Evaluation of Differential Evolution and Particle Swarm Optimization Algorithms at Training of Neural Network for Stock Prediction

Hatem Abdul- Kader [1] and Mustafa Abdul Salam [2]
[1] Faculty of Computer and Information, Menoufiya University, Egypt
[2] Higher Technological Institute, Egypt

Abstract: *This paper presents the comparison of two meta-heuristic approaches: Differential Evolution (DE) and Particle Swarm Optimization (PSO) in the training of feed-forward neural network to predict the daily stock prices. Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on a financial exchange. The successful prediction of a stock's future price could yield significant profit. The feasibility, effectiveness and generic nature of both DE and PSO approaches investigated are exemplarily demonstrated. Comparisons will be performed between the two approaches in terms of the prediction accuracy and convergence characteristics. The proposed model is based on the study of historical data, technical indicators and the application of Neural Networks trained with DE and PSO algorithms. Results presented in this paper show the potential of both algorithms applications for the decision making in the stock markets, but DE gives better accuracy compared with PSO.*

Keywords: *Evolutionary algorithms, Differential evolution, Particle swarm optimization, feed-forward neural network, technical indicators, and stock prediction.*

Received February 21, 2010; Accepted March 20, 2011

1. Introduction

STOCK price prediction has been at focus for years since it can yield significant profits. Predicting the stock market is not a simple task, mainly as a consequence of the close to random-walk behavior of a stock time series. Fundamental and technical analysis was the first two methods used to forecast stock prices. Neural networks are the most commonly used technique [6]. The role of artificial neural networks in the present world applications is gradually increasing and faster algorithms are being developed for training neural networks [10]. In general, back-propagation is a method used for training neural networks. Gradient descent, conjugate gradient descent, resilient, BFGS quasi-Newton, one-step secant, Levenberg-Marquardt and Bayesian regularization are all different forms of the back-propagation training algorithm. For all these algorithms storage and computational requirements are different, some of these are good for pattern recognition and others for function approximation but they have drawbacks in one way or other, like neural network size and their associated storage requirements. Certain training algorithms are suitable for some type of applications only, for example an algorithm that performs well for pattern recognition may not for classification problems and vice versa, in addition

some cannot cater for high accuracy/performance. It is difficult to find a particular training algorithm that is the best for all applications under all conditions all the time [24]. The perceived advantages of evolution strategies as optimization methods motivated the authors to consider such stochastic methods in the context of training artificial neural networks and optimizing the structure of the networks [1]. A survey and overview of evolutionary algorithms in evolving artificial neural networks can be found in [30].

Differential evolution (DE) is introduced by Kenneth Price and Rainer Storn in 1995. DE algorithm is like genetic algorithms using similar operators; crossover, mutation and selection. DE can find the true global minimum regardless of the initial parameter values. The main difference in constructing better solutions is that genetic algorithms rely on crossover while DE relies on mutation operation. DE is successfully applied to many artificial and real optimization problems and applications, such as aerodynamic shape optimization [32], automated mirror design [24], optimization of radial active magnetic bearings [12], and mechanical engineering design [23]. A differential evolution based neural network training algorithm, introduced in [9, 23, and 30]. PSO is proposed algorithm by James Kennedy and Russell Eberhart in 1995, motivated by social behavior of organisms such as bird flocking and fish schooling [28]. The main difference of particle swarm optimization concept from the evolutionary computing is that flying potential solutions through hyperspace are accelerating toward "better" solutions, while in evolutionary computation schemes operate directly on potential solutions which are represented as locations

in hyperspace [13]. Neural networks are used in combination with PSO in many applications, like neural network control for nonlinear processes in [23], feedforward neural network training in [3, 5, 15, 16, 27, 32, 33]. PSO algorithm is used in prediction and forecasting in many applications, like prediction of chaotic systems in [22], electric load forecasting in [25, 26], time series prediction in [2, 29, 31] and stock market decision making in [18, 19, 20].

