International Journal of Mathematical Archive-7(7), 2016, 140-150
Available online through www.ijma.info ISSN 2229 – 5046

USING BEST REPLACEMENT OPTIMIZATION (BRO) TECHNIQUES TO PREDICTING NON-LINEAR DYNAMIC SYSTEM: EVIDENCE FROM S&P CNX NIFTY FIFTY STOCK INDEX

Dr. T. CHITRA KALARANI[1], S. INDRAKALA[2]*

**[1,2*]Associate professor, Department of Mathematics,
Kunthavai Naachiyaar Government Arts College for Women (Autonomous),
Thanjavur – 613 007. India.**

(Received On: 16-07-16; Revised & Accepted On: 31-07-16)

ABSTRACT

Stock market analysis is one of the most important and hard problems in finance analysis field. Recently, the usage of intelligent systems for stock market prediction has been widely established. In this study, we aim to design a mathematical model for Stock price prediction which can provide an accurate direction for financial firms and private investors. With knowledge of the movement of stock price, an investor can make profitable decision and reduce risk return trade-off. In this paper, the Best Replacement Optimization algorithm is proposed, which is used for the S&P CNX NIFTY stock index analysis. The results provide better forecasting accuracy than previous methods.

Key Word: *Fuzzy Time Series, Particle Swarm Optimization, Prediction, Stock Index Price.*

I. INTRODUCTION

Prediction is a kind of estimation about the development mode that how things are going on in the future. As an essential guidance towards social production, prediction has always been a highlight in academic research field. As for Stock price prediction, the widely applied methods have been limited to the traditional ones, such as time serial analysis, regression analysis, and gray prediction model and so on. However, with the scale of data becoming larger and requirement for prediction accuracy becoming higher, limitations of these traditional methods appear to be obvious. Therefore, many researchers have paid attention to the prediction model optimization.

Predicting the trend of the stock market is considered as a tough and challenging task in financial time-series forecasting. The main reason is because of the uncertainties involved in the movement of the stock market. Various factors affect the stock market and some of them include traders' expectations, events related to the government and economic conditions. Hence, predicting the movement of stock market price is quite a difficult job.

In the past years several models and techniques had been developed to stock price prediction. Among them are artificial neural networks (ANNs) model which are very popular due to its ability to learn patterns from data and infer solution from unknown data. Few related works that engaged ANNs model to stock price prediction are [2, 3, 8]. Akhter Mohiuddin Rather [2011], in his work, used prediction based neural networks approach for stock returns. An autoregressive neural network predictor was used to predict future stock returns. Various error metrics have been used to evaluate the performance of the predictor. Experiments with real data from National stock exchange of India (NSE) were employed to examine the accuracy of this method. Data from date 02-01-2007 till 22-03-2010 for: TCS, BHEL, Wipro, Axis Bank, Maruthi and Tata Steel were taken. The result was not accurate but he suggested the use of better neural predictive systems and training methods for minimizing the prediction errors for the future work. D. Ashok kumar *et al.* [2013] discussed some basic ideas of time series data, need of ANN, importance of stock indices, survey of the previous works and it investigates neural network models for time series in forecasting. In their study, for performance of between BSE100 stock market index and NIFTY MIDCAP50 stock market index is studied by neural network model and measured aggregation where observed viz., MAE, MAPE, PMAD, MSE and RMSE. The result shows that the performance is comparatively best. From the result they observed that an optimal feedback weighting factor learning rate is 0.28, momentum is 0.5 and epoch is 2960. The model achieved the lower prediction error and it may be fit into any stock market data. Aditya Nawani *et al.* [2013] discuss how data mining techniques can be applied to design a market capital prediction system for trading firms. Their study shows how neural networks can be utilized,

*Corresponding Author: S. Indrakalakala***
***Assistant professor, Department of Mathematics, Kunthavai Naachiyaar Government Arts College for Women (Autonomous), Thanjavur – 613 007. India.*

in combination with the Graphical user Interface of MATLAB, GUIDE, to make accurate predictions. When implemented, the trained system can be used to forecast the market capital for a particular combination of input parameters. The accuracy of this method was high because the results obtained were found to be comparable to the output expected.

Rest of the paper is organized as follows. In Section 2, we have presented a brief overview for Best Replacement Optimization Model. In Section 3, we have presented prediction based algorithm and experimental results are given in Section 4. Finally, some concluding remarks are given in Section 5

A. The Best Replacement Optimization Model Development

The Best Replacement Optimization Model (abbreviated as BRO) is a novel population-based stochastic Prediction algorithm and an alternative solution to the complex non-linear optimization problem. The PSO algorithm was first introduced by Dr. Kennedy and Dr. Eberhart in 1995 and its basic idea was originally inspired by simulation of the social behavior of animals such as bird flocking, fish schooling and so on. It is based on the natural process of group communication to share individual knowledge when a group of birds or insects search food or migrate and so forth in a searching space, although all birds or insects do not know where the best position is. But from the nature of the social behavior, if any member can find out a desirable path to go, the rest of the members will follow quickly.

