# Evaluation of Differential Evolution and Particle Swarm Optimization Algorithms at Training of Neural Network for prediction

Abdul Salam, M.E

Teaching Assistant at Higher
Technological Institute (H.T.I)
10th of Ramadan City, Egypt
mustafa.abdo@ymail.com

H.M.Abdulkader

Faculty of Computer and
Information, Menoufiya University,
Egypt
hatem6803@yahoo.com

W.F. Abdul Wahed,

Head of Operations Research and
Decision Support Systems, Faculty of
Computers and Information, Menoufiya
University, Egypt

*Abstract: This paper presents the comparison of two metaheuristic 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 were made 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, stock prediction.*

## 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 behaviour 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 [1]. The role of artificial neural networks in the present world applications is gradually increasing and faster algorithms are being developed for training neural networks [2]. 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 [3]. 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 [4]. A survey and overview of evolutionary algorithms in evolving artificial neural networks can be found in [5].

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 [16], automated mirror design [3], optimization of radial active magnetic bearings [18], and mechanical engineering design [10]. A differential evolution based neural network training algorithm, introduced in [5,10,23]. 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 [6]. 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 [9]. Neural networks are used in combination with PSO in many applications, like neural network control for nonlinear processes in [10], feedforward neural network training in [11] – [17].  PSO algorithm is used in prediction and forecasting in many applications, like prediction of chaotic systems in [23], electric load forecasting in [25],[26], time series prediction in [28]-[30] and stock market decision making in [31]-[33].


The main Purpose of this paper is to propose of using two modern Artificial intelligence technique 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.

## 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. [19]. 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 [20]. The DE algorithm is shown in figure 1.

### 2.1 Population Structure

The current population, symbolized by $P_{\mathbf{x}}$, is composed of those vectors, $x_{i,g}$, that have already been found to be acceptable either as initial points, or by comparison with other vectors:

$$Px,g = (xi,g) , i=0,1,....,Np-1, g=0,1,...,gmax, xi,g=(xj,i,g), j=0,1,...,D-1 \tag{1}$$

Once initialized, DE mutates randomly chosen vectors to produce an intermediary population, $P_{v,g}$, of $N_p$ mutant vectors, $\mathbf{V}_{i,g}$:

$$P_{v,g}=(V_{i,g}), i=0,1,...,N_p-1, g=0,1,...,gmax, \tag{2}$$
$$V_{i,g}=(V_{j,i,g}), j=0,1...D-1$$

Each vector in the current population is then recombined with a mutant to produce a trial population, $P_{\mathbf{u}}$, of $N_{\mathbf{p}}$ trial vectors, $\mathbf{u}_{i,g}$:

$$P_{u,g} = (u_{i,g}), i=0,1,...,N_p-1, g=0,1,...,gmax, \tag{3}$$
$$u_{i,g} = (u_{j,i,g}), j=0,1,...,D-1$$

During recombination, trial vectors overwrite the mutant population, so a single array can hold both populations.
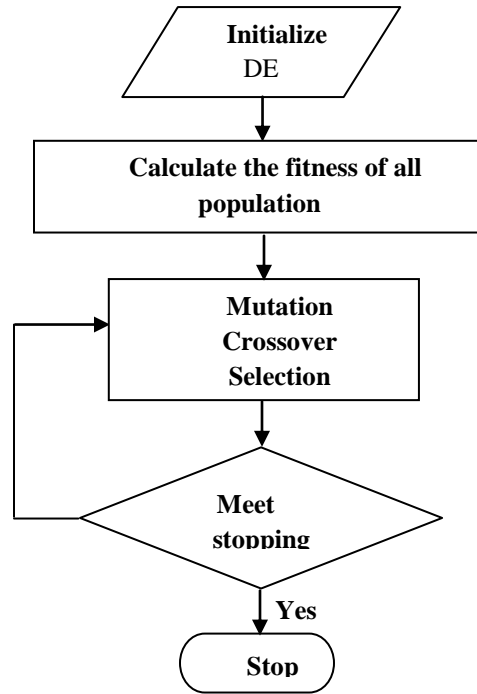
**Figure 1 the DE algorithm**

