ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 9, November 2012

Load Forecasting of Andhra Pradesh Grid using PSO, DE Algorithms

CH Jeevan Kumar, M Veerakumari

***Abstract*— Load forecasting is an important component for power system energy management system. Precise load forecasting helps the electric utility to make unit commitment decisions, reduce spinning reserve capacity and schedule device maintenance plan properly. Besides playing a key role in reducing the generation cost, it is also essential to the reliability of power systems. Load forecasting plays an important role in power system planning, operation and control. Planning and operational applications of load forecasting requires a certain 'lead time' also called forecasting intervals. On the basis of lead time, load forecasts can be divided into four categories: very short-term forecasts, short-term forecasts, medium-term forecasts and long-term forecasts. In the present paper STLF is attempted and implemented using Matlab7.1 with evolutionary algorithms PSO and DE for the Andhra Pradesh Grid. The Load data of Andhra Pradesh grid of every month from the year 2007to 2011 was collected .Forecasted the Load data of next 5 years as one set and consecutive 5years as another set.**

***Index Terms*— DE and PSO Algorithms.**

I. INTRODUCTION

Electric load forecasting is the process used to forecast future electric load, given historical load and weather information and current and forecasted weather information. In the past few decades, several models have been developed to forecast electric load more accurately .Load forecasting can be divided into three major categories:

A. *Long-term electric load forecasting (LTLF):* used to supply electric utility company management with prediction of future needs for expansion, equipment purchases, or staff hiring. LTLF the prediction time can be as long as 10 years and above. A precise long term load-forecasting is essential for monitoring and controlling power system operation.

B. *Medium-term load forecasting (MTLF):* used for the purpose of scheduling fuel supplies and unit maintenance .MTLF the prediction time is 2-5years.

C. *Short-term load forecasting (STLF):* used to supply necessary information for the system management of day-to-day operations and unit commitment. STLF the prediction time is every next hour, day by day, week by week and monthly.

With the recent trend of deregulation of electricity markets, STLF has gained more importance and greater challenges. In the market environment, precise forecasting is the basis of electrical energy trade and spot price establishment for the system to gain the minimum electricity purchasing cost. In the real-time dispatch operation, forecasting error causes more purchasing electricity cost or breaking-contract penalty cost to keep the electricity supply and consumption balance. There are also some modifications of STLF models due to the implementation of the electricity market.

One of the objectives of this paper is to find a way to detect the erroneous data, eliminate them and evaluate the real data, another objective of this paper is to develop some new and practical models and algorithms with some up-to-date techniques. The power system operators always have very good intuition in manual load forecasting with their long time working experience. Therefore it is an attempt to combine the operators' experience with the presented models in a convenient way.

II. LOAD FORECASTING OVERVIEW

2.1 Characteristics of the Power System Load:

The system load is the sum of all the consumers' load at the same time. The objective of system STLF is to forecast the future system load. Good understanding of the system characteristics helps to design reasonable forecasting models and select appropriate models in different situations. Various factors influence the system load behavior, which can be mainly classified into the following categories

- Weather
- Time
- Economy
- Random disturbance

2.2 Classification of Developed Load Forecasting Methods:

In terms of lead time, load forecasting is divided into four categories:

- Long-term forecasting with the lead time of more than one year
- Mid-term forecasting with the lead time of one week to one year
- Short-term load forecasting with the lead time of 1 to 168 hours
- Very short-term load forecasting with the lead time shorter than one day

The research approaches of load forecasting can be mainly divided into two categories: statistical methods and artificial

***CH Jeevan Kumar**, Electrical and Electronics Engineering, Sir C.R.R College of Engineering, Eluru, INDIA, Phone/ Mobile No. 9885876752.*

***M Veerakumari,** Electrical and Electronics Engineering Sir C.R.R College of Engineering, .Eluru, INDIA, Phone/Mobile No. 9492225471.*

intelligence methods [1]. In statistical methods, equations can be obtained showing the relationship between load and its relative factors after training the historical data, while artificial intelligence methods try to imitate human beings' way of thinking and reasoning to get knowledge from the past experience and forecast the future load.