The PSO algorithm basically learned from animal's activity or behavior to solve optimization problems. In PSO, each member of the population is called a particle and the population is called a swarm. Starting with a randomly initialized population and moving in randomly chosen directions, each particle goes through the searching space and remembers the best previous positions of itself and its neighbors. Particles of a swarm communicate good positions to each other as well as dynamically adjust their own position and velocity derived from the best position of all particles. The next step begins when all particles have been moved.

Finally, all particles tend to fly towards better and better positions over the searching process until the swarm move to close to an optimum of the fitness function. The PSO method is becoming very popular because of its simplicity of implementation as well as ability to swiftly converge to a good solution. It does not require any gradient information of the function to be optimized and uses only primitive mathematical operators.

Best replacement Optimization Algorithm as a modified version of PSO Algorithm. In addition, there are few parameters to adjust in BRO Algorithm. That's BRO is an ideal optimization problem solver in optimization problems. As compared with other optimization methods, it is faster, cheaper and more efficient. BRO is well suited to solve the non-linear, non-convex, continuous, discrete, integer variable type problems.

The Basic Model of BRO algorithm

This chapter discusses a conceptual overview of the BRO algorithm and its parameters selection strategies and mathematical explanation. Consider the global optimum of an n-dimensional function defined by

$$f(x_1, x_2, x_3, \ldots., x_n) = f(x^*) \tag{1}$$

where $\boldsymbol{x_i}$ is the search variable, which represents the set of free variables of the given function. The aim is to find a value $\boldsymbol{x}^*$such that the function $\boldsymbol{f}(\boldsymbol{x}^*)$is either a maximum or a minimum in the search space.

Derivation of the BRO Equations

The kinematic equation by which a particle's final position vector can be calculated from its initial position and velocity if acceleration is constant over the time period:

$$\vec{x} = \vec{x}_o + \vec{v}_0 t + \frac{1}{2}\vec{a}t^2 \tag{2}$$

t between position updates is 1 iteration; hence, and t the corresponding dimensional analysis can be dropped to simplify iterative computations. Instead, authors generally use k to denote values of the current iteration and $k + 1$ for values of the ensuing iteration. Using this notation, the basic position update equation of physics can be rewritten for iterative computation as

$$\vec{x}(k+1) = \vec{x}(k) + \vec{v}(k) + \frac{1}{2}\vec{a}(k) \tag{3}$$

The *global best*, which the best solution is found by the swarm through the current iteration, k. Modeling cognition, each particle also iteratively accelerates toward its *personal best*, which is the best solution that it personally has found.

The cognitive and social acceleration constants, $c_1$and c_2 respectively, determine how aggressively particles accelerate based on the cognitive and social information available to them: these can be set identically, or a preference can be given to either acceleration type. The subtraction in Equation (3) ensures that any particle which is distant from the social best will accelerate toward it more strongly than a particle nearby. Similarly, the subtraction in Equation (3) ensures that any particle that is distant from its cognitive best will accelerate toward it more strongly than were it nearby. As a conceptual example, one could imagine children playing both indoors and out when their mother announces that dinner is ready: those farther away might be expected to run more quickly toward the dinner table.

Social acceleration:

$$c_2(\vec{g}(k) - \vec{x}_i(k)) \tag{4}$$

where

c_2 is the social acceleration constant ,
$\vec{x}_i(k)$ is theposition vector of particle"i" at iteration"k",
$\vec{g}(k)$ is the global best of all particles at iteration "k"

Cognitive acceleration:

$$c_1(\vec{g}(k) - \vec{x}_i(k)) \tag{5}$$

where:

c_1 is the cognitive acceleration constant,
$\vec{x}_i(k)$ is the position vector of particle"i" at iteration"k",
$P_i(k)$ is the personal best of particle "i" at iteration "i"

Equation (3) defines the social acceleration of Global Best (Gbest). Local Best (Lbest) limits each particle's social sphere to knowledge of the best solution found by its neighbors instead of immediately granting each particle knowledge of the best solution found so far by the entire search team.

Substituting the sum of these two acceleration terms for $\vec{a}(k)$in Equation (3), while applying the subscript adopted in (4) and (5), produces Equation (6).

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k) + \frac{1}{2}c_1\left(\vec{P}_i(k) - \vec{x}_i(k)\right) + \frac{1}{2}c_2\left(\vec{g}(k) - \vec{x}_i(k)\right) \tag{6}$$

Having replaced physical acceleration in the position update equation of physics with social and cognitive modeling, the next step toward producing a stochastic search algorithm is the replacement of with a pseudo-random number sampled per dimension from the uniform distribution between 0 and 1, $U(0,1)$. Note that the expected or mean value of the distribution is still. Designating the first vector of pseudo-random numbers as $\vec{r}_{1i}$ and the second as $\vec{r}_{2i}$ produces Equation (6).

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k) + c_1\vec{r}_{1i} \circ \left(\vec{P}_i(k) - \vec{x}_i(k)\right) + c_2\vec{r}_{2i} \circ \left(\vec{g}(k) - \vec{x}_i(k)\right) \tag{7}$$

For convenience, the rather long Equation (7) is separated into a velocity update equation (8) and a position update equation (9). This primarily helps with record keeping since each value can be stored separately for post-simulation analysis. Substituting Equation (8) into (9) shows equivalency to (7).

