and showed promising results [4].

The paper is organized as follows. In section II the architecture of the LLWNN is explained, in section III the training algorithm and the adaptations we developed are analyzed, in section IV are our experiments and their results and finally concluding remarks are given in section V.

II. LOCAL LINEAR WAVELET NEURAL NETWORK

The architecture of the LLWNN as proposed in [5] is shown in Fig. 1. Its output is given by

$$y = \sum_{i=1}^{M} (\omega_{i0} + \omega_{i1}x_1 + \cdots + \omega_{in}x_n)|a_i|^{-\frac{1}{2}}\psi\left(\frac{x - b_i}{a_i}\right) \quad (1)$$

where $x = \{x_1, x_2, \ldots, x_n\}$ are the inputs. It is important to note the use of a linear model, $u_i = \omega_{i0} + \omega_{i1}x_1 + \cdots + \omega_{in}x_n$, instead of a straightforward weight. For an n-dimensional input space, the multivariate wavelet function can be calculated by

$$\psi(x) = \prod_{i=1}^{n} \psi(x_i) \quad (2)$$

The scale and translation parameters along with the local linear model parameters (weight) are initialized randomly at start and are optimized by the learning algorithm analyzed in section III.

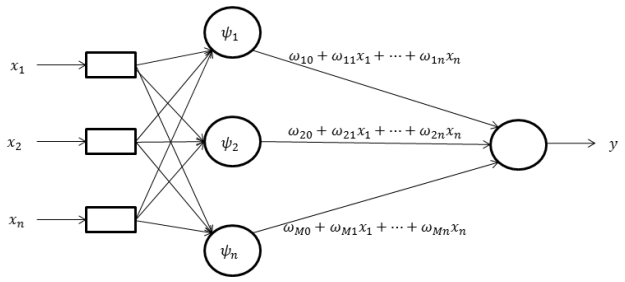

Figure 1. Architecture of the Local Linear Wavelet Neural Network

III. LEARNING ALGORITHM

A. Basic PSO Model

The PSO algorithm is an iterative evolutionary algorithm that conducts searches using a population of randomly initialized particles [6, 7]. Each particle represents a potential solution and has a position vector $\vec{p}_i$ and a velocity vector $\vec{v}_i$. In each iteration, a fitness function f_i representing the quality of a solution is calculated for each particle using $\vec{p}_i$ as input. Each particle stores the best position $\vec{b}_i$, corresponding to the best fitness function it has achieved. The best position accomplished of all particles in the swam is also stored as $\vec{b}_g$. The velocity of all particles is calculated in each iteration t by

$$\vec{v}_i(t+1) \leftarrow w * \vec{v}_i(t) + c_1 * \vec{U}(0,1) * \left(\vec{b}_i - \vec{p}_i\right) + c_2 * \vec{U}(0,1) * \left(\vec{b}_g - \vec{p}_i\right) \quad (3)$$

The parameter w called the inertia weight controls the magnitude of the old velocity, parameters c_1, c_2 are positive constants and U(0,1) is a random number in the [0,1]. According to this equation, particles are searching around their individual best position and the global best position. Each particle updates its position according to equation

$$\vec{p}_i \leftarrow \vec{p}_i + \vec{v}_i \quad (4)$$

B. Adaptive PSO

The main disadvantage of the classic PSO approach is that the particles tend to cluster too closely. When a local optimum is found, particles are drawn towards it with little chance to escape. The possibility of converging to local optima is very high and severely limits the exploration capability of the swarm. It would be desirable to let some particles escape and explore other areas.

In this direction we propose a mixed model of a special reinitializing PSO combined with a particle velocity derived from a probability density function. A similar probability density function was first presented in [5] and was appropriately adjusted for the purpose of this work. In each iteration a percentage of the total particles have their position randomly reinitialized in an area around the global best position achieved by the swarm and their velocity updated according to a probability density function.

