Comparative Study between Differential Evolution and Particle Swarm Optimization Algorithms in Training of Feed-Forward Neural Network for Stock Price prediction.

Mustafa E. Abdual-Salam[1], Hatem M. Abdul-Kader[2], and Waiel F. Abdel-Wahed[3]

(Corresponding author: Mustafa E. Abdul-Salam)

Higher Technological Institute 10th of Ramadan City[1]

Mustafa.abdo@ymail.com [1]

waiel_fathi@yahoo.com [3]

Faculty of Computers and Information, Minufiya University[2]

Hatem6803@yahoo.com[2]

***Abstract*—This paper presents a comparison between two stochastic, population based and real-valued algorithms. These algorithms are namely Differential Evolution (DE) and Particle Swarm Optimization (PSO). These algorithms are used in the training of feed-forward neural network to be used in the prediction of the daily stock market 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 algorithms are demonstrated. These algorithms are proposed to solve the problems of traditional training techniques like local minima and overfitting. Comparisons were made between the two approaches in terms of the prediction accuracy, convergence speed and generalization ability. 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. The simulation results presented in this paper show the potential of both two algorithms in solving the problems of traditional training techniques. DE algorithm is better than PSO algorithm in prediction accuracy, convergence speed and handling fluctuated stock time series.**

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

I. 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].

Particle Swarm Optimization **(**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].

This 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.

II. DIFFERENTIAL EVOLUTION 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.

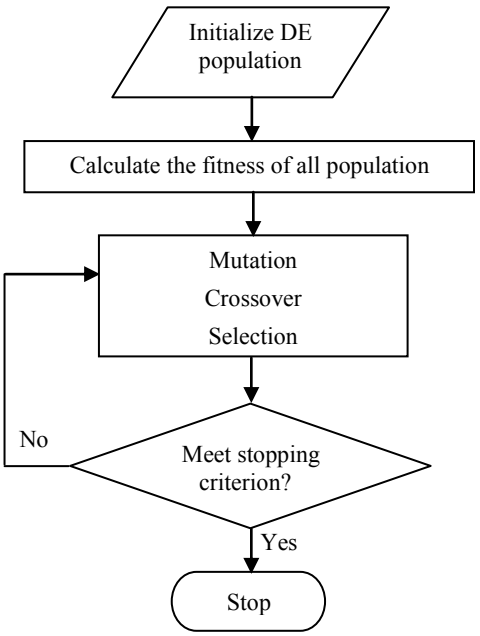

Fig. 1 The DE algorithm

A. 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:

$$P_{\boldsymbol{x,g}} = (x_{i,g}) \ , \ i=0,1,....,N_{p-1}, \ g=0,1,...,g_{max} \ , \qquad \textbf{(1)}$$
$$x_{i,g}=(x_{j,i,g}), \ j=0,1,...,D-1$$

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

$$P_{v,g}=(V_{i,g}), \ i=0,1,...,N_p-1, \ g=0,1,...,g_{max}, \qquad \textbf{(2)}$$
$$V_{i,g}=(V_{j,i,g}), \ j=0,1...D-1$$

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, $\mathbf{u}_{i,g}$:

$$P_{u,g} = (u_{i,g}), \ i=0,1,...,N_p-1, \ g=0,1,...,g_{max}, \qquad \textbf{(3)}$$
$$u_{i,g} = (u_{j,i,g}), \ j=0,1,...,D-1$$

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

B. 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 jth parameter of the ith vector is

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

C. 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.

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

The scale factor, $F \in (0,1+)$, 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.

D. 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:

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

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

E. 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

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

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].

III. 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

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

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

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

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 figure2.