2.2.1 Time Series Model:

Time series methods are based on the assumption that the data have an internal structure, such as autocorrelation, trend or seasonal variation. The methods detect and explore such a structure. Time series have been used for decades in such fields as economics, digital signal processing, as well as electric load forecasting. A time series is a sequence of data points, measured typically at successive time instants spaced at uniform time intervals. Time series analysis comprises methods for analyzing time series data in order to extract meaningful statistics and other characteristics of the data. Time series forecasting is the use of a model to predict future values based on previously observed values. Time series analysis is also distinct from spatial data analysis where the observations typically relate to geographical locations. Models for time series data can have many forms and represent different stochastic processes. When modeling variations in the level of a process, three broad classes of practical importance are the autoregressive (AR) models, the integrated (I) models, and the moving average (MA) models. These three classes depend linearly on previous data points. Combinations of these ideas produce autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) models. The autoregressive fractionally integrated moving average (ARFIMA) model generalizes the former three. Fan and McDonald [2] and Cho et al. [3] described implementations of ARIMAX models for load forecasting. Extensions of these classes to deal with vector-valued data are available under the heading of multivariate time-series models and sometimes the preceding acronyms are extended by including an initial "V" for "vector".

2.2.2 Neural Networks:

The use of artificial neural networks (ANN or simply NN) has been a widely studied load forecasting technique since 1990 [5]. Neural networks are essentially non-linear circuits that have the demonstrated capability to do non-linear curve fitting. The outputs of an artificial neural network are some linear or non-linear mathematical function of its inputs. The inputs may be the outputs of other network elements as well as actual network inputs. In practice network elements are arranged in a relatively small number of connected layers of elements between network inputs and outputs. Feedback paths are sometimes used. The most popular artificial neural network architecture for load forecasting is back propagation. This network uses continuously valued functions and supervised learning. That is, under supervised learning, the actual numerical weights assigned to element inputs are determined by matching historical data (such as time and weather) to desired outputs (such as historical loads) in a pre-operational "training session". Artificial neural networks with unsupervised learning do not require pre-operational training.

Bakirtzis et al. [6] developed an ANN based short-term load forecasting model for the Energy Control Center of the Greek Public Power Corporation. In the development they used a fully connected three-layer feed forward ANN and a back propagation algorithm was used for training. Input variables include historical hourly load data, temperature, and the day of week. The model can forecast load profiles from one to seven days. Also Papalexopoulos et al. [7] developed and implemented a multi-layered feed forward ANN for short-term system load forecasting. In the model three types of variables are used as inputs to the neural networks: seasonal related inputs, weather related inputs, and historical loads. Khotanzad et al [8] described a load forecasting system known as ANNSTLF. It is based on multiple ANN strategy that captures various trends in the data. In the development they used a multilayer perceptron trained with an error back propagation algorithm. ANNSTLF can consider the effect of temperature and relative humidity on the load. It also contains forecasters that can generate the hourly temperature and relative humidity forecasts needed by the system. An improvement of the above system was described in [9]. In the new generation, ANNSTLF includes two ANN forecasters: one predicts the base load and the other forecasts the change in load. The final forecast is computed by adaptive combination of these forecasts. The effect of humidity and wind speed are considered through a linear transformation of temperature. At the time it was reported in [8], ANNSTLF was being used by 35 utilities across the USA and Canada. Chen et al. [10] also developed a three layer fully connected feed forward neural network and a back propagation algorithm was used as the training method. Their ANN though considers electricity price as one of the main characteristics of the system load. Many published studies use artificial neural networks in conjunction with other forecasting techniques such as time series [11] and fuzzy logic [12].

2.2.3 Expert Systems:

Rule-based forecasting makes use of rules, which are often heuristic in nature, to do accurate forecasting. Expert systems incorporate rules and procedures used by human experts in the field of interest into software that is then able to automatically make forecasts without human assistance. Ho et al. [13] proposed a knowledge-based expert system for the short-term load forecasting of the Taiwan power system. Operators' knowledge and the hourly observation of system load over the past five years are employed to establish eleven day-types. Weather parameters were also considered. Rahman and Hazim [14] developed a site-independent technique for short-term load forecasting. Knowledge about the load and the factors affecting it is extracted and represented in a parameterized rule base. This rule-based system is complemented by a parameter database that varies from site to site. The technique is tested in different sites in the United States with low forecasting errors. The load model, the rules and the parameters presented in the paper have been designed using no specific knowledge about any particular site. Results improve if operators at a particular site

are consulted.