For convenience, the rather long Equation (7) is separated into a velocity update equation (8) and a position update equation (9). This primarily helps with record keeping since each value can be stored separately for post-simulation analysis. Substituting Equation (6.8) into (6.9) shows equivalency to (7).

$$\vec{v}_i(k+1) = \vec{v}_i(k) + c_1\vec{r}_{1,i} \circ \left(\vec{P}_i(k) - \vec{x}_i(k)\right) + c_2\vec{r}_{2,i} \circ \left(\vec{g}(k) - \vec{x}_i(k)\right) \tag{8}$$

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1) \tag{9}$$

Since its conception, Equation (8) has developed two mechanisms by which to improve search behavior. The inertia weight, ω roughly simulates friction in a computationally inexpensive manner by $\omega\epsilon(-1,1)$ carrying over to the next iteration only a user-specified percentage of the current iteration's velocity. This is done by multiplying the velocity of the current iteration by 1 as shown in the first term of Equation (10). The constriction models use a constriction coefficient instead, but the popular Type 1" parameters can be converted to Clerc's Equivalents for use in Equation (10).

$$\vec{v}_i(k+1) = \omega\vec{v}_i(k) + c_1\vec{r}_{1,i} \circ \left(\vec{P}_i(k) - \vec{x}_i(k)\right) + c_2\vec{r}_{2,i} \circ \left(\vec{g}(k) - \vec{x}_i(k)\right) \tag{10}$$

$$\omega = \omega_{max} - \frac{(\omega_{max} - \omega_{min}) \times iter}{Max\ iter}$$

where

$\omega_{max} = initial\ weight, \omega_{min} = final\ weight, iter = current\ itertation\ number,$
$Max\ iter = maximum\ \ itertation\ number$

The equation (10) is enhanced by limiting the iteration using threshold level of global fitness value where the Global Fitness Value ranges from Global Fitness± 1.

$$\vec{v}_i(k+1) = \sum_{i=GF-1}^{i=GF+1}\left(\omega\vec{v}_i(k) + c_1\vec{r}_{1,i} \circ \left(\vec{P}_i(k) - \vec{x}_i(k)\right) + c_2\vec{r}_{2,i} \circ \left(\vec{g}(k) - (k)\right)\right) \quad (11)$$

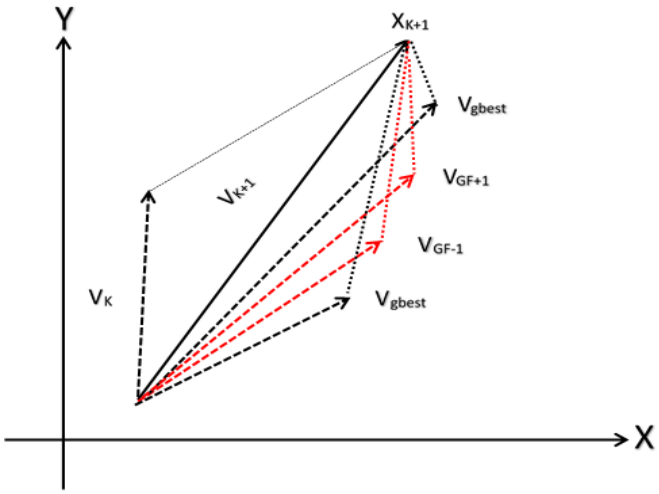

Figure-1: Concepts of BRO

X_k	:	Current searching point
X_{k+1}	:	Modified searching point
V_k	:	Current velocity
V_{k+1}	:	Modified velocity
V_{pbest}	:	Velocity based on pbest
V_{gbest}	:	Velocity based on gbest
GF	:	Global fitness value

The equation (11) improves the efficiency of the iteration and reduces the processing. The other restriction imposed on velocity is essentially a speed limit. Rather than limiting the vector magnitude itself, the computationally simpler approach of limiting each component is implemented as shown in Equation (11), which limits the magnitude indirectly.

$$\vec{v}_{i,j}(k+1) = sign\left(\vec{v}_{i,j}(k+1)\right) \times max\left(\left|\vec{v}_{i,j}(k+1)\right|, v_j^{max}\right) \quad (12)$$

where $j \in \{1,2,\ldots,n-1,n\}$, and n denotes the problem dimensionality

This limits the maximum step size on dimension j by clamping: (i) values greater thanv_j^{max} to amaximum value of v_j^{max} , and (ii) values less than $-v_j^{max}$ to a minimum of $-v_j^{max}$. From a physical perspective, particles with clamped velocities are analogous to birds with limited flying speeds. Considering the psychological aspects of the algorithm, clamped velocities could also be considered analogous to self-limited emotive responses.

Concerning the calculation of v_j^{max} , suppose the feasible candidates for an application problem are [12, 20] on some dimension to be optimized. Clamping velocities to 50%, for example, of x_{max} would allow particles to take excessively large steps relative to the range of the search space on that dimension; in this case, the maximum step size would be 0.5*20 =10 . But stepping 10 units in any direction when the search space is only 8 units wide would be nonsensical. Since real-world applications are not necessarily centered at the origin of Euclidean space, it is preferable to clamp velocities based on the range of the search space per dimension in order to remove dependence on the frame of reference; hence, subscript j is included in Equation (11) for sake of generality; but it can be dropped for applications with the same range of values per dimension.