The main Purpose of this paper is to propose of using two modern Artificial intelligence techniques in neural network training. Stock predication is selected as an application to evaluate the efficiency of the both proposal training Algorithms for its complexity and sensitivity.

The paper is organized as follows: Section 2 presents the Differential Evolution algorithm; Section 3 presents Particle swarm optimization algorithm; Section 4 is devoted for the proposed system and implementation of Differential Evolution and particle swarm optimization algorithms in stock prediction; In Section 5 the results are discussed. The main conclusions of the work are presented in Section 6.

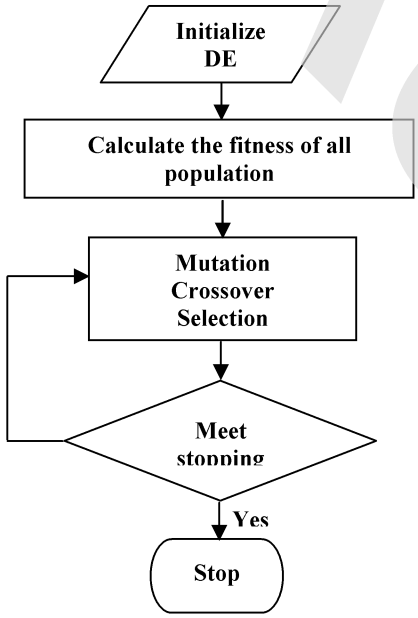

Figure 1. The DE algorithm.

2. Differential Evolution Training Algorithm

Price and Storn developed DE to be a reliable and versatile function optimizer. The first written publication on DE appeared as a technical report in 1995 (Price and Storn 1995). Like nearly all EAs, DE is a population-based optimizer that attacks the starting point problem by sampling the objective function at multiple, randomly chosen initial points. [11]. DE algorithm like genetic algorithms using similar operators; crossover, mutation and selection. DE has three advantages; finding the true global minimum regardless of the initial parameter values, fast convergence, and using few control parameters. The main difference in constructing better solutions is that genetic algorithms rely on crossover while DE relies on mutation operation. This main operation is based on the differences of randomly sampled pairs of solutions in the population. The algorithm uses mutation operation as a search mechanism and selection operation to direct the search toward the prospective regions in the search space. The DE algorithm also uses a non-uniform crossover that can take child vector parameters from one parent more often than it does from others. By using the components of the existing population members to construct trial vectors, the recombination (crossover) operator efficiently shuffles information about successful combinations, enabling the search for a better solution space [8]. The DE algorithm is shown in figure 1.

2.1. Population Structure

The current population, symbolized by Px, is composed of those vectors, xi,g, that have already been found to be acceptable either as initial points, or by comparison with other vectors:

$$Px,g = (xi,g)\ ,\ i=0,1,\ldots,Np-1,\ g=0,1,\ldots,gmax.$$
$$xi,g=(xj,i,g),\ j=0,1,\ldots,D-1 \quad (1)$$

Once initialized, DE mutates randomly chosen vectors to produce an intermediary population, Pv,g, of Np mutant vectors, Vi,g:

$$Pv,g=(Vi,g),\ i=0,1,\ldots,Np-1,\ g=0,1,\ldots,gmax.$$
$$Vi,g=(Vj,i,g),\ j=0,1\ldots D-1 \quad (2)$$

Each vector in the current population is then recombined with a mutant to produce a trial population, Pu, of Np trial vectors, ui,g:

$$Pu,g = (ui,g),\ i=0,1,\ldots,Np-1,\ g=0,1,\ldots,gmax,$$
$$ui,g = (uj,i,g),\ j=0,1,\ldots,D-1 \quad (3)$$

During recombination, trial vectors overwrite the mutant population, so a single array can hold both populations.