The velocity vector is derived from (5) where r_i is a random number in [0, 1] and the value of β varies in each iteration according to (6) and ranges in the [0.6-0.05] area. The $notFound$ variable is normally equal to 1 but becomes $1/2$ if no better fitness value has been obtained for a number of iterations and will revert back to 1 when one is obtained. k is the current iteration and max_{iter} is the total number of iterations. According to (6), when the end number of iterations is approaching or no new solutions are being found, β takes small values. This is desirable as small values of β will lead to bigger values of $\vec{v}_i$ according to (5) and thus force particles to search a lot further than the local optima.

$$\vec{v}_i = \begin{cases} \frac{1}{\beta}\ln\left(\frac{r_i}{0.5}\right) if\ 0 < r_i \leq 1 - q_i \\ -\frac{1}{\beta}\ln\left(\frac{1 - r_i}{0.5}\right)\ if\ 1 - q_i \leq r_i < 1 \end{cases} \quad (5)$$

$$\beta = \frac{notFound * (0.2 + \frac{max_{iter} - k}{max_{iter}})}{2} \quad (6)$$

After calculating the velocity vector, the new positions are updated according to (7) where k is the current iteration. The number of particles that follow this strategy vary according to the remaining iterations of the algorithm and the rate of discovery of better fitness values. When no new solution is found for some time the algorithm takes a more aggressive approach and assigns more particles to this strategy.

$$\vec{p}_i = \vec{b}_g \pm U(0,1) * \frac{k}{max_{iter}} + \vec{v}_i \quad (7)$$

C. Parallel Implementation

Dynamically changing environments require frequent retraining of the neural network so as to reflect the most recent information. The PSO algorithm has inherent parallelization as particles can calculate their fitness value and update their positions in parallel.

We chose to utilize the massively parallel CUDA architecture to implement a very fast PSO training algorithm for the LLWNN. A kernel is launched where each particle is assigned to a CUDA block. In each block, the training set is divided among a number of threads which calculate the forecasting error. By summing these we get the total error

which is the fitness function of the particle. Blocks are executed in parallel. After calculating the fitness value of all particles the kernel terminates and a second kernel is launched that will find the best fitness value of all the particles and its corresponding position, the new global best. Finding the minimum value is a linear complexity problem and the calculations involved are trivial even for a relatively high number of particles (up to 1024). Standard CUDA parallel reduction routines have been used for this step. After identifying the global best position a third and final kernel will be launched where the particles will update their positions in parallel be each being assigned to a CUDA block.

The particle's velocity and present and best position along with the training set are kept in the block's shared memory allowing for very fast access times compared with the global memory and thus fast calculations. Due to the low memory requirements, a multiprocessor's shared memory isn't saturated and thus it can accommodate its maximum number of blocks. The number of threads on each block is selected in accordance with CUDA limitations so as to maximize multiprocessor occupancy. If the training set or the particle's dimensions cannot be accurately divided to the number of threads, then the remainders are assigned unequally to the existing threads. The memory occupancy and the apportionment of CUDA resources explained above are best illustrated in Fig.2.

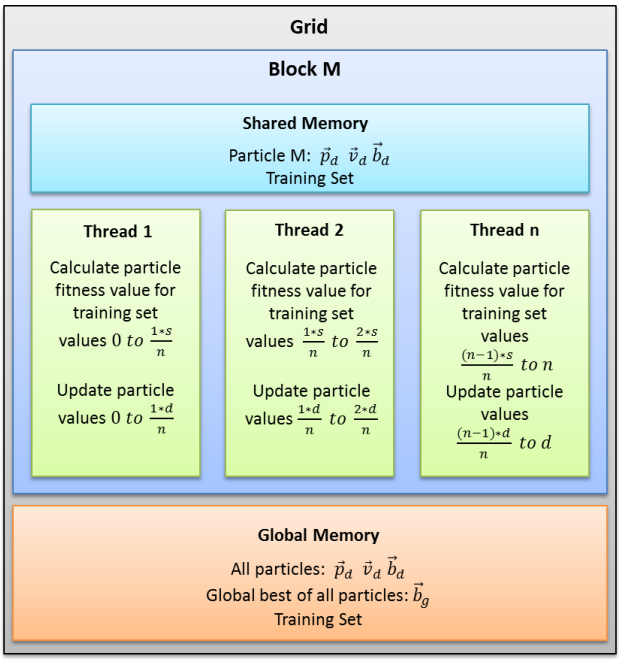

Figure 2. Cuda Resources allocation and calculation model. M is the number of Particles, N is the number of threads, s is the training set size and d is the dimension of the particle.

IV. EXPERIMENTS

The efficiency of our approach is evaluated by forecasting load and day ahead MCP in real Energy Markets and in a simulated environment by being a part of a PowerTac agent.