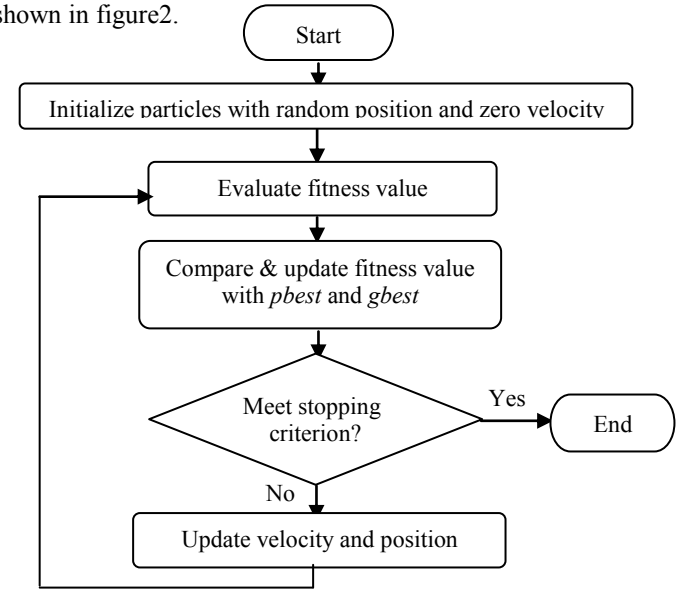

Fig. 2 The PSO algorithm.

IV. THE PROPOSED MODEL

The proposed methodology is to train multilayer feedforward neural network with DE algorithm and also PSO 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.

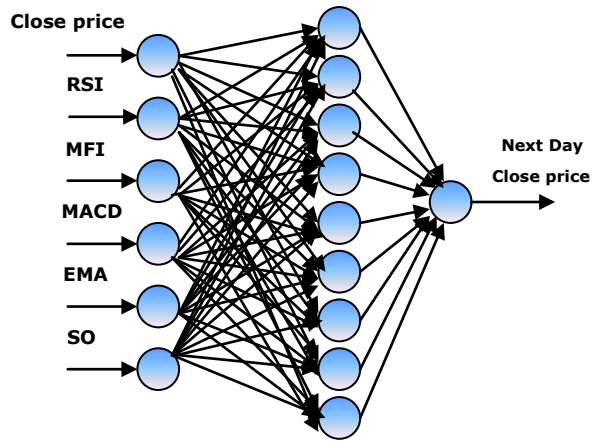

Fig. 3 Architecture of ANN of proposed model.

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.

A. USED TECHNICAL INDICATORS IN THIS MODEL.

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

- *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)] **(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.

  *Money Flow (MF) = Typical Price * Volume.* **(11)**

  Money Ratio (MR) = (Positive MF / Negative MF). **(12)**

  MFI = 100 – (100/ (1+MR)). **(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 = [α *Today's Close] + [1-α* Yesterday's EMA].* **(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 = [(Close price – Lowest price) / (Highest Price – Lowest Price)] * 100* **(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*EMA of Closing prices] – [0.15*EMA of closing prices]* **(16)**

  *Signal Line = 0.2*EMA of MACD* **(17)**

B. TRAINING NEURAL NETWORK USING DE ALGORITHM.

In standard training processes, both the input vector **x** and the output vector **y** are known, and the synaptic weights in *W* are adapted to obtain appropriate functional mappings from the input **x** to the output **y**. Generally, the adaptation can be carried out by minimizing the network error function **E** which is of the form

$$E(y, f(x,W)) : (y^{D1}, x^{D2}, W^{D3}, f) \to R. \quad \textbf{(18)}$$

The optimization goal is to minimize the objective function $E(y, f(x,W))$ by optimizing the values of the network weights. The DE training steps are as follow.

Step1. Initialize population:

$$P_G = (w_{1,G},..., w_{NP,G})^T, \quad G = 1,..., G_{\max} \quad \textbf{(19)}$$

$$w_{i,G} = (w_{1,i,G},..., w_{D,i,G}), \quad i = 1,..., NP \quad \textbf{(20)}$$

Where D is the number of weights in the weight vector and in $w_{i,G}$, i is index to the population and G is the generation to which the population belongs.

Step2. Evaluate all the candidate solutions inside population for a specified number of iterations.

Step3. For each candidate in the population, selects the random variables $r1, r2, r3 \in \{1,2,..., NP\}$.

Step4. Apply mutation operator to yield a mutant vector according to

$$v_{j,i,G+1} = w_{j,r3,G} + F \cdot (w_{j,r1,G} - w_{j,r2,G}) \quad \textbf{(21)}$$

Step5. Apply crossover to produce trial vector according to

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{If } rand_j[0,1) \le CR \\ w_{j,i,G} & \text{Otherwise,} \end{cases} \quad \textbf{(22)}$$

Step6. Apply selection between the trial vector and target vector according to

$$W_{i,G+1} = \begin{cases} U_{i,G+1} \text{ If} & E(y, f(x, W_{i,G+1}) \le E(y, f(x, W_{i,G})) \\ W_{i,G}, & \text{Otherwise} \end{cases} \quad \textbf{(23)}$$

Each individual of the temporary population is compared to its counterpart in the current population. Assuming that the objective function is to be minimized, the vector with the lower objective function value wins a place in the next generation's population. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. The interesting point concerning DE's replacement scheme is that a trial vector is only compared to one individual, not to all individuals in the current population. It is also noteworthy that the replacement scheme ensures that the population does not diverge from or lose the best solution found so far [35].