Swarm Initialization

The optimization process begins by randomly initializing positions between a minimum and maximum per dimension as per Relation (12). The most common benchmarks use the same minimum and maximum per dimension. For application problems, however, these might differ depending on the characteristics being optimized; hence, the general formula is provided, which uses subscript j to indicate the dimension.

$$x_{i,j}(k=0)\epsilon U\left(x_j^{min}, x_j^{max}\right) \quad (13)$$

Velocities are similarly initialized according to Relation (13). For application problems with a different range of feasible values on one dimension than on another, different step sizes per dimension would make sense; hence, the general form is presented, which avoids unnecessarily imposing the same range of feasible values on all characteristics to be optimized.

$$v_{i,j}(k=0)\epsilon U\left(-v_j^{max}, v_j^{max}\right) \tag{14}$$

Each particle's personal best is initialized to its starting position as shown in Equation (15).

$$\vec{P}_i(k=0) = \vec{x}_i(k=0) \tag{15}$$

The global best is always the best of all personal bests as shown in Equation (16).

$$\vec{\mathrm{g}}(k) = \arg min_{\forall p_t(k)} f|\left(P_t(k)\right) \tag{16}$$

Iterative Optimization Routine

Once the swarm has been initialized, particles iteratively: (i) accelerate (i.e. adjust their velocity vectors) toward the global best and their own personal bests, (ii) update and clamp their velocities, (iii) update their positions, and (iv) update their personal bests and the global best. This routine is repeated until reaching a user-specified termination criterion.

For convenience, the relevant equations are restated below as needed in order of implementation.

$$\vec{v}_i(k+1) = \sum_{i=GP-1}^{i=GP+1}\left(\omega\vec{v}_i(k) + c_1\vec{r}_{1,i}\circ\left(\vec{P}_i(k) - \vec{x}_i(k)\right) + c_2\vec{r}_{2,i}\circ\left(\vec{g}(k) - \vec{x}_i(k)\right)\right) \tag{11}$$

$$\vec{v}_i(k+1) = sign\left(\vec{v}_i(k+1)\right) \times max\left(|\vec{v}_i(k+1)|, v_j^{max}\right) \tag{12}$$

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1) \tag{9}$$

A particle's personal best is only updated when the new position offers a better function value:

$$\vec{P}_i(k+1) = \begin{cases} \vec{x}_i(k+1) if\ f\left(\vec{x}_i(k+1)\right) < f\left(\vec{P}_i(k)\right) \\ \vec{x}_i(k+1)\ otherwise \end{cases} \tag{17}$$

The global best is always the best of all personal bests:

$$\vec{\mathrm{g}}(k+1) = \arg min_{\forall p_t(k)} f\left(P_t(k+1)\right) \tag{18}$$

Rather than accelerating due to external physical forces, particles adjust toward solutions of relative quality. Each position encountered as particles swarm is evaluated and compared to existing bests. Though the behavior of each individual is simple, the collective result is an optimization algorithm capable of maximizing or minimizing problems that would be difficult to tackle with straightforward mathematical analyses, either because the problem is not well understood in advance or simply because the problem is quite complicated.

The acceleration coefficients $c_1$and c_2, together with the random values r_1 and r_2, maintain the stochastic influence of the cognitive and social components of the particle's velocity respectively. The constant $c_1$expresses how much confidence a particle has in itself, while $c_2$expresses how much confidence a particle has in its neighbors. There are some properties of c_1 and c_2 :

Case-1: When, $c_1 = c_2 = 0$ then all particles continue flying at their current speed until they hit the search space's boundary. Therefore, from the equations (5) and (6), the velocity update equation is calculated as

$$\vec{v}_i(k+1) = \vec{v}_i(k) \tag{19}$$

Case-2: When $c_1 > 0$ and $c_2 = 0$, all particles are independent. The velocity update equation will be

$$\vec{v}_i(k+1) = \sum_{i=GP-1}^{i=GP+1}\left(\omega\vec{v}_i(k) + c_1\vec{r}_{1,i}\circ\left(\vec{P}_i(k) - \vec{x}_i(k)\right)\right) \tag{20}$$

On the contrary, when $c_2 > 0$ and $c_1 = 0$, all particles are attracted to a single $(i.e.\ G_{best})$ point in the entire swarm and the update velocity will become

$$\vec{v}_i(k+1) = \sum_{i=Gb-1}^{i=Gb+1}\left(\omega\vec{v}_i(k) + c_2\vec{r}_{2,i}\circ\left(\vec{g}(k) - \vec{x}_i(k)\right)\right) \tag{21}$$

$$\vec{v}_i(k+1) = \sum_{Gb-1}^{Gb+1}\left(\omega\vec{v}_i(k) + c_2\vec{r}_{2,i}\circ\left(G_{best} - \vec{x}_i(k)\right)\right) \tag{22}$$

for $gbest$ BRO.

or

$$\vec{v}_i(k+1) = \sum_{i=0}^{lim_{max}}\left(\omega\vec{v}_i(k) + c_2\vec{r}_{2,i}\circ\left(L_{best} - \vec{x}_i(k)\right)\right) \tag{22}$$

for $lbest$ BRO

Case-3: When $c_1 = c_2$, all particles are attracted towards the average $P^t_{best\,,i}$ and G_{best}.