The number of inputs of the LLWNN is 48 of which the 24 correspond to the last 24 hours and the other 24 are the hours of the "to be forecasted" day of the previous week. The number of PSO particles is 1024 and the number of iterations is 20000. The training set is 3 weeks and the test set is the following 2 weeks. The mother wavelet function used in all the experiments is

$$\psi(x) = \frac{-x^2}{2} e^{-x^2/\sigma^2}$$

where σ is the standard deviation of the training set. We adopted the following, widely used accuracy measures.

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|F_t - A_t|}{F_t + A_t}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2}$$

In all the time series forecasting figures displayed below, the actual value is presented by the green line, the forecasted value by the red line and the error, which is the difference of the actual and the forecasted value, by the blue line.

A. Classic and Adaptive PSO Fitness values achieved

According to the above LLWNN and PSO setup the RMSE error of the load forecasting experiment detailed in section D is measured across the different PSO variations. Each PSO variation was executed 50 times and the max, mean and min RMSE calculated are shown in table II. As is apparent, our approach is superior to the classic PSO. Our model shows consistently good results and a small variation between high and low error rates.

TABLE I. COMPARISON BETWEEN PSO VARIATIONS

RMSE	Fitness Value Achieved	
	Classic PSO	***Adaptive PSO***
Max	458.60	136.23
Mean	291.30	125.44
Min	180.04	120.28

B. Runtime of the CUDA Parallel Implementation

According to the same setup again we conducted experiments to measure the performance increase between a parallel implementation in CUDA, in an 8 CPU thread environment and a single CPU thread environment. The tests were conducted on an i7 4770k CPU and an NVidia GTX 970 and results are shown in table. The CUDA implementation is massively faster than both the multithreaded and single threaded CPU. We can conclude that our PSO model is an ideal candidate to be implemented in a parallel architecture.

TABLE II. COMPARISON OF RUNTIME AMONG IMPLEMENTATIONS

Runtime	Implementations		
	GTX 970	***CPU 8 threads***	***CPU 1 thread***
seconds	78	1003	3321

C. *Load Forecasting in the Greek Energy Market*

For the Load Forecasting experiments, 3 time periods of the Greek Energy Market in year 2010 were selected. Each time period represents different seasonal patterns and variations. A daily and weekly periodicity is observable along with some random inconsistencies. In table III we can see the hourly error for multiple hour forward forecasts. The SMAPE error gradually increases in accordance to the forecasting timeframe but remains close to the 2% daily average. The winter and summer periods seem more unpredictable than the autumn, mainly because of the more extreme weather conditions that create spikes in energy demand. In Fig. 3 and 4 we can see the actual and forecasted load and the error for the summer period for a 1 and a 24 hour forward forecast. It is evident that in the 1 hour forward forecast the predicted load almost identically matches the actual while in the 24 hour forward forecast the predicted load can accurately follow the actual but misses some of the highs and lows.

TABLE III. LOAD FORECASTING TEST SET RESULTS

Location	*Hours Forw.*	*SMAPE*		*RMSE*
		Mean	*Max*	
Greece Winter 2010 Weeks 10-11	1	1.23%	6.86%	181.62
	8	2.36%	12.21%	347.04
	16	3.33%	12.34%	474.16
	24	3.40%	14.24%	469.13
Greece Summer 2010 Weeks 27-28	1	0.75%	4.06%	133.18
	8	2.03%	7.98%	353.63
	16	7.66%	2.71%	459.66
	24	2.91%	9.16%	503.01
Greece Autumn 2010 Weeks 46-47	1	0.73%	2.91%	102.35
	8	1.06%	4.38%	151.45
	16	1.08%	4.93%	152.68
	24	1.20%	4.66%	171.83

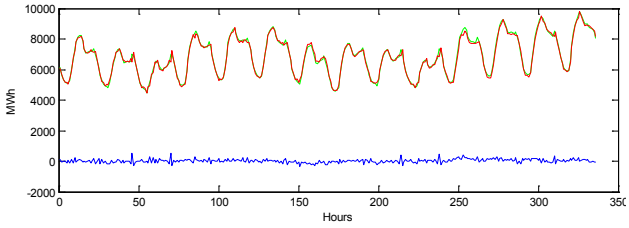

Figure 3. Greek Energy Market Summer 2010 Weeks 27-28, 1 hour forward actual (green) and forecasted (red) load values