2.2 Requirements of the Load Forecasting Process

In nearly all the energy management systems of the modern control centers, there is a short-term load forecasting module. A good Load Forecasting system should fulfill the requirement of accuracy, fast speed, automatic bad data detection, friendly interface, automatic data access and automatic forecasting result generation.

- Accuracy
- Fast Speed
- Automatic Bad Data Detection
- Friendly Interface
- Automatic Data Access
- Automatic Forecasting Result Generation

III. PARTICLE SWARM OPTIMIZATION & DIFFERENTIAL EVOLUTION

The particle swarm simulates the kind of social optimization, a problem is given, and some were to evaluate a proposed solution to it exits the form of fitness function. A communication structure or social network is defined; assigning neighbors for each individual for interact with. Then a population of individuals defined as random guesses at the problem solutions is initialized .These individuals are candidate solutions .they are also known as the particles, hence the name particle swarm an iterative process to improve these candidate solution is set in motion.

The particles iteratively evaluate the fitness of candidate solutions and remember the location where they had the better success the individuals best solution is called the particle best or the local best. Each particle makes this information availability of their neighbors. They are also able to see where their neighbors have had success. Movements through the search space are guided by these successes, with the population usually converging, by the end of trial, on a problem solution better than that of non-swarm approach using the same methods.

The swarm is typically modeled by particles in multidimensional space that have a position and a velocity. These particles fly through hyperspace and have two essential reasoning capabilities: their memory of their own best position and knowledge of the global or their neighborhood's best. In a minimization optimization problem, "best" simply meaning the position with the smallest objective value Members of swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. So a particle has the following information to make a suitable change in its position and velocity. A global best that is known to all and immediately updated when a new best position is found by any particle in the swarm Neighborhood best that the particle obtains by communication with subset of the swarm. The local best, which is the best solution that the particle has seen;

As the swarm iterates, the fitness of the global best solution improves. It could happen that all particles being influenced by the global best eventually approach the global best, and there on the fitness never improves despite however many runs the PSO is iterated thereafter. The particles also move about in the search space in close proximity to the global best and not exploring the rest of search space. This phenomenon is called 'convergence'. If the inertial coefficient of the velocity is small, all particles could slowdown until they approach zero velocity at the global best. The selection of coefficients in the velocity update equations affects the convergence and the ability of the swarm to find the optimum. One way to come out of the situation is to reinitialize the particles positions at intervals or when convergence is detected.

3.1 Parameter Selection in Particle Swarm Optimization

Particle swarm optimization (PSO) is a novel optimization method [15]. To ensure convergence of PSO adjustments of various parameters need to be carefully adjusted in order to achieve better performance of the algorithm. In the subsequent section, the detailed implementation strategies of the PSO are described.

Inertia Weight:

The inertia weight *w* is employed to control the impact of the previous history of velocity, thus to influence the tradeoff between global (wide ranging) and local (near by) exploration abilities of the "flying points". In PSO, the balance between the global and local exploration abilities is mainly controlled by inertia weights. w often decreases linearly from about 0.9 to 0.4 during the run.

$$\omega = \omega \max - \left[\frac{\omega \max - \omega \min}{iter \max} \right] * iter$$

Where

w - Inertia weight factor

w_{max} - maximum value of weighting factor

w_{min} - minimum value of weighting factor

$iter_{max}$ - maximum number of iterations

iter - current number of iteration

Acceleration Constant:

The constants c_1 and c_2 represent the weighting factor and are tuned in the process. The constants c_1 and c_2 represent the weighting factor of the acceleration terms that pull each particle toward the p_{best} and g_{best} positions. Low values allow particles to roam far from the target regions before being tugged back.

Swarm Size:

The swarm size or the number of individuals inside the population is determined by the integer parameter 'P'. For very small values of P the possibility of being trapped in local optima is very likely. Larger population will increase the computation time requirements.

Number of Iterations:

It is also a user defined parameter that determines the number of iterations for which algorithm has to run. Based upon computational experience a few hundred of iterations are usually sufficient to observe significant improvement in the solution providing that the initial solutions are feasible.