## 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, $b_L$ and $b_U$. 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 $j$th parameter of the $i$th vector is

$$x_{j,i,0}=rand_j(0,1).(b_{j,U}-b_{j,L})+b_{j,L}. \tag{4}$$

## 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.

$$V_{i,g}=x_{r0,g}+F.(\ x_{r1,g}-\ x_{r2,g}) \tag{5}$$

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:

$$u_{i,g} = (u_{j,i,g}) = \begin{cases} v_{i,i,g} & \text{if } rand_j (0,1) \leq Cr \text{ or} \\ & j = j_{rand} \\ x_{i,i,g} & \text{otherwise} \end{cases} \tag{6}$$

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, $u_{i,g}$, has an equal or lower objective function value than that of its target vector, $x_{i,g}$, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for at least one more generation

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

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. [18]

One possibility could be a hybrid of traditional optimization methods and evolutionary algorithms as studied in [2, 4, 11].

## 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

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

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

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

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.
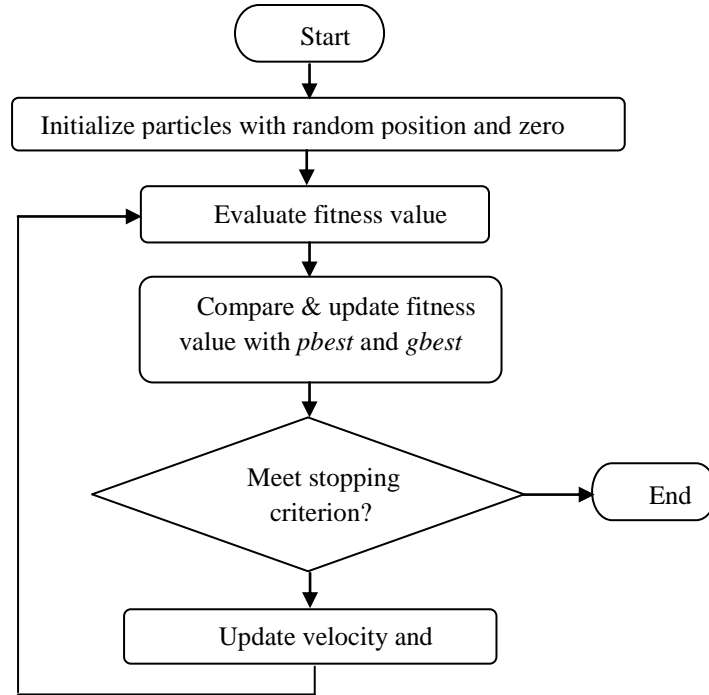
**Figure 2 PSO algorithm**

## 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.

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.

$$RSI = 100- [100 / (1+RS)] \tag{10}$$

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.

$$\text{Money Flow (MF)} = \text{Typical Price} * \text{Volume.} \tag{11}$$
$$\text{Money Ratio (MR)} = (\text{Positive MF} / \text{Negative MF}). \tag{12}$$
$$MFI = 100 - (100/ (1+MR)). \tag{13}$$

*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.

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

*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.

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

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

$$MACD = [0.075*\text{EMA of Closing prices}] - [0.15*\text{EMA of closing prices}] \tag{16}$$
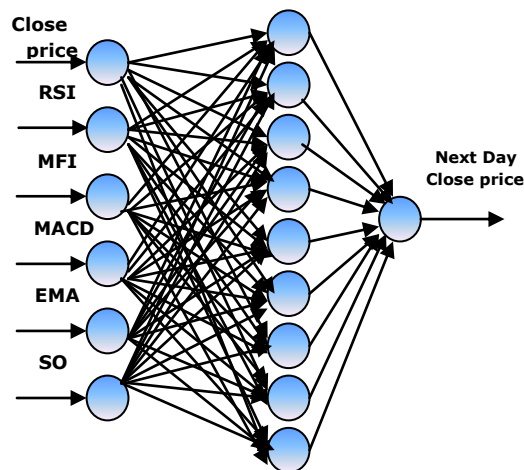$$\text{Signal Line} = 0.2*\text{EMA of MACD} \tag{17}$$



**Figure3. Architecture of neural network of proposed model.**

## 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.

Figure 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 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. outlines Mean squared error with regularization performance function (MSEREG) which measures network**