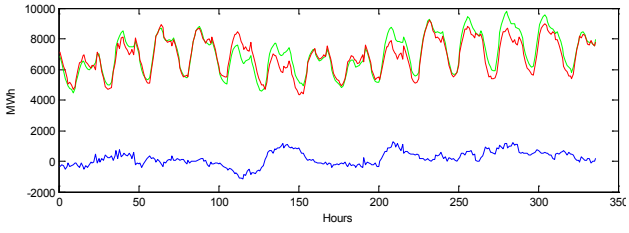

Figure 4. Greek Energy Market Summer 2010 Weeks 27-28, 24 hour forward actual (green) and forecasted (red) load values

D. *Price forecasting in ISO-NE Main 2010*

For the day ahead MCP forecasting experiments data were selected from New York (NYISO), Dominion(PJM), Maine(ISO-NE) and Houston(ERCOT) in different times of the year. The price time series here displays some daily and weekly periodicity though not as strong. Many random inconsistencies and rapid changes are noted, typical of a price time series. In table IV is the hourly error for multiple hour forward forecasts. The best results are obtained in the spring and autumn periods in Maine and Houston. The maximum error is met in New York during the summer probably due to spikes caused by the intense heat. In Fig. 5 and 6 we can see the actual and forecasted price and the error for a 1 and a 24 hour forward forecast in Maine. The highs and lows here are very prominent and especially in the 24 hour forward forecast, impossible to catch. This is to be expected for a price forecast.

TABLE IV. PRICE FORECASTING TEST SET RESULTS

Location	*Hours Forw.*	*SMAPE*		*RMSE*
		Mean	*Max*	
New York Summer 2012 Weeks 29-30	1	20.02%	4.21%	9.93
	8	10.42%	50.12%	30.91
	16	12.59%	55.52%	35.08
	24	11.04%	64.68%	32.13
Dominion Spring 2007 Weeks 24-25	1	3.06%	13.58%	4.71
	8	7.22%	28.96%	12.20
	16	8.82%	33.24%	15.95
	24	10.23%	28.85%	18.04
Maine Spring 2010 Weeks 20-21	1	3.04%	20.51%	4.56
	8	5.06%	20.34%	7.01
	16	4.89%	19.23%	6.69
	24	5.55%	29.75%	7.45
Houston Autumn 2012 Weeks 41-42	1	3.78%	16.02%	2.77
	8	4.87%	21.93%	3.91
	16	4.82%	19.50%	3.89
	24	4.94%	21.15%	4.24

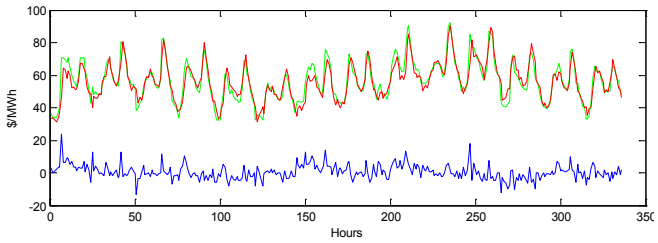

Figure 5. Main, Spring 2010 Weeks 20-21, 1 hour forward actual (green) and forecasted (red) price values

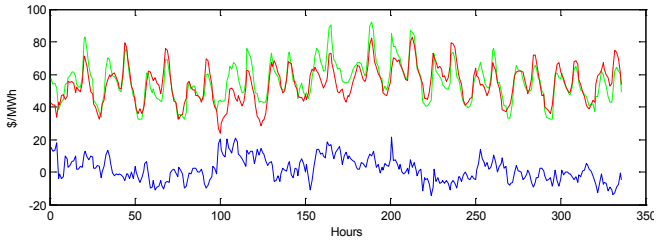

Figure 6. Maine, Spring 2010 Weeks 20-21 24 hour forward actual (green) and forecasted (red) price values

E. PowerTac Competition experiments

PowerTAC [8] is a competitive simulation platform, which provides a promising ground for conducting research on Tariff Markets. In particular, it models a liberalized energy market where competing broker agents try to maximize profit by trading energy in the wholesale and the retail market.