C. TRAINING NEURAL NETWORK USING PSO ALGORITHM.

PSO does not just train one network, but rather trains a network of networks. PSO builds a set number of ANN and initializes all network weights at random values and starts training each one. On each pass through a data set, PSO compares each network's fitness. The network with the highest fitness is considered the global best. The other networks are updated based on the global best network rather than on their personal error or fitness.

Each neuron contains a position and velocity. The position corresponds to the weight of a neuron. The velocity is used to update the weight. The velocity is used to control how much the position is updated. If a neuron is further away (the position is further from the global best position) then it will adjust its weight more than a neuron that is closer to the global best. The training process is listed below.

Step 1: Initialization

The velocity and position of all particles are randomly set to within pre-specified or legal range.

Step 2: Velocity Updating

For each iteration, the velocity of all particles is updated according to equation (8).

Step 3: Position Updating

Assuming unit time interval between successive iterations, the positions of all particles are updated according to equation (9).

Step 4 Memory Updating

The personal best position, $\mathbf{y}_i$, associated with particle i is updated at the next time step, $t + 1$, according to

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \ge f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases} \quad \textbf{(24)}$$

As with EAs, the fitness function $f(y_i(t))$ measures how close the corresponding solution is to the optimum, i.e. the fitness function quantifies the performance, or quality, of a particle (or solution).

The Global best is calculated according to

$$y(t) \in \left\{ y_0(t), \ldots, y_{ns}(t) \middle| f(\hat{y}(t)) = \min\{f(y_0(t), \ldots, f(y_{ns}(t))\} \right\} \quad \textbf{(25)}$$

The value of interia weight (w) is extremely important to ensure convergent behavior, and to optimally tradeoff exploration and exploitation. The inertia weight is adjusted according to

$$w_i(t+1) = w(0) + (w(n_t) - w(0)) \frac{e^{mi(t)} - 1}{e^{mi(t)} + 1} \quad \textbf{(26)}$$

Where the relative improvement, mi, is estimated as

$$m_i(t) = \frac{f(y_i(t)) - f(x_i(t))}{f(y_i(t)) + f(x_i(t))} \quad \textbf{(27)}$$

With $w(n_t) \approx 0.5$ and $w(0) < 1$

Step 5: Termination Checking

Repeats Step 2 to Step 4 until certain termination condition is met, such as pre-defined number of iterations is reached or failed to have progress for certain number of iterations. Once a particle has achieved the required fitness, a solution has been found.

V. RESULTS AND DISCUSSION

Neural networks are trained and tested with datasets form September 2004 to September 2007. All datasets are available on [34]. Datasets are divided into training part (70%) and testing part (30%).

Figures (4-9) 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) 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, 9) outline different time series with changing trends. The DE can easily cope up with the fluctuation existing in the time series better than PSO algorithm.

Table (I) outlines mean squared error with regularization performance function (MSEREG) which measures network 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.

Figure 10 displays the MSEREG function. Anyone can notice from the figure the advances of the DE algorithm over PSO algorithm in error value and the ability of coping with fluctuations happened in the time series.

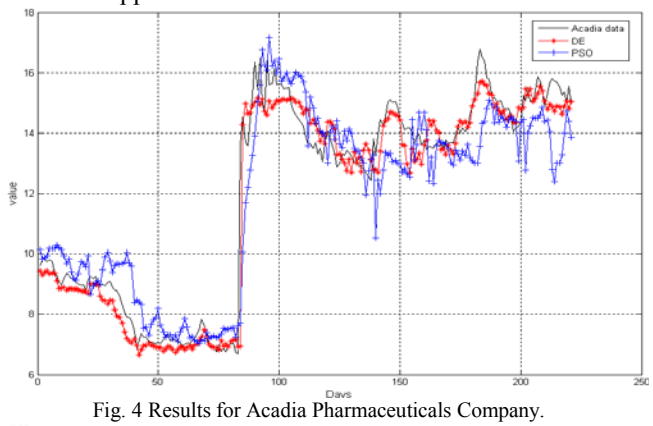

Fig. 4 Results for Acadia Pharmaceuticals Company.