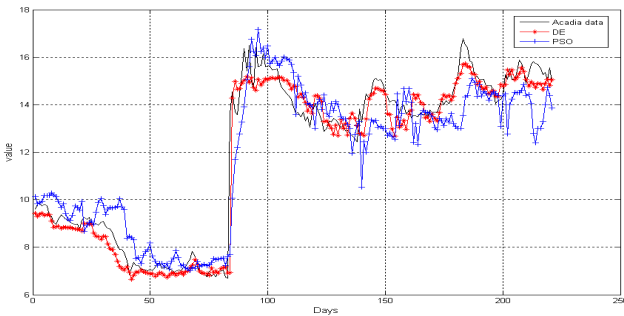| Error<br>Company | MSERG | | MSE | | MAE | |
|---|---|---|---|---|---|---|
| | DE | PSO | DE | PSO | DE | PSO |
| ACADIA | ٠.٥٧ | 1.50 | ٠.٤٩ | 1.46 | ٠.٤٦ | ٠.٩٠ |
| SHLO | ١.٠٤ | ١.٤٣ | 1.01 | 1.39 | ٠.٨٠ | ٠.٩٥ |
| FTWR | ٠.١٧ | ٠.٤٨ | ٠.١٦ | ٠.٣٥ | ٠.٣٠ | ٠.٤٩ |
| HYZ | ٠.٠٦ | ٠.١٩ | ٠.٠٥ | 0.11 | ٠.١٧ | ٠.٢٥ |
| NET | ٠.٧١ | ٢.٢٧ | ٠.٦٥ | ١.٩٤ | ٠.٦٢ | ١.١٤ |
| CALPER | ٠.٠٥ | ٠.١٦ | ٠.٠٢ | ٠.٠٩ | ٠.١٢ | ٠.٢٥ |
| FORD | ٠.٣٤ | ٣.١٠ | ٠.٢٢ | ٣.٠٩ | ٠.٣٩ | ١.٣٠ |
| **Table (1) Error functions for the two algorithms.** | | | | | | |

**Figure 4 Results for Acadia Pharmaceuticals Company.**
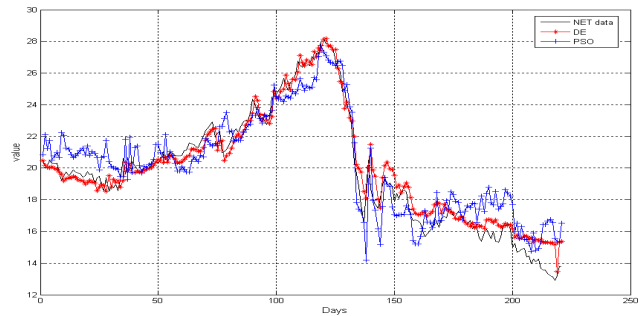


**Figure 8 Results for Strategic Internet (SIII.OB) Company.**
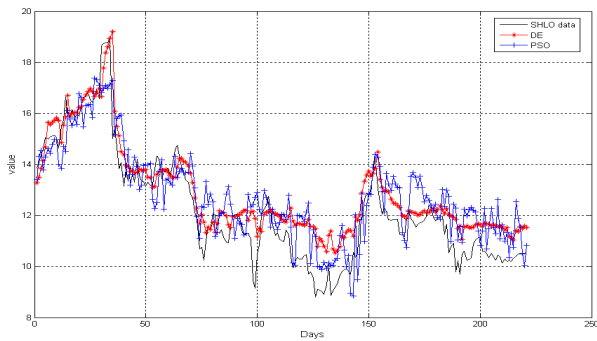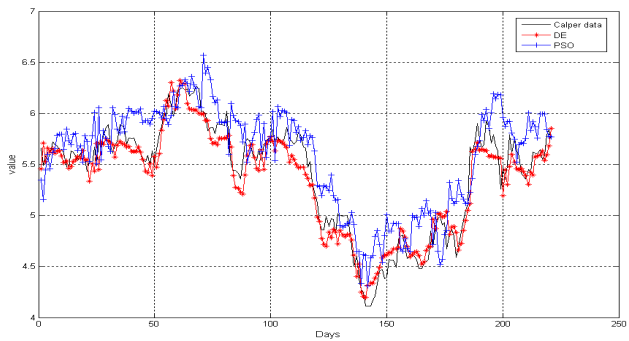