Velocity of Particle and its Updating:

One of the important factors that affect performance of the PSO, Specifically the speed of the convergence is the particle's velocity of 'flying' inside the problem space. This parameter limits the steps taken by particles at every an

iteration. A small value can cause the particle to get trapped in local optima; on the other hand, a too large value can cause oscillation around a certain position. This problem of proper selection of velocity can be eliminated by using adaptive velocity to PSO's position updating. The following equation represents the velocity update equation.

$V_{id} = W \times V_{id} + C_1 \times \text{rand}( ) \times (P_{id} - X_{id}) + C_2 \times \text{Rand}( ) \times (P_{gd} - X_{id}) \quad \rightarrow (1)$

Where c_1 and c_2 are positive constants, called cognitive and social parameters respectively, both are equal to 2 in general cases. 'W' is inertia weight factor; a large weight factor facilitates a global search while a small inertia weight facilitates a local search. Shi asserted that, by linearly decreasing inertia weight from a relatively large value to a small one through the course of the PSO run, the PSO tends to have more global search ability at the beginning of the run and have more local search ability near the end of the run. He suggested that an inertia weight starting from 0.9 linearly decreasing to 0.4 during a run be adopted to give the PSO a better performance.

3.2 Differential Evolution:

Differential evolution was first proposed over 1994–1996 by Storn and Price at Berkeley as a new evolutionary algorithm (EA) [4]. Differential evolution (DE) is a stochastic direct search optimization method. It is generally considered as an accurate, reasonably fast, and robust optimization method. The main advantages of DE are its simplicity and therefore easy use in solving optimization problems requiring a minimization process with real-valued and multimodal (multiple local optima) objective functions. DE uses a non uniform crossover that makes use of child vector parameters to guide through the minimization process. The mutation operation with DE is performed by arithmetical combinations of individuals rather than perturbing the genes in individuals with small probability compared with one of the most popular EAs, genetic algorithms (GAs). Another main characteristic of DE is its ability to search with floating point representation instead of binary representation as used in many basic EAs such as GAs. The characteristics together with other factors of DE make it a fast and robust algorithm as an alternative to EA, and it has found an increasing application in a number of engineering areas including power engineering.

According to Price [16], the main advantages of a DE include

- Fast and simple for application and modification
- Effective global optimization capability
- Parallel processing nature
- Self-referential mutation operation
- Effective on integer, discrete, and mixed parameter optimization
- Ability to handle non differentiable, noisy, and/or time-dependent objective functions

Initial Population:

DE is a parallel direct search method using a population of N parameter vectors for each generation. At generation G, the population P^G is composed of X_i^G , i = 1, 2. . . N. The initial population P^{G0} can be chosen randomly under uniform probability distribution if there is nothing known about the problem to be optimized:

$$X_i^G = X_{i(L)} + rand_i[0,1] * (X_{i(H)} - X_{i(L)}) \quad \rightarrow (2)$$

where $x_{i(L)}$ and $x_{i(H)}$ are the lower and higher boundaries of d-dimensional vector $x_i = \{x_{j,i}\} = \{x_{1,i}, x_{2,i}, \ldots, x_{d,i}\}^T$. If some a priori knowledge is available about the problem, the preliminary solution can be included to the initial population by adding normally distributed random deviations to the nominal solution.

Mutation:

The objective of mutation is to enable search diversity in the parameter space as well as to direct the existing object vectors with suitable amount of parameter variation in a way that will lead to better results at a suitable time. It keeps the search robust and explores new areas in the search domain. For real parameter optimization, mutation is the process of adding a randomly generated number to one or more parameters of an existing object vector. Zero mean probabilistic distribution should be used in generating mutation vectors. For real parameter optimization, a dynamically scalable zero means probabilistic distribution should be used in generating the mutation vector for each parameter. This is usually achieved by self-adaptation of the mutation process in evolutionary strategies. In DE, the mutation vectors are generated by adaptively scaling and correlating the output of predefined, multivariate probability distribution. The DE mutation of a vector is achieved by adding the weighted difference of two randomly selected vectors as in (3):