Concerning the PowerTac competition experiments our goal was to increase the profitability of Aristotle's University Electrical and Computer Engineering agent called Mertacor [9, 10] by forecasting electricity prices in the wholesale energy market and buying the energy to cover the agent's demand in the most competitive price.

In the PowerTAC Environment, for every timeslot, energy can be bought in any of the 24 auctions taking place in the preceding 24 timeslots. The goal is to buy the energy the agent needs at the most profitable combination of prices. To utilize this, a simple strategy was adopted. Using our forecasting methodology we attempted to forecast the mean clearing price of the 24 auctions of a given timeslot in the future (usually 24 timeslots forward) by using as input the mean prices of past timeslots. By forecasting the expected mean clearing price of a 24 timeslots forward timeslot the agent can make intelligent decisions by attempting to bid a little lower than the mean price at first and gradually increase the bids in case the opportunity to buy is running out and the energy needed has not been covered. In spite of being a naïve strategy, it tends to make a makeable difference in the agent's profitability.

The experiment conducted consists of an enhanced Mertacor agent utilizing our methodology competing against the exact same agent without our methodology and the default PowerTac broker. In table VI we can see the forecasting error of a 24 timeslots forward forecast. In table V we can see that the profitability of the enhanced agent increased by almost 80%, proving the efficiency of our system in a different from the classic forecasting application, albeit in a simulated environment.

TABLE V. POWERTAC AGENT PROFITABILITY

Agents	**Profitability**	
	Mertacor	***Enhanced Mertacor***
Profit in $	1,970,528	3,433,582

TABLE VI. POWERTAC PRICE FORECASTING TEST SET RESULTS

Location	***Hours Forw.***	***SMAPE***		***RMSE***
		Mean	***Max***	
PowerTAC	24	4.74%	33.41%	3.34

V. CONCLUSION

In this paper we developed a massively parallel training algorithm based on the classic PSO to train a LLWNN for the purpose of load and day ahead MCP forecasting in electricity markets. As no standard forecasting benchmark exists it is difficult to directly compare with other suggested models. Nevertheless, our approach provides a relatively low error for both load and price forecasting up to par if not better than most computational intelligence and machine learning works in the field as presented in [1, 11]. In addition, our learning algorithm is implemented in parallel in the CUDA architecture achieving an order of magnitude better training time and thus presenting a robust solution to real time dynamic systems that require frequent model retraining and adaptation. This is a novel addition to the PowerTAC trading agent and can also prove valuable to any real time agent attempting time series forecasting.

REFERENCES

[1] R. Weron, "Electricity price forecasting: A review of the state-of-the-art with a look into the future," Hugo Steinhaus Center, Wroclaw University of Technology2014.

[2] W. Ting and Y. Sugai, "A wavelet neural network for the approximation of nonlinear multivariable function," in *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, 1999, pp. 378-383.

[3] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the sixth international symposium on micro machine and human science,* vol. 1, pp. 39-43, 1995.

[4] Y. Song, Z. Chen, and Z. Yuan, "New chaotic PSO-based neural network predictive control for nonlinear process," *Neural Networks, IEEE Transactions on,* vol. 18, pp. 595-601, 2007.

[5] Y. Chen, B. Yang, and J. Dong, "Time-series prediction using a local linear wavelet neural network," *Neurocomputing,* vol. 69, pp. 449-465, 2006.

[6] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," 2002.

[7] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, 1998, pp. 69-73.

[8] J. Babic and V. Podobnik, "An Analysis of Power TAC 2013 Trial," in *AAAI Workshop: Trading Agent Design and Analysis*, 2013.

[9] T. G. Diamantopoulos, A. L. Symeonidis, and A. C. Chrysopoulos, "Designing Robust Strategies for Continuous Trading in Contemporary Power Markets," in *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, ed: Springer, 2013, pp. 30-44.

[10] E. Ntagka, A. Chrysopoulos, and P. A. Mitkas, "Designing Tariffs in a Competitive Energy Market using Particle Swarm Optimization Techniques."

[11] A. Ahmad, M. Hassan, M. Abdullah, H. Rahman, F. Hussin, H. Abdullah*, et al.*, "A review on applications of ANN and SVM for building electrical energy consumption forecasting," *Renewable and Sustainable Energy Reviews,* vol. 33, pp. 102-109, 2014.