**Figure 5 Results for Shiloh Industries Company.**



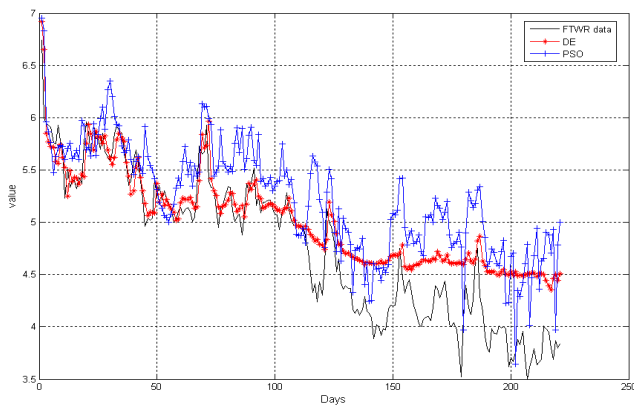**Figure 9 Results for Caliper Life Sciences Company.**



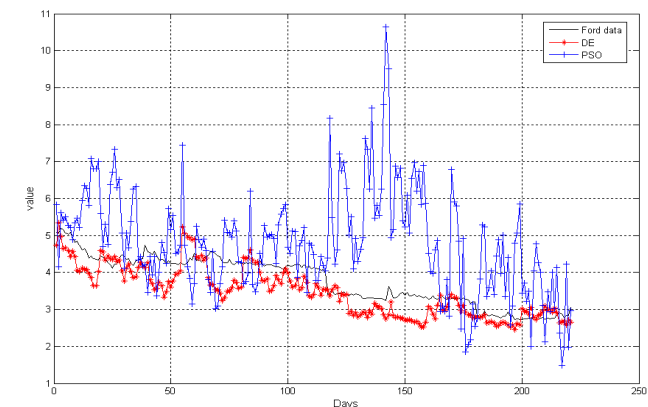**Figure 6 Results for Fiber Tower Company.**



**Figure 10 Results for Ford Company.**

# 6. CONCLUSIONS

In this paper, Differential Evolution (DE) and Particle Swarm Optimization (PSO) algorithms are applied in the training and testing of feed-forward neural network for stock price prediction. The simulation results show the potential of both two algorithms. Both algorithms Convergence

to a global minimum can be expected. Both algorithms can avoid local minima problem which all gradient descending methods fall on it. Easy tuning of both algorithms parameters. Optimum found by both algorithms is never worse than the initial optimum found by a gradient based method. But DE converges to global minimum faster than PSO algorithm. DE algorithm gives better accuracy than PSO algorithm especially in fluctuated time series.

## 6. References

[1]     Olivier coupelon, Blaise Pascal University: Neural network modeling for stock movement prediction, state of art. 2007

[2]     Howard Demuth Mark Beale Martin Hagan: Neural Network Toolbox for Use with MATLAB® 2006.

[3]     Tejen Su, Jyunwei Jhang and Chengchih Hou: A hybrid artificial neural networks and particle swarm optimization for function approximation; ICIC International 2008 ISSN 1349-4198.

[4]     B. Al-kazemi and C.K. Mohan. Training feedforward neural networks using multiphase particle swarm optimization. In Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on, pages 2615 – 2619 vol.5, 2002.

[5]     Yao,X.: Evolving artificial neural networks, Proceedings of the IEEE, 87(9) (1999),1423–1447.

[6]     D. N. Wilke. Analysis of the particle swarm optimization algorithm, Master's Dissertation, University of Pretoria, 2005.

[7]     Jovita Nenortaite and Rimvydas Simutis: Application of Particle Swarm Optimization Algorithm to Stocks' Trading System.2004.

[8]     Khalil A.S.: An Investigation into Optimization Strategies of Genetic Algorithms and Swarm Intelligence.. Artificial Life (2001).

[9]     Kennedy J., Spears W.M.: Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem
Generator.Processes.http://www.aic.nrl.navy.mil/%7Espears/papers/wcci98.pdf. Current as of December 15th, 2003.

[10]    Ying Song, Zengqiang Chen, and Zhuzhi Yuan. New chaotic pso-based neural network predictive control for nonlinear process. IEEE Transactions on Neural Networks, 18:595 –601, 2007.

[11]    M. Carvalho and T.B. Ludermir. Particle swarm optimization of feed-forward neural networks with weight decay. In Hybrid Intelligent Systems, 2006. HIS '06. Sixth International Conference on, pages 5 – 5, 2006.