$$X_i^{'G+1} = X_i^G + f_1 \bullet (X_{r1}^G - X_{r2}^G) \quad \rightarrow (3)$$

Where G represents the G^{th} generation; r1 ≠ r2 ≠ i, and r1, r2 are randomly selected integers within the population size N, that is, r1, r2 ϵ {1, 2, . . . , N}. The condition that r1, r2 ≠ i is to avoid the mutation process becomes an arithmetic crossover process. Otherwise, without losing generality, if we assume r2 = i, we have (4) as

$$X_i^{'G+1} = X_i^G + f_2 \bullet (X_{r1}^G - X_i^G) \quad \rightarrow (4)$$

Clearly, (4) is a linear combination of two vectors. It is not a mutation but rather a arithmetic crossover. Both (3) and (4) can be used to guide the search process, however in different ways.

The mutation vector defined in (3) does not have any information of the original vector x_i^G. The difference defined by randomly selected (x^G_{r1} _ x^G_{r2}) has zero mean, and the scale factor f_1 only changes the scale without introducing bias into the search process. Population diversity is ensured with (3). In comparison, the crossover vector obtained in (4) contains the original vector x_i^G with only one addition vector x_{r1}^G. The constant scaling factor f_2 together with the vector (x^G_{r1} _ x^G_i) tends to move the newly generated vector $x_i^{'G+1}$ either toward or away from x_{r1}^G. As a result, a nonzero f_2 introduces a bias in searching process through (4).

$$X_i^{'G+1} = (X_{r1}^G + X_{r2}^G + X_{r3}^G)/3 + (p_2 - p_1) \bullet (X_{r1}^G - X_{r2}^G) + (p_3 - p_2) \bullet (X_{r2}^G - X_{r3}^G) + (p_1 - p_3) \bullet (X_{r3}^G - X_{r1}^G) \quad \rightarrow (5)$$

The trigonometric mutation operation of (5) biases the new trial solution heavily in the direction of the best one of three individuals chosen for mutation. It is a local search operator. According to Price, after comparison with other mutation operations, DE mutation can be considered as globally

correlated and therefore effectively enhances the DE's global optimization capabilities.

Crossover:

Crossover or recombination is the main operator for GAs and a complementary process for DE. Crossover aims at reinforcing prior successes by generating child individuals out of existing individuals or object vector parameters. The basic recombination process is a discrete recombination. The crossover constant CR is used to determine if the newly generated individual is to be recombined. Alternatively, the arithmetic crossover formula of (4) can be used to achieve the rotational invariance, which is otherwise difficult to achieve with the discrete recombination. The resulting expression of the mutation and crossover processes are given in (6) as a combination of (3)–(4),

$$X_i^{'G+1} = X_i^G + f_2 \bullet (X_{r3}^G - X_i^G) + F \bullet (X_{r1}^G - X_{r2}^G) \quad \rightarrow (6)$$

Where the randomly generated integers r1 ≠ r2 ≠ r3 ≠ i , F is the mutation constant, and f2 controls the crossover constant. f_2 can take from [0, 1] and remain constant throughout the evolution process. Generally, with a population size of 20d (d is the problem dimension), F = 0.8 and f2 = 0.5 appear to be reasonably good value to start a DE process. It can be seen that f2 = 1 is the discrete recombination model with CR = 1, and f2 = 0 represents a mutation-only model. With the discrete crossover probability CR ∈ [0, 1], starting from a lower limit of 0 for separable objective functions, a satisfactory range of CR appears to be within 0.8–1.0.

IV. RESULT ANALYSIS

A Matlab code was written to execute the PSO and DE algorithms .Using the actual data of Andhra Pradesh grid, the Load demand of 2012 -2016 years is forecasted using the two Algorithms.

Table.1 Monthly load data of Andhra Pradesh state Grid had been considered for 2007 to 2011

	2007	2008	2009	2010	2011
JAN	7520	8656	9262	9990	10848
FEB	8267	8933	9780	10448	11232
MAR	8641	9162	9997	10880	11829
APR	8337	8486	9934	10396	11579
MAY	7692	8503	9062	9353	10474
JUN	7273	8145	8763	9041	10108
JUL	7866	8515	9951	8904	10542
AUG	8607	8590	10294	9740	11377
SEP	8054	8926	10151	9883	11522
OCT	8301	93z4	6849	10428	11591
NOV	7494	8958	9802	9106	10835
DEC	8157	8482	9309	9683	10453