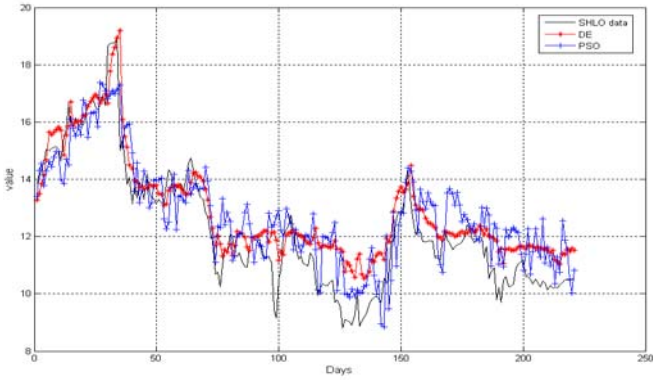

Fig. 5 Results for Shiloh Industries Company.

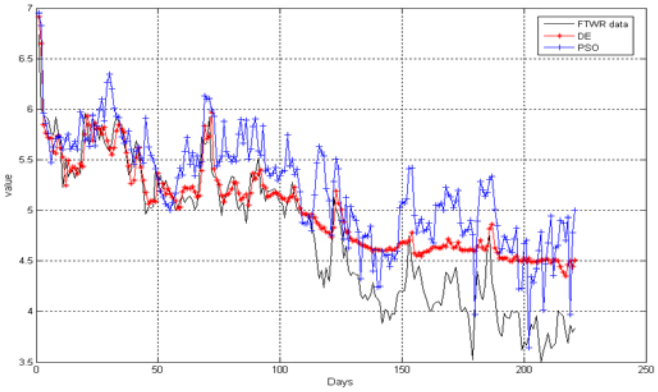

Fig. 6 Results for Fiber Tower Company.

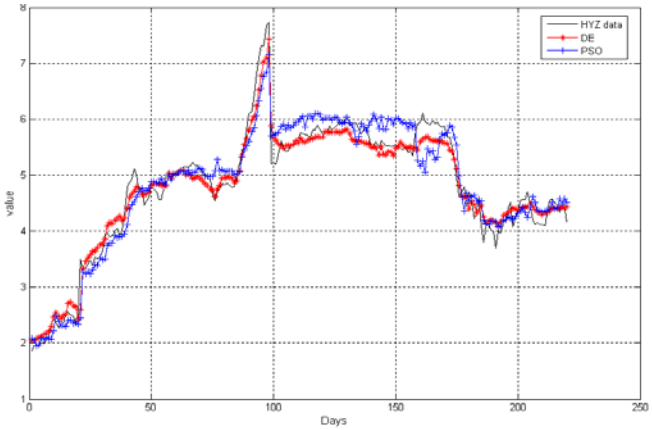

Fig. 7 Results for Hayz Lemmerz Company.

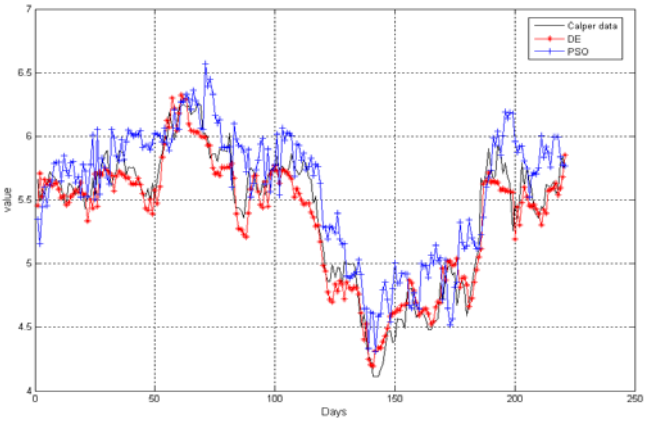

Fig. 8 Results for Caliper Life Sciences Company.