[12]    Zhang Chunkai, Li Yu, and Shao Huihe. A new evolved artificial neural network and its application. In Intelligent Control and Automation, 2000. Proceedings of the 3rd World Congress on, pages 1065 – 1068 vol.2, 2000.

[13]   Hong-Bo Liu, Yi-Yuan Tang, Jun Meng, and Ye Ji. Neural networks learning using vbest model particle swarm optimisation. In Proceedings of 2004 International Conference on Machine Learning and Cybernetics, pages 3157 – 3159 vol.5, 2004.

[14]   R. Mendes, P. Cortez, M. Rocha, and J. Neves. Particle swarms for feedforward neural network training. In Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on, pages 1895 – 1899, 2002.

[15]   Cui-Ru Wang, Chun-Lei Zhou, and Jian-Wei Ma. An improved artificial fish-swarm algorithm and its application in feed-forward neural networks. In Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, pages 2890 – 2894 Vol.5, 2005.

[16]   Chunkai Zhang, Huihe Shao, and Yu Li. Particle swarm optimisation for evolving artificial neural network. In 2000 IEEE International Conference on Systems, Man, and Cybernetics, pages 2487 – 2490 vol.4, 2000.

[17]   Fuqing Zhao, Zongyi Ren, Dongmei Yu, and Yahong Yang. Application of an improved particle swarm optimization algorithm for neural network training. In Neural Networks and Brain, 2005. ICNN&B '05. International Conference on, pages 1693 – 1698, 2005.

[18]   Chia-Feng Juang. 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.

[19]   Chia-Feng Juang and Yuan-Chang Liou. On the hybrid of genetic algorithm and particle swarm optimization for evolving recurrent neural network. In Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, pages 2285 – 2289 vol.3, 2004.

[20]   H.A. Firpi and E.D. Goodman. Designing templates for cellular neural networks using particle swarm optimization. In Applied Imagery Pattern Recognition Workshop, 2004. Proceedings. 33rd, pages 119 – 123, 2004.

[21]   Yuehui Chen, Jiwen Dong, Bo Yang, and Yong Zhang. A local linear wavelet neural network. In Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on, pages 1954 – 1957 Vol.3, 2004.

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

[23]   F.A. Guerra and L.D.S. Coelho. Radial basis neural network learning based on particle swarm optimization to multistep prediction of chaotic lorenz's system. In Hybrid Intelligent Systems, 2005. Fifth International Conference on, page 3 pp., 2005.

[24]   Changhui Deng, XinJiang Wei, and LianXi Guo. Application of neural network based on pso algorithm in prediction model for dissolved oxygen in fishpond. In

Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, pages 9401 – 9405, 2006.

[25]    Changyin Sun and Dengcai Gong. Support vector machines with pso algorithm for short-term load forecasting. In 2006. ICNSC '06. Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, pages 676– 680, 2006.

[26]    Wei Sun, Ying-Xia Zhang, and Fang-Tao Li. The neural network model based on pso for short-term load forecasting. In Machine Learning and Cybernetics, 2006 International Conference on, pages 3069 – 3072, 2006.

[27]    Jinchun Peng, Yaobin Chen, and R. Eberhart. Battery pack state of charge estimator design using computational intelligence approaches. In Battery Conference on Applications and Advances, 2000. The Fifteenth Annual, pages 173 – 177, 2000.

[28]    X. Cai, N. Zhang, G.K. Venayagamoorthy, and II Wunsch D.C. Time series prediction with recurrent neural networks using a hybrid pso-ea algorithm. In Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, pages 1647 – 1652 vol.2, 2004.

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

[30]    Chunkai Zhang and Hong Hu. Using pso algorithm to evolve an optimum input subset for a svm in time series forecasting. In 2005 IEEE International Conference on Systems, Man and Cybernetics, pages 3793 – 3796 Vol. 4, 2005.

[31]    Jovita Nenortaitė. A particle swarm optimization approach in the construction of decision-making model. ISSN 1392 -124x information technology and control, 2007, vol.36, no.1a

[32]    N.G. Pavlidis, D. Tasoulis, M.N. Vrahatis. Financial Forecasting Through Unsupervised Clustering and Evolutionary Trained Neural Networks. 2003 Congress on Evolutionary Computation, Canberra Aust-ralia, 2003.

[33]    J. Nenortaite, R. Simutis. Stocks' Trading System Based on the Particle Swarm Optimization Algorithm. Lecture Notes on Computer Science, 2004, 3039, 843-850.