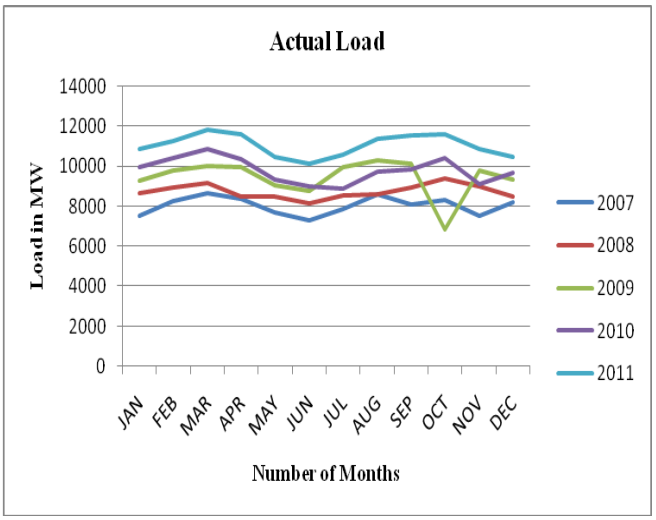

Fig.1 Load profile of Andhra Pradesh January to December (2007 to 2011) in MW

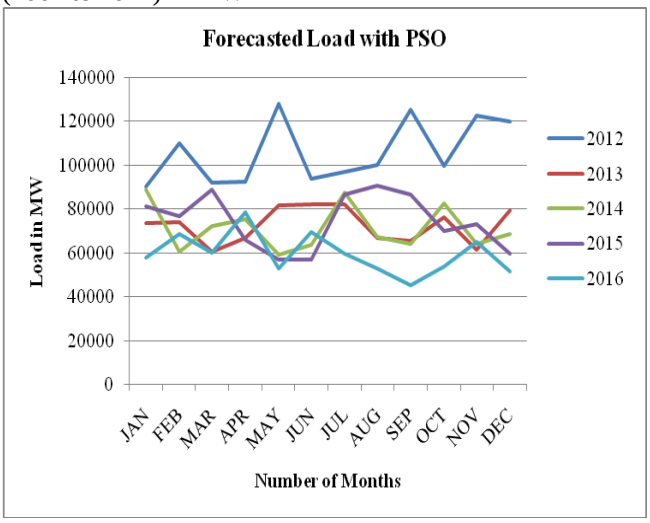

Fig.2 Load profile of Andhra Pradesh January to December (2012 to 2016) in MW with

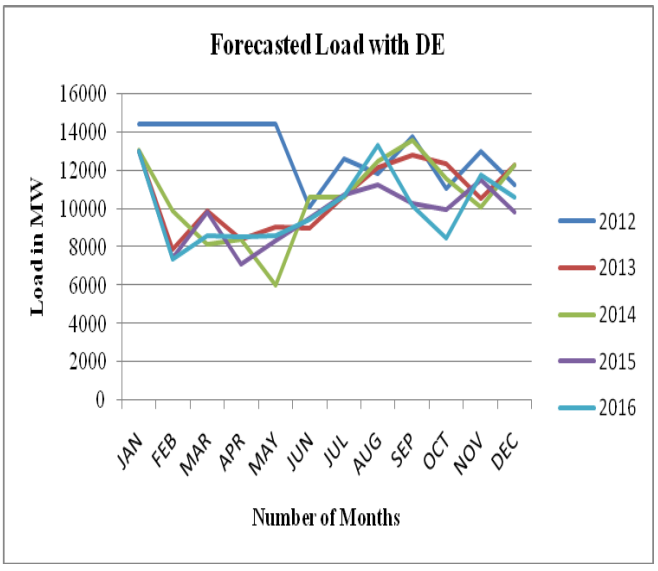

Fig.3 Load profile of Andhra Pradesh January to December (2012 to 2016) in MW with DE

V. CONCLUSION

The main purpose of the paper was to investigate application of computational intelligence methods (PSO &DE) in short term load forecasting. Due to the evolutionary algorithms PSO and DE the load demand in future can be forecasted betterly and with in short span of time, when compare to the conventional methods though there is increase in the year 2012, there is drastic variation in 2012 to 2016 in the results of PSO and DE algorithms. The numerical values of the PSO and DE are given bellow.