2.2. Initialization

Before the population can be initialized, both upper and lower bounds for each parameter must be specified. These 2D values can be collected into two, D-dimensional initialization vectors, bL and bU. Once initialization bounds have been specified, a random number generator assigns each parameter of every vector a value from within the prescribed range. For example, the initial value (g = 0) of the jth parameter of the ith vector is

$$xj,i,0=randj(0,1).(bj,U-bj,L)+bj,L. \quad (4)$$

2.3. Mutation

Once initialized, DE mutates and recombines the population to produce a population of Np trial vectors. In particular, differential mutation adds a scaled, randomly sampled, vector difference to a third vector.

$$Vi,g=xr0,g+F.(xr1,g- xr2,g) \quad (5)$$

The scale factor, F ∈ (0,1+), is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F, effective values are seldom greater than 1.0.

2.4. Crossover

To complement the differential mutation search strategy, DE also employs uniform crossover. Sometimes referred to as discrete recombination, (dual) crossover builds trial vectors out of parameter values that have been copied from two different vectors. In particular, DE crosses each vector with a mutant vector:

$$ui,g=(uj,i,g)=\begin{cases} v_{j,i,g} & \text{if } rand_j(0,1) \le Cr \\ & j=j_{rand} \\ x_{j,i,g} & \text{otherwise} \end{cases} \quad (6)$$

The crossover probability, Cr ∈ [0,1], is a user-defined value that controls the fraction of parameter values that are copied from the mutant.

2.5. Selection

If the trial vector, ui,g, has an equal or lower objective function value than that of its target vector, xi,g, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation

$$xi,g+1=\begin{cases} u_{i,g} & \text{if } f(u_{i,g}) < f(x_{i,g}) \\ & \\ x_{i,g} & \text{otherwise} \end{cases} \quad (7)$$

Once the new population is installed, the process of mutation, recombination and selection is repeated until the optimum is located, or a prespecified termination criterion is satisfied, e.g., the number of generations reaches a preset maximum, gmax [12].

3. Particle Swarm Optimization Algorithm

PSO is a relatively recent heuristic search method which is derived from the behavior of social groups like bird flocks or fish swarms. PSO moves from a set of points to another set of points in a single iteration with likely improvement using a combination of deterministic and probabilistic rules. The PSO has been popular in academia and industry, mainly because of its intuitiveness, ease of implementation, and the ability to effectively solve highly nonlinear, mixed integer optimization problems that are typical of complex engineering systems. Although the "survival of the fittest" principle is not used in PSO, it is usually considered as an evolutionary algorithm. Optimization is achieved by giving each individual in the search space a memory for its previous successes, information about successes of a social group and providing a way to incorporate this knowledge into the movement of the individual. Therefore, each individual (called particle) is characterized by its position $\vec{x}_i$, its velocity $\vec{v}_i$, its personal best position $\vec{p}_i$ and its neighborhood best position $\vec{p}_g$.

The elements of the velocity vector for particle i are updated as

$$\upsilon_{ij} \leftarrow \omega\upsilon_{ij} + c_1 q(x_{ij}^{pb} - x_{ij}) + c_2 \gamma(x_j^{sb} - x_{ij}), j = 1,.., \quad (8)$$

Where w is the inertia weight, x_i^{pb} is the best variable vector encountered so far by particle i, and x^{sb} is the swarm best vector, i.e. the best variable vector found by any particle in the swarm, so far. c_1 and c_2 are constants, and q and r are random numbers in the range [0, 1]. Once the velocities have been updated, the variable vector of particle i is modified according to

$$x_{ij} \leftarrow x_{ij} + \upsilon_{ij}, j = 1,...,n. \quad (9)$$

The cycle of evaluation followed by updates of velocities and positions (and possible update of x_i^{pb} and x^{sb}) is then repeated until a satisfactory solution has been found. PSO algorithm is shown in Figure 2.