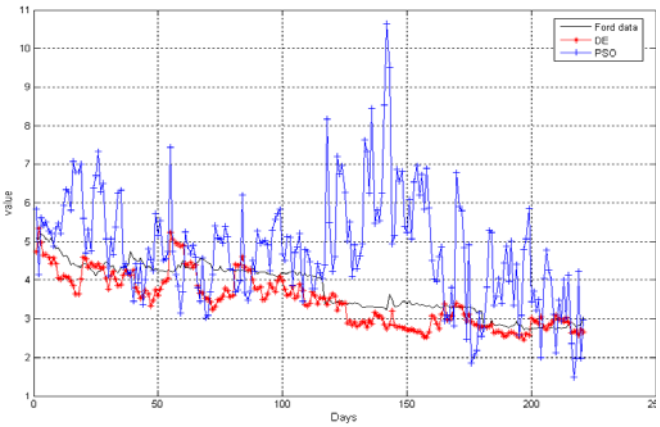

Fig. 9 Results for Ford Company.

TABLE I

ERROR FUNCTIONS FOR THE DE AND PSO 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

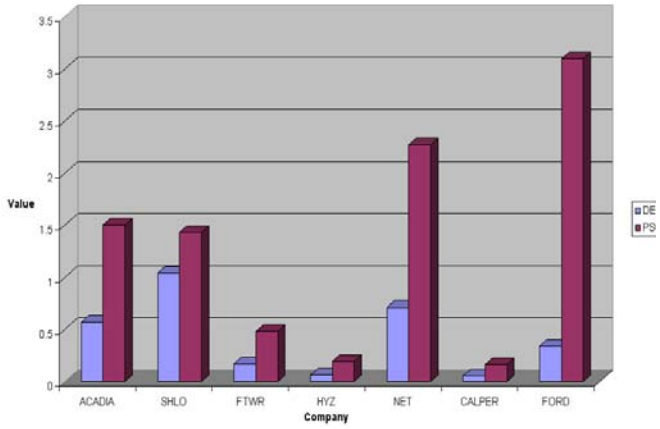

Fig. 10 MSEREG function of DE and PSO.

VI. 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 to be used in the prediction of daily stock market prices. The simulation results show the potential of both two algorithms in overcoming the traditional training algorithms problems. 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. DE converges to global minimum faster than PSO algorithm. DE algorithm gives better accuracy than PSO algorithm.DE algorithm is better than PSO algorithm in the prediction of fluctuated time series.

REFERENCES

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

[2] Howard, D., Mark, B., and Martin, H. *"Neural Network Toolbox™ 6 User's Guide for MATLAB®",* Mathworks, *2009.*

[3] Tejen, S., Jyunwei, J., and Chengchih, H."*A hybrid artificial neural networks and particle swarm optimization for function approximation*", International Journal of Innovative Computing, Information and Control (ICIC 2008), Vol. 4, No. 9, September 2008.

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

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

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

[7] Jovita, N., and Rimvydas, S."*Application of Particle Swarm Optimization Algorithm to Stocks' Trading System", Springer, 2004.*

[8] Khalil A.S." *An Investigation into Optimization Strategies of Genetic Algorithms and Swarm Intelligence."* portal of Swarm Intelligence at National Chin-Yi Institute of Technology in Taiwan, 2001.

[9] Kennedy J., Spears W.M. "*An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the MultimodalProblem"*.,http://www.aic.nrl.navy.mil/%7Espears/papers/wci98.pdf. Current as of December 15th, 2003.

[10] Ying, S., Zengqiang, C., and Zhuzhi, Y. "*New chaotic pso-based neural network predictive control for nonlinear process",* IEEE Transactions on Neural Networks, Vol.18, pp.595 –601, 2007.

[11] Carvalho, M. and Ludermir, T.B. *"Particle swarm optimization of feed-forward neural networks with weight decay"*, Proceedings of the 6th International Conference on Hybrid Intelligent Systems "HIS '06", pp.5–9 Vol.1, Auckland, New Zealand, 13-15 December 2006.

[12] Zhang, C., Li, Y., and Shao, H. *"A new evolved artificial neural network and its application"*, Proceedings of the 3rd World Congress on Intelligent Control and Automation, pp. 1065–1068 Vol.2, Peoples R China, 26 June- 2 July 2000.

[13] Hong, B.L., Yi, T., Jun, M., and Ye, J. *"Neural networks learning using vbest model particle swarm optimization"*, Proceedings of International Conference on Machine Learning and Cybernetics, pp. 3157 – 3159 Vol.5, Shanghai, China, 6-29 August 2004.