Case-4: When $c_1 \gg c_2$, each particle is more strongly influenced by its personal best position, resulting in excessive wandering. In contrast, when $c_1 \gg c_2$ then all particles are much more influenced by the global best position, which causes all particles to run prematurely to the optimal.

Normally, c_1 *and* c_2 are static, with their optimized values being found empirically. Wrong initialization of c_1 *and* c_2 may result in divergent or cyclic behavior. From the different empirical researches, it has been proposed that the two acceleration constants should be $c_1 = c_2 = 2$

Therefore, in a BRO method, all particles are initiated randomly and evaluated to compute fitness together with finding the personal best (best value of each particle) and global best (best value of particle in the entire swarm). After that a loop starts to find an optimum solution. In the loop, first the particles' velocity is updated by the personal and global bests, and then each particle's position is updated by the current velocity. The loop is ended with a stopping criterion predetermined in advance.

Basically, two BRO algorithms, namely the Global Best (gbest) and Local Best (lbest) BRO, have been developed which differ in the size of their neighborhoods. Originally, there are two differences between the '$gbest$' BRO and the '$lbest$' BRO: One is that because of the larger particle interconnectivity of the $gbest$ BRO, sometimes it converges faster than the $lbest$ BRO. Another is due to the larger diversity of the $lbest$ BRO; it is less susceptible to being trapped in local minima.

B. BRO Algorithm:

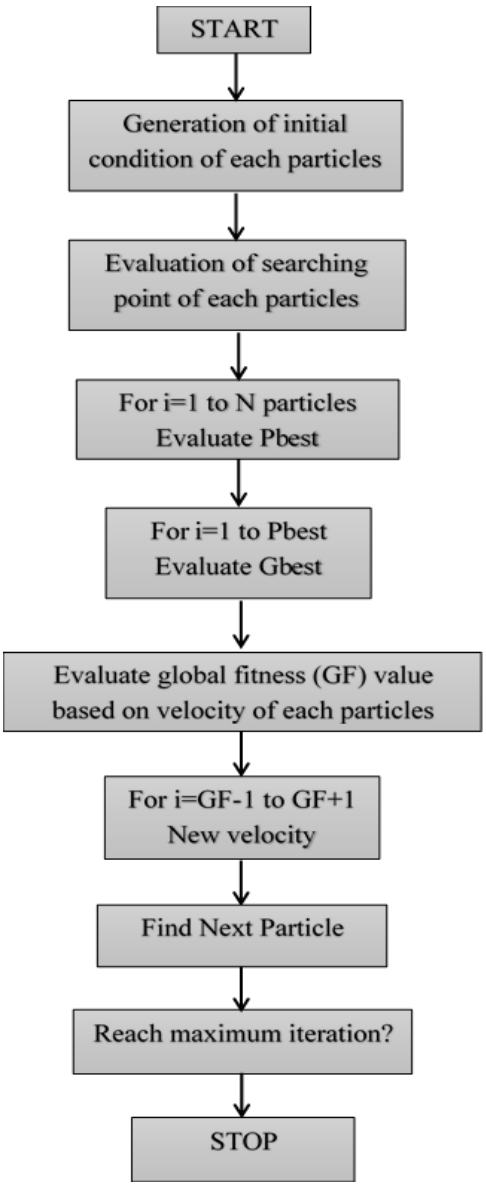

Figure-2: General flowchart for PRO

Input:
m: The swarm size; c_1, c_2: positive acceleration constants;
w: inertia weight
$MaxV$: Maximum velocity of particles
$MaxGen$: Maximum generation
$MaxFit$:Maximum fitness value
Output:
$Pgbest$: Global best position
Begin
Swarms $\{x_{id}, v_{id}\} = Generate(m)$; /* Initialize a population of particles with random positions and velocities on S dimensions*/
$Pbest(i) = 0;\ i = 1, \dots, m, d = 1, \dots, S$
$Gbest = 0;\ Iter = 0;$
$While(Iter < MaxGen\ and\ Gbest < MaxFit)$
$\{For(every\ particle\ i)$
$\{Fitness(i) = Evaluate(i);$
$IF\big(Fitness(i) > Pbest(i)\big)$
$\{Pbest(i) = Fitness(i);\ p_{id} = x_{id};\ d = 1, \dots, S\}$
$IF(Fitness(i) > Gbest)$
$\{Gbest = Fitness(i);\ gbest = i;$
}
}
$Iter = Iter + 1;$
$\}/* R_1\ and\ R_2\ are\ two\ random\ functions\ in\ the\ range\ [0,1] */$
$Detect\{gbest\}$
$While(GF - 1)$
$v_{id} = w * v_{id} + c_1 * R_1 * (p_{id} - x_{id}) + c_2 * R_2 * (p_{gd} - x_{id})$
}
do
$GF = GF + 1$
$Use\ threshold\ level\ of\ global\ fitness\ value\ where\ the\ Global\ Fitness\ Value\ ranges\ from\ Global\ Fitness \pm 1.$

$$\vec{v}_i(k+1) = \sum_{GF-1}^{GF+1} (\ vid = w * vid + c1 * rand() * (pid - xid) + c2 * Rand() * (pgd - xid)$$