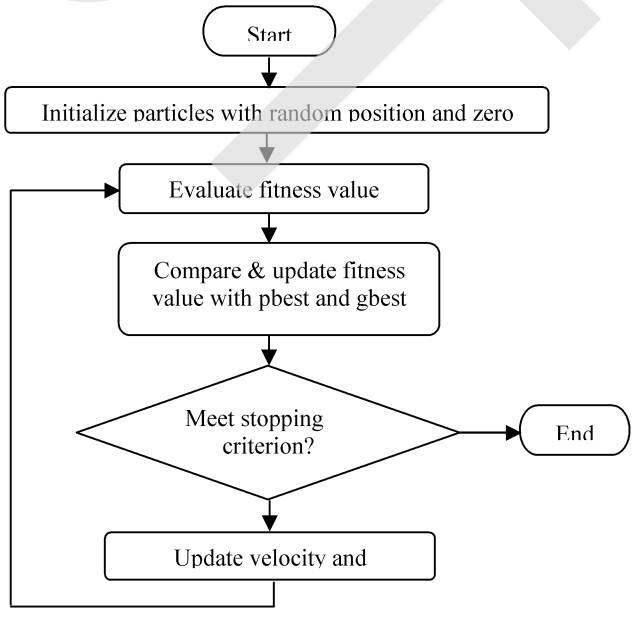

Figure 2. PSO algorithm.

4. The Proposed Model

The proposed methodology is to train multilayer feed forward neural network with Differential Evolution (DE) algorithm and also Particle Swarm Optimization algorithm to be used in the prediction of daily stock prices. The proposed model is based on the study of historical data, technical indicators and the application of Neural Networks trained with DE and PSO algorithms. Neural network architecture contains one input layer with six inputs neurons represent the historical data and derived technical indicators, one hidden layers and single output layer as shown in figure 3.

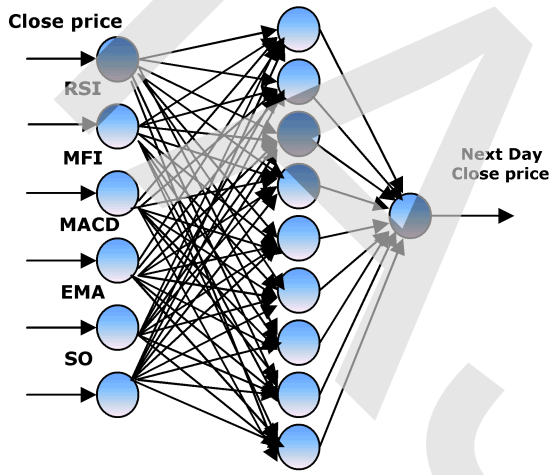

Figure 3. Architecture of neural network of proposed model.

The two algorithms were tested for many companies which cover different stock sectors, like Drug Manufacturers, Industries, Utilities, Communications, life science and Automotives. These companies are Acadia Pharmaceuticals Inc. (ACAD), Shiloh Industries Inc. (SHLO), FiberTower Corporation (FTWR), Hayes Lemmerz International Inc. (HAYZ), Strategic Internet (SIII.OB) , Caliper Life Sciences, Inc. (CALP) and Ford.

Five technical indicators are calculated from the raw datasets for neural networks inputs:

- *Relative Strength Index (RSI)*: A technical momentum indicator that compares the magnitude of recent gains to recent losses in an attempt to determine overbought and oversold conditions of an asset. The formula for computing the Relative Strength Index is as follows.

$$RSI = 100- [100 / (1+RS)] \quad (10)$$

Where RS = Avg. of x days' up closes / Average of x days' down closes.

- *Money Flow Index (MFI):* This one measures the strength of money in and out of a security. The formula for MFI is as follows.

$$\text{Money Flow (MF)} = \text{Typical Price} * \text{Volume}. \quad (11)$$

$$\text{Money Ratio (MR)} = (\text{Positive MF} / \text{Negative MF} \quad (12)$$

$$MFI = 100 – (100/ (1+MR)). \quad (13)$$

- *Exponential Moving Average (EMA):* This indicator returns the exponential moving average of a field over a given period of time. EMA formula is as follows.

$$EMA = [\alpha * \textit{Today's Close}] + [1-\alpha* \text{Yesterday's EMA}]. \quad (14)$$