[14] Mendes, R., Cortez, P., Rocha, M., and Neves, J. *"Particle swarms for feedforward neural network training"*, Proceedings of the International Joint Conference on Neural Networks "IJCNN '02", pp. 1895 – 1899 Vol.1, Hawaii, USA , 12-17 May 2002.

[15] Wang, C.R., Zhou, C.L., and Ma, J.W. *"An improved artificial fish-swarm algorithm and its application in feed-forward neural networks",* Proceedings of the 4th International Conference on Machine Learning and Cybernetics, pp. 2890 – 2894 Vol.5, Guangzhou, China, 18 - 21 August 2005.

[16] Chunkai, Z., Huihe, S., and Yu, L." *Particle swarm optimisation for evolving artificial neural network".* IEEE International Conference on Systems, Man, and Cybernetics, pp. 2487 – 2490, Vol.4, 2000.

[17] Fuqing, Z., Zongyi, R., Dongmei, Y., and Yahong, Y. *"Application of an improved particle swarm optimization algorithm for neural network training"*, Proceedings of International Conference on Neural Networks and Brain "ICNN&B '05", pp. 1693 – 1698, Beijing, China, 13-15 October 2005.

[18] Chia, F.J." *A hybrid of genetic algorithm and particle swarm optimization for recurrent network design"*, IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol.34,pp.997– 1006, 2004.

[19] Chia,F. J., and Yuan,C.L." *On the hybrid of genetic algorithm and particle swarm optimization for evolving recurrent neural network"*, In Proceedings of IEEE International Joint Conference on Neural Networks, pp. 2285 – 2289 Vol.3, 2004.

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

[21] Yuehui, C., Jiwen D., Bo, Y., and Yong, Z. *"A local linear wavelet neural network. In Intelligent Control and Automation"*, WCICA 2004. Fifth World Congress on, pp. 1954 – 1957 Vol.3, 2004.

[22] Reynolds, P.D. , Duren, R.W., Trumbo, M.L., and Fpga., R.J. "*Implementation of particle swarm optimization for inversion of large neural networks"*, In Proceedings 2005 IEEE Swarm Intelligence Symposium, SIS 2005, pp. 389 – 392,Vol.1, 2005.

[23] Guerra, F.A., and Coelho, L.D.S., *"Radial basis neural network learning based on particle swarm optimization to multi step prediction of chaotic Lorenz's system"*, Proceedings of 5th International Conference on Hybrid Intelligent Systems, pp. 512-523 Vol.1, Rio de Janeiro, Brazil, 06-09 November 2005.

[24] Changhui, D., XinJiang, W., and LianXi, G."*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, pp.9401 – 9405, Vol.1, 2006.

[25] Changyin, S. and Dengcai, G., *"Support vector machines with pso algorithm for short-term load forecasting",* Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control" ICNSC '06", pp. 676– 680, Florida, USA, 23-25 April 2006.

[26] Wei, S., Ying, X., and Fang, L. "*The neural network model based on pso for short-term load forecasting"*, In Machine Learning and Cybernetics, 2006 International Conference on, pp. 3069 – 3072, 2006.

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

[28] Cai, X., Zhang, N., Venayagamoorthy, G.K., and 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, pp. 1647 – 1652, Vol.2, 2004.

[29] Yang, Y., Chen, R.S.,Ye, Z.B. and Liu, Z. "*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, pp.12-16, 2005.

[30] Chunkai, Z., and Hong, H., *"Using pso algorithm to evolve an optimum input subset for a svm in time series forecasting"*, Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp.3793 – 3796 Vol. 4, Hawaii, USA ,10-12 October 2005.

[31] Nenortaitė, J., *"A particle swarm optimization approach in the construction of decision-making model"*, International Journal of Information Technology and Control, pp.158 – 163 Vol.36, 2007.

[32] Pavlidis, N.G., Tasoulis, D., Vrahatis, M.N., *"Financial Forecasting Through Unsupervised Clustering and Evolutionary Trained Neural Networks"*, The 2003 Congress on Evolutionary Computation "CEC'03", pp. 2314 – 2321 Vol.4, Canberra, Australia, 8 - 12 December 2003.

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

[34] http://finance.yahoo.com web site.

[35] JARMO, I.,JON I-KRISTIAN, K., and JOUNI, L. *"Differential Evolution Training Algorithm for Feed-Forward Neural Networks"*, Neural Processing Letters, Kluwer Academic Publishers Hingham, MA, USA, pp. 93–105, Vol.17, No.1, 2003.