SOFTWARE EFFORT ESTIMATION USING GENETIC ALGORITHM

Mari Kumar B[1], Dr. Latha P[2], Praynlin E[3]

[1,3]Government College of Engineering, Tirunelveli, India

[2] Department of Computer Science and Engineering, Anna University

[1]marikumar106@gmail.com, [2]plathamuthuraj@gmail.com, [3]praynlin25@gmail.com

Abstract

A feed forward back propagation neural network is most commonly used to the form of artificial neural network. This algorithm being a correct procedure, it accurate result in the neural network. The estimate of this method as the training of Neural Network is compared with that of genetic algorithm, that the form of based on estimate the software effort estimation. The comparison of two methods is used to accuracy of the software effort estimation.

***Keywords**:* Artificial neural network (ANN), Feed forward back propagation neural network (FFBPNN), Genetic Algorithm (GA), Software Effort Estimation.

I. INTRODUCTION

Software effort estimation is the prediction of hours of work is done and the number of workers needs to develop a project [2].The machine learning method by using the neural networks. A neural network has been found as one of the best techniques for software cost estimation. Numerous researchers and scientists are constantly working on developing new software cost estimation techniques using neural networks. Adaptive learning machine based on neural network to estimate the software cost using COCOMO model. Feed forward back propagation has a multi-layer architecture with one or more hidden layer(s) between its input and output layers [3]. A genetic algorithm (GA) is a search technique used in computing to find true or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and recombination. This paper investigates the efficiency of FFBPNN and GA in terms of coverage the accuracy for training a neural network can estimate the software effort estimation.

II. ARTIFICIAL NEURAL NETWORK

Artificial neural systems can be considered as simplified mathematical models of brain-like systems and they function as parallel distributed computing networks. However, in contrast to conventional computers, which are programmed to perform specific task, most neural networks must be taught, or trained. They can learn new associations, new functional dependencies and new patterns. The main idea for the field of Neural Networks (NN) originated from the desire to produce artificial systems capable of sophisticated, perhaps “intelligent”, computations similar to the biological neurons in brain structures. Neural networks consist of layers of interconnected nodes, where each node produces a non-linear function of its input [17]. The nodes in the network are divided into the ones from the input layer going through the network to the ones at the output layer through some nodes in a hidden layer. The NN process starts by developing the structure of the network and establishing the technique used to train the network with using an existing data set. Therefore, there are three main entities: the neurons (nodes), the interconnection structure, and the learning algorithm. The most common technique in the use of the neural network for prediction is known as back-propagation trained feed-forward networks. Neural networks have been used in the software reliability modelling domain as well as software risk analysis[18]. Neural network architectures are divided into two groups:

- feed-forward networks where no loops in the network path occur and
- feedback networks that have recursive loops of the different architectures,

the feed-forward back propagation is the most commonly used in the algorithm[19] An Artificial neural network is usually defined as a network composed of a large number of simple processors (neuron) that are massively interconnected, operate in parallel, and learn from experience. These are the primary known characteristics of biological neural system that are the easiest to exploit in artificial neural system. The input units are merely distribution units, which provide all of the measured variables to all of the neurons[16]. Neural networks training time is very much affected by the size of the training data and by the network architecture. Large numbers of records. high bmensionality of each record. number of layers in the network, and number of artificial neurons, are all important factors that affect the amount of training time a model[15]. Many different models of neural network have been proposed for software effort estimation. The feed forward with Back propagation learning algorithm are most commonly used in the effort estimation area. The networks are connected in the neurons are arranged in layers in to the forms in shown in figure 1. This paper network generates output (effort) by back propagation is initial inputs (attributes) through subsequent layers of processing element to the output layer[4].

The use of the neural network approach to estimate the software effort requires certain decisions and choices about the architecture, learning algorithm and the activation functions [5].