- *Stochastic Oscillator (SO):* The stochastic oscillator defined as a measure of the difference between the current closing price of a security and its lowest low price, relative to its highest high price for a given period of time. The formula for this computation is as follows.

$$\%K = [(\text{Close price} – \text{Lowest price}) / (\text{Highest Price} – \text{Lowest Price})] * 100 \quad (15)$$

- *Moving Average Convergence/Divergence (MACD):* This function calculates difference between a short and a long term moving average for a field. The formulas used for calculating MACD and its signal as follows.

$$MACD = [0.075*\text{EMA of Closing prices}] – [0.15*\text{EMA of closing prices}] \quad (16)$$

$$\text{Signal Line} = 0.2*\text{EMA of MACD} \quad (17)$$

5. Results And Discussion

Neural networks are trained and tested with datasets form September 2004 to September 2007. All datasets are available on http://finance.yahoo.com web site. Datasets are divided into training part (70%) and testing part (30%). The used software is Matlab and Microsoft excel.

Figures (4-10) outlines the application of different training algorithms at different data sets with different sectors of the market. The used data sets present different market trends. In figures (4, 7) which present results of the two algorithms on two different sectors which are Pharmaceuticals and utilities sectors; one can remark that the predicted curve using both DE and PSO algorithms give a good accuracy with a little advance to DE algorithm and the datasets are not fluctuated.

Figures (5, 8, 9) outline the application of the two algorithms on a different market sectors. From figures one can remark the enhancement in the error rate achieved by the DE algorithm.

Figures 6, 10 outlines different time series with changing trends. The DE can easily cope up with the fluctuation existing in the time series better than PSO algorithm.

Performance will be evaluated as the weight sum of two factors: the mean squared error and the mean

squared weights and biases; mean square error (MSE) and mean absolute error (MAE) performance functions. It can be remarked that the DE always gives an advance over the PSO algorithms in all performance functions and in all trends and sectors. DE performs better than PSO especially in cases with fluctuations in the time series function.

Table 1. Error functions for the two algorithms.

Error / Company	MSERG		MSE		MAE	
	DE	**PSO**	**DE**	**PSO**	**DE**	**PSO**
ACADIA	0.57	1.50	0.49	1.46	0.46	0.90
SHLO	1.04	1.43	1.01	1.39	0.80	0.95
FTWR	0.17	0.48	0.16	0.35	0.30	0.49
HYZ	0.06	0.19	0.05	0.11	0.17	0.25
NET	0.71	2.27	0.65	1.94	0.62	1.14
CALPER	0.05	0.16	0.02	0.09	0.12	0.25
FORD	0.34	3.10	0.22	3.09	0.39	1.30

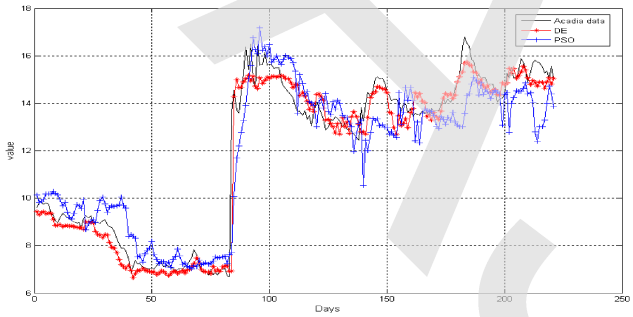

Figure 4. Results for Acadia Pharmaceuticals Company.

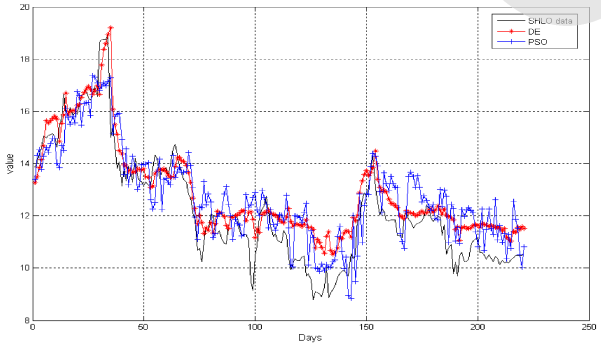

Figure 5. Results for Shiloh Industries Company.