$Improves\ the\ efficiency\ of\ the\ iteration\ and\ reduce\ the\ processing.$
End

Function Evaluate()
$For(every\ particle\ i)$
{
$For(every\ d)$
$\{\ v_{id} = w * v_{id} + c_1 * R_1 * (p_{id} - x_{id}) + c_2 * R_2 * (p_{gd} - x_{id})$
$IF(vid > MaxV)$
$\{v_{id} = MaxV;\}$
$IF(vid < -MaxV)$
$\{vid = -MaxV;\}$
$xid = xid + vid$
}
}

Pseudo code:

Step-1: The Input dataset S is taken from Nifty fifty Companies,

Step-2: Initialize position and velocity of all the particles randomly in the N dimension space from dataset S.

Step-3: Evaluate the fitness value of each particle, and update the global optimum position.

Step-4: According to changing of the gathering degree and the steady degree of particle swarm, determine whether all the particles are re-initialized or not.

Step-5: Determine the individual best fitness value. Compare the l_p of every individual with its current fitness value. If the current fitness value is better, assign the current fitness value to l_p.

Step-6: Determine the current best fitness value in the entire population. If the current best fitness value is better than the g_p , assign the current best fitness value to g_p .

Step-7: For each particle, update particle velocity,

Step-8: Repeat the iteration of the particle using $gbest$ fitness value and limit the Iteration of the particle.

Step-9: Update particle position.

Step-10: Repeat Step2 - 7 until a stop criterion is satisfied or a predefined number of iterations are completed. While maximum iterations or minimum error criteria is not attained Particles' velocities on each dimension are clamped to a maximum velocityv_{max} . If the sum of accelerations would cause the velocity on that dimension to exceed v_{max} , which is a parameter specified by the user. Then the velocity on that dimension is limited to v_{max} .

It is necessary to prevent the explosion of the swarm using the parameter v_{max} , where v_{max} is the point of saturation of the velocity, if the velocity of a particle isgreater than $Vmax$ or smaller than $-v_{max}$ it is valorized as v_{max} . If $Vmax$ is toosmall there is not enough exploration beyond locally good regions (may fall into local optimal), if too large can be overcome with good solutions. Other parameters to consider are:

- The number of particles (swarm size).
- The number of generations or iterations.
- The inertia weight (w).
- The reason for cognitive learning (c_1).
- The reason of social learning (c_2).

The recommended values for the parameters are the following:

The number of particles is between 10 and 40. The number of generations is between 100 and 200. While more higher are the values of these parameters increases grows the chance of finding the optimum, it increases grows the computational cost. It is recommended that c_1 = c_2 = 1.5 or c_1 = c_2 = 2, since the low values allow to explore more regions before going to the objective. The weight of the inertia (w) controls the impact of the historical velocity; high values facilitate the global exploration and the small ones the local exploration while an appropriate value produces a balance between global and local search by reducing the amount of generations required. The rule is to give a high initial value and gradually decrease,

$$w(k) = Wmax - (Wmax - Wmin)/Ncmax * k,$$

It is suggested that $Wmax$ = 1.4,$Wmin$ = 0.4. Other interesting alternative to prevent the explosion of the swarmis using the restriction coefficient defined by expression (23)

$$const\ coe\ f\ f = \frac{2}{2-\phi-\sqrt{\phi^2-4\phi}} \quad (23)$$

Where $\varphi = c_1 + c_2 > 4$

III. EXPERIMENTAL RESULTS AND ANALYSIS

The data for the stock market prediction experiments has been collected for Nifty 50. The experimental data used consists of technical indicators and daily prices of the indices. The total number of samples for the stock indices is 3462 trading days. Each sample consists of the closing prices, opening prices, lowest prices, highest prices and total volume of stocks traded for the day. The data is divided into two sets training and testing sets. The training set consists of 365 samples and rest is set aside for testing. All the inputs are normalized to values between -1 to +1. The normalization is carried out by expressing the data in terms of the maximum and minimum value of the dataset.

Date	Open Price	High Price	Low Price	Close Price	Total Volume	No of Trades	Turnover in(Rs.in Lakh)
9/11/2014	223.35	216.1	219.55	222.7	50,78,407	47,593	1,11,65,93,545.00
9/10/2014	222.9	218	219	219.1	35,78,141	29,873	78,85,46,803.50
9/9/2014	224.45	220.5	223.9	221.2	34,24,536	26,656	75,94,06,149.10
9/8/2014	225.8	222.1	223.45	223.85	34,63,052	35,702	77,48,32,714.30
9/5/2014	227.6	221.1	227.1	222.8	51,33,487	46,291	1,14,67,33,300.00
9/4/2014	235.1	226.05	235	227.05	1,06,90,351	55,956	2,43,45,62,223.00
9/3/2014	242	235.35	240.55	236.85	40,41,846	32,857	96,21,29,678.30
9/2/2014	243.1	237	238	240.3	36,34,223	37,961	87,47,04,332.40
9/1/2014	243.5	237.05	240.7	238.5	54,59,840	53,841	1,31,10,24,781.00