Table.2 comparison of load in (MW) for the December month using PSO and DE

YAER	MONTH	PSO	DE
2012	DEC	119800	11271
2013	DEC	79130	12281
2014	DEC	68550	12268
2015	DEC	59660	9851
2016	DEC	51750	10610

Differential evolution achieves better solution and requires less CPU time than conventional method.

REFERENCES

[1] G. Gross, F. D. Galiana, 'Short-term load forecasting', Proceedings of the IEEE, 1987, 75(12), 1558 – 1571.
[2] J.Y. Fan, J.D. McDonald, 'A real-time implementation of short – term load forecasting for distribution power systems', IEEE Transactions on Power Systems, 1994, 9, 988 – 994.
[3] M.Y. Cho, J.C. Hwang, C.S. Chen, 'Customer short-term load forecasting by using ARIMA transfer function model', Proceedings of the International Conference on Energy Management and Power Delivery, EMPD, 1995, 1, 317 – 322.
[4] Storn R, Price K. Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, March 1995. Available at ftp.ICSI.Berkeley.edu/pub/techreports/1995/tr-95-012.ps.Z.
[5] M. Peng, N.F. Hubele, G.G. Karady, 'Advancement in the application of neural networks for short-term load forecasting', IEEE Transactions on Power Systems, 1992, 7, 250 – 257.
[6] A.G.Bakirtzis, et al., 'A neural network short-term load forecasting model for the Greek power system', IEEE Transactions on Power Systems, 1996, 11, 858 – 863.
[7] A.D. Papalexopoulos, S. Hao, T.M. Peng, 'An implementation of a neural network based load forecasting model for the EMS', IEEE Transactions on Power Systems, 1994, 9, 1956 – 1962.
[8] A. Khotanzad, et al., 'ANNSTLF - A neural-network-based electric load forecasting system', IEEE Transactions on Neural Networks, 8 (1997), 835 – 846.
[9] A. Khotanzad, R.A.Rohani, D. Maratukulam, 'ANNSTLF – Artificial neural network shortterm load forecaster - Generation three', IEEE Transactions on Neural Networks, 1998, 13, 1413 – 1422.
[10] B.J. Chen, M.W. Chang, C.J. Lin, 'Load forecasting using support vector machines: A study on EUNITE competition 2001', http://www.csie.ntu.edu.tw/˜cjlin/libsvm. 2002.
[11] T.W.S. Chow, C.T. Leung, 'Nonlinear autoregressive integrated neural network model for short-term load forecasting', IEE Proceedings, Generation, Transmission and Distribution, 1996, 143, 500 – 506.
[12] S.E. Skarman, M. Georgiopoulous, 'Short-term electrical load forecasting using a fuzzy ARTMAP neural network', Proceedings of the SPIE, (1998), 181 - 191.
[13] K.L. Ho et al, 'Short-term load forecasting of Taiwan power system using a knowledge based expert system', IEEE Transactions on Power Systems, 1990, 5, 1214 – 1221.
[14] Kennedy and R. Eberhart, Swarm Intelligence, USA, Morgan Kaufmann Publishers, 2001.
[15] Storn R, Price K. Minimizing the real functions of the ICEC'96 contest by differential evolution. Proc. of IEEE Int. Conf. on Evolutionary Computation, Nagoya, Japan, 1996.
[16] Eiben AE, Smith JE. Introduction to evolutionary computing. Berlin: Springer; 2003.

CH Jeevan Kumar received the B.Tech Degree in Electrical and Electronics Engineering from Chirala Engineering College, Chirala, in 2010. He is M.E student at the Sir C R Reddy College of Engineering, Eluru, India. His research intrests in load forecasting.

M Veerakumari received the B.E Degree in Electrical and Electronics Engineering from RVR & JC College of Engineering, Guntur, in 2001. She has received her M.E Degree in Power systems from Osmania University, Hyderabad, 2003. She is currently doing her Ph.D in Andhra University, Vishakapatnam. Presently she is working as Sr.Assistant Professor in the department of Electrical and Electronics Engineering at Sir C R Reddy college of Engineering, Eluru. Her research interests are Load Forecasting, Deregulations of power systems, Power system planing.