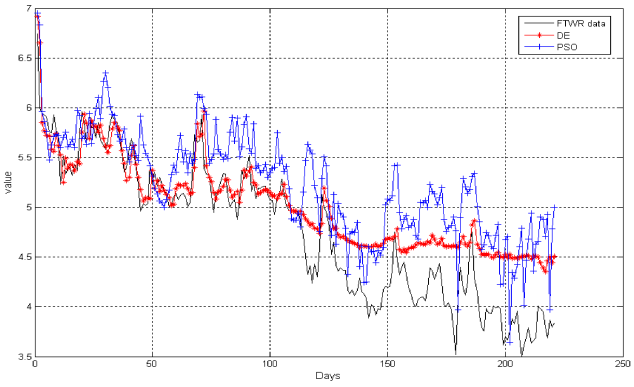

Figure 6. Results for Fiber Tower Company.

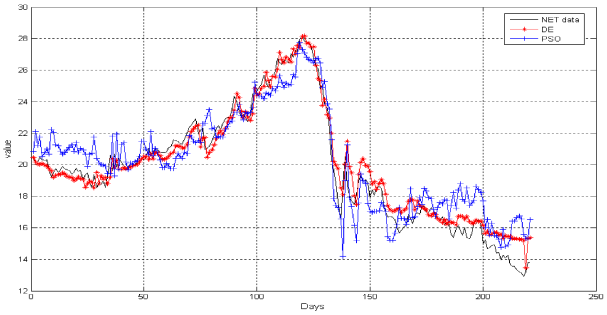

Figure 7. Results for Strategic Internet (SIII.OB) Company.

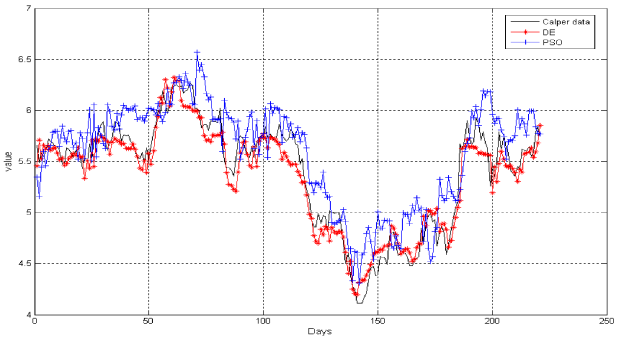

Figure 9. Results for Caliper Life Sciences Company.

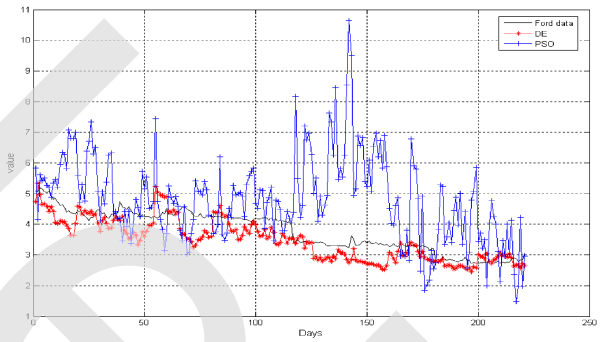

Figure 10. Results for Ford Company.

6. Conclusions

In this paper, Differential Evolution (DE) and Particle Swarm Optimization (PSO) algorithms are proposed for training and testing feed-forward neural network for stock price prediction. The simulation results show the potential of both two algorithms. Both algorithms are convergence to a global minimum. This proposal improves the performance of feedforward neural network. Both algorithms can be used to avoid local minima problem which all gradient descending methods fall on it. The main advantage of both algorithms is the ability of tuning algorithms parameters. But DE converges to global minimum faster than PSO algorithm. DE algorithm gives better accuracy than PSO algorithm especially in fluctuated time series. This indicates that stock predication is more accurate if the feedforward neural network with DE training is used.