Table-1: Indicates Sample Testing dataset of BHEL Company

Parameters	**High Prices**		**Low Prices**	
Dataset	Analysis Set(Training)	Validation Set(Test	Analysis Set(Training)	Validation Set(Test)
Total Objects	143191	3146	143191	3146
Objects Covered	3146	251	3146	251
Min Support	117	178	104.65	172.1
Max Support	2580	577	2561	566.85
Average Support	586.827217	317.8278443	575.993815	313.0538922
Min Accuracy	0.019369048	0.012131716	0.025824442	0.013231013
Max Accuracy	0.045675963	0.026943577	0.061074954	0.029251131
Average Accuracy	0.027031321	0.02191375	0.036029315	1.084262864

Table-2: Statistical Analysis of Sample Input Training Dataset of BHEL Company

Parameters	**Value**
Number lower bound	-5
Number upper bound	5
Population Size	251
Number of iterations each day	251
Acceleration constant	1 2
Acceleration constant	2 2
Initial inertia weight	0.9
Final inertia weight	0.4
Minimum Error Gradient	$1 * 10^{-25}$
Epochs Before Error Gradient Termination	15
ω_{max}	0.975
ω_{min}	0.389

Table-3: BRO Parameters for estimation of coefficients Parameter value

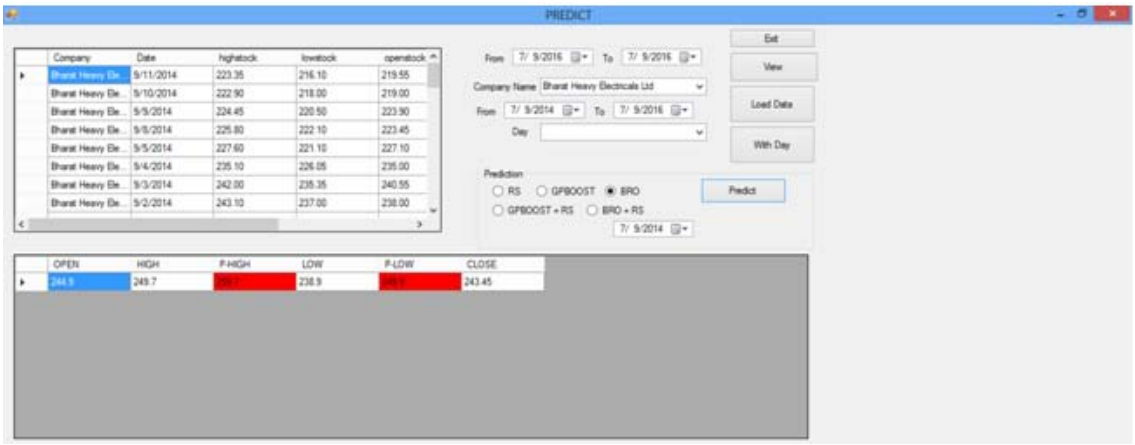

Figure-3: BRO Prediction Result Screen Shot

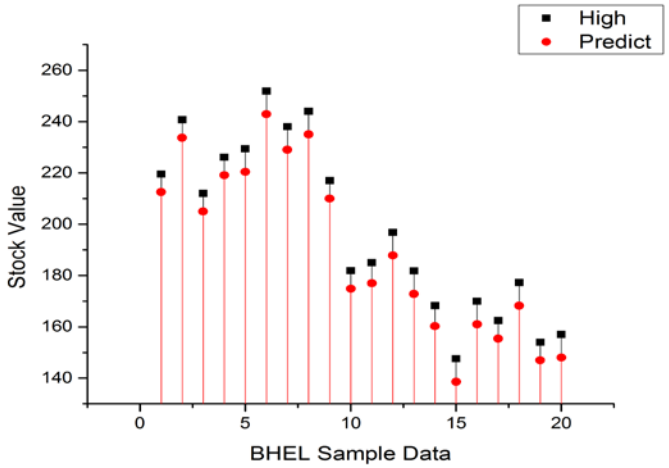

Figure-4: BRO Prediction of High Price Value in in BHEL Company.

Figure 4indicates the Best Replacement Optimization Prediction of High Value in Stock Market Data from Jan 2008 to Sep 2014. The Sample Data is used to detect GP and predict the Low value for BHEL Company.

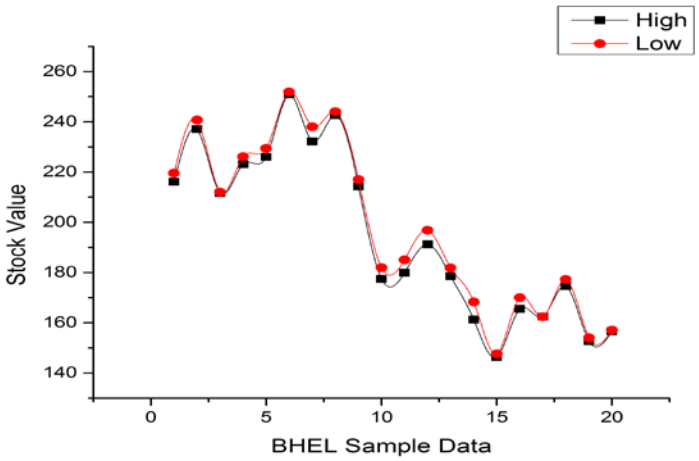

Figure- 5: BRO Prediction of Low Price Value and High Price Value in BHEL Company.