References

[1] Al-kazemi B. and Mohan, C.K. "Training feedforward neural networks using multi-phase particle swarm optimization," *Proc. 9th International Conference on Neural Information*, pp 2615 – 2619, 2002.

[2] Cai X., Zhang N., Venayagamoorthy G.K., and II Wunsch D.C. "Time series prediction with recurrent neural networks using a hybrid pso-ea algorithm," *Proc. IEEE International Joint Conference in Neural Networks*, pp 1647 – 1652, 2004.

[3] Carvalho M. and Ludermir T.,"Particle swarm optimization of feed-forward neural networks with weight decay," *Sixth International Conference on Hybrid Intelligent System HIS*, pp5 – 5, 2006.

[4] Chen Y., Dong J., Yang B., and Zhang Y., "A local linear wavelet neural network," *Fifth World Congress In Intelligent Control and Automation, WCICA*, pp 1954 – 1957, 2004.

[5] Chunkai Z., Yu L., and Huihe S., "A new evolved artificial neural network and its application. In Intelligent Control and Automation," *Proc. of the 3rd World Congress on*, pp 1065 – 1068, 2000.

[6] Coupelon O., "Neural network modeling for stock movement prediction ", *Neural Network Modeling For Stock Movement Prediction*, 2007.

[7] Deng C., Wei X., and Guo L., "Application of neural network based on pso algorithm in prediction model for dissolved oxygen in fishpond," *The Sixth World Congress In Intelligent Control and Automation, WCICA*, pp 9401 – 9405, 2006.

[8] Firpi H. and Goodman E., "Designing templates for cellular neural networks using particle swarm optimization," *Proc. In Applied Imagery Pattern Recognition Workshop*, 33rd, pp 119 – 123, 2004.

[9] Guerra F., et al, "Radial basis neural network learning based on particle swarm optimization to multistep prediction of chaotic lorenz's system" *Fifth International Conference in Hybrid Intelligent Systems*, pp 3 ., 2005.

[10] Hagan M., Demuth H., and Beale M., "Neural Network Toolbox for Use with MATLAB," 2006.

[11] Juang C. and Liou Y., "On the hybrid of genetic algorithm and particle swarm optimization for evolving recurrent neural network," *Proc. IEEE International Joint Conference on Neural Networks*, pp 2285 – 2289 , 2004.

[12] Juang C., "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man and Cybernetics,* Part B, 34:997– 1006, 2004.

[13] Kennedy J., Spears W., "Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator," *proc.http://www.aic.nrl.navy.mil/%7Espears/papers/wcci98.pdf*, 2003.

[14] Khalil A., "An Investigation into Optimization Strategies of Genetic Algorithms and Swarm Intelligence," *Artificial Life*, 2001.

[15] Liu H., et al, "Neural networks learning using vbest model particle swarm optimization," *In Proc. International Conference on Machine Learning and Cybernetics*, pp 3157 – 3159, 2004.

[16] Mendes R., Cortez P., Rocha M., and Neves J., "Particle swarms for feedforward neural network training. In Neural Networks," *Proc. International Joint Conference on IJCNN,* pp 1895 – 1899, 2002.

[17] Nenortaite J. and Simutis R., "Application of Particle Swarm Optimization Algorithm to Stocks' Trading System," 2004.

[18] Nenortaite J. and Simutis R., "Stocks' Trading System Based on the Particle Swarm Optimization Algorithm," *Lecture Notes on Computer Science (J. Nenortaite, and R. Simutis)*, 2004.

[19] Nenortaitė J., "A particle swarm optimization approach in the construction of decision-making model," *information technology and control*, vol.36, no.1a, 2007.

[20] Pavlidis N., Tasoulis D., Vrahatis M.,"Financial Forecasting Through Unsupervised Clustering and Evolutionary Trained Neural Networks," *proc. Congress on Evolutionary Computation, Canberra Australia* , 2003.

[21] Peng J., Chen Y., and Eberhart R., "Battery pack state of charge estimator design using computational intelligence approaches," *proc. The Fifteenth Annual Conference in Battery on Applications and Advances*, pp 173 – 177, 2000.

[22] Reynolds P.D., Duren R.W., Trumbo M.L. and Marks R.J., "FPGA Implementation of particle swarm optimization for inversion of large neural networks," Proc. *IEEE Swarm Intelligence Symposium, SIS*, pages 389 – 392, 2005.

[23] Song Y., Chen Z., and Yuan Z., "New chaotic pso-based neural network predictive control for nonlinear process," *IEEE Transactions on Neural Networks*, pp18:595 –601, 2007.

[24] Su T., Jhang J. and Hou C., "A hybrid artificial neural networks and particle swarm optimization for function approximation," *ICIC International*, 2008.

[25] Sun C. and Gong D., "Support vector machines with pso algorithm for short-term load forecasting," *Proc. IEEE International Conference on Networking, Sensing and Control, ICNSC '06*, pp 676– 680, 2006.

[26] Sun W., Zhang Y., and Li F., "The neural network model based on pso for short-term load forecasting," *International Conference on Machine Learning and Cybernetics*, pp 3069 – 3072, 2006.

[27] Wang C., Zhou C. and Ma J., "An improved artificial fish-swarm algorithm and its application in feed-forward neural networks, "*Proc. International Conference on Machine Learning and Cybernetics*, pp 2890 – 2894, 2005.

[28] Wilke D., "Analysis of the particle swarm optimization algorithm," *University of Pretoria*, 2005.

[29] Yang Y., Chen R.S., Ye Z.B., and Liu Z., "Fdtd. time series extrapolation by the least squares supports vector machine method with the particle swarm optimization technique," *Proc. Conference in Microwave Conference APMC, Asia-Pacific*, pp 3, 2005.

[30] Yao X., "Evolving artificial neural networks," *proc. IEEE*, pp 1423- 1447, 1999.

[31] Zhang C., et al, "Particle swarm optimisation for evolving artificial neural network," *IEEE International Conference on Systems*, pp 2487 – 2490, 2000.

[32] Zhang C., et al, "Using pso algorithm to evolve an optimum input subset for a svm in time series forecasting," *IEEE International Conference on Systems, Man and Cybernetics*, pp 3793 – 3796, Vol. 4, 2005.

[33] Zhao F., et al, "Application of an improved particle swarm optimization algorithm for neural network training," *International Conference in Neural Networks and Brain, ICNN&B*, pp 1693 – 1698, 2005.

Hatem Abdul-kader obtained his B.S. and M.SC. (by research) both in Electrical Engineering from the Alexandria University , Faculty of Engineering , Egypt in 1990 and 1995 respectively. He obtained his Ph.D. degree in Electrical Engineering also from Alexandria University, Faculty of Engineering, Egypt in 2001 specializing in neural networks and applications. He is currently a Associate professor in Information systems department, Faculty of Computers and Information, Information systems department, Faculty of Computers and Information, Menoufiya University, Egypt since 2004. He has worked on a number of research topics and consulted for a number of organizations. He has contributed more than 30+ technical papers in the areas of neural networks, Database applications, Information security and Internet applications.

Mustafa Abdul Salam was born on November 01, 1981 in Anshas, Sharkia, Egypt. He received the B.S from Faculty of Computers & Informatics, Zagazig University, Egypt in 2003 with grade very good, and obtains master degree in information system from faculty of computers and information, menufia university, Egypt in 2009. He is working in Higher Technological Institute, 10th of Ramadan city as teaching assistance at Faculty of Computer and informatics. Mustafa has contributed more than 5+ technical papers in the areas of Artificial Neural Network (ANN), Particle Swarm Optimization (PSO), and Differential Evaluation (DE) in international conferences.