The Figure 5. Indicates the Best Replacement Optimization Prediction of High Value and Low value in Stock Market Data from Jan 2008 to Sep 2014. The Sample Data is used to detect BRO and predict the Low value for BHEL Company.

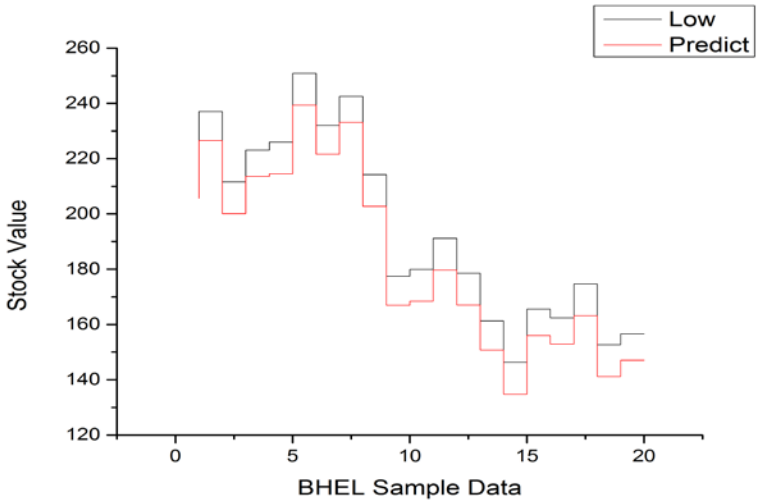

Figure-6: BRO Prediction of Low Price Value in BHEL Company

Figure 6.indicates the Best Replacement Optimization Prediction of Low Value in Stock Market Data from Jan 2008 to Sep 2014. The Sample Data is used to detect GP and predict the Low value for BHEL Company.

IV. CONCLUSION

The BRO model with perturbation is presented to increase the prediction accuracy for long term stock market indices prediction. The PSO represents the passive congregation biological mechanism which enables all particles in the swarm to perform the global search in the whole search space. But, this model every particle has a limitation in iteration based on threshold value of the fitness function, which is the best solution visited by that specific particle. The performance measurements of our proposed model is obviously better than the other standard model in which only particles subjected to performing the global fitness value. The proposed model offers lesser computational complexity, better prediction accuracy ad lesser training time compared to other model. Thus the proposed model is a new promising forecasting model for stock market prediction.

REFERENCES

[1] Aditya Nawani, Himanshu Gupta, Narina Thakur (2013), "Prediction of Market Capital for Trading Firms through Data Mining Techniques", International Journal of Computer Applications (0975 – 8887) Volume 70. No.18.
[2] Ashok kumar D and Murugan S (2013), "Performance Analysis of Indian Stock Market Index using Neural Network Time Series Model", International Conference on Pattern Recognition, Informatics and Mobile Engineering (PRIME), IEEE, 978-1-4673-5845-3.
[3] Atsalakis G.S and Kimon P.V (2009), "Forecasting stock market short-term trends using a neuro-fuzzy methodology", Expert Systems with Applications, vol. 36, no. 7, pp.10696–10707.
[4] Chen, M-Y. (2011), "Predicting corporate financial distress based on integration of decision tree classification and logistic re- gression", Expert Systems with Applications, 38, 11261 11272.
[5] Dopuch, N., Holthausen, R. & Leftwich, R. (1987), "Predicting audit qualifications with financial and market variables", The Accounting Review, 63(3), 431-453.
[6] Kaur A., & Singh M. (2012), 'An Overview of PSO - Based Approaches in Image Segmentation', Published in: Journal of Engineering and Technology, Volume 2, Page(s): 1349- 1357.
[7] Mehrara, M., Moeini A., Ahrari M. *et al.* (2010), 'Using Technical Analysis with Neural Network for Forecasting Stock Price Index in Tehran Stock Exchange', Published in: Middle Eastern Finance and Economics, Volume 6, Page(s): 50-61.
[8] Mitras S. K (2009), "Optimal Combination of Trading Rules Using Neural Networks", International Business Research, vol. 2, no. 1, pp. 86-99.
[9] Mohamed M.M (2010), "Forecasting stock exchange movements using neural networks: empirical evidence from Kuwait", Expert Systems with Applications, vol. 27, no. 9, pp.6302–6309.
[10] Tsang E. P. and Martinez-Jaramillo S, (2004), 'Computational finance', Published in: IEEE Computational Intelligence Society Newsletter, 2004, Page(s): 3–8.
[11] Wilke D.N (2005), 'Analysis of the particle swarm optimization algorithm', Masters Dissertation, University of Pretoria.

Source of support: Nil, Conflict of interest: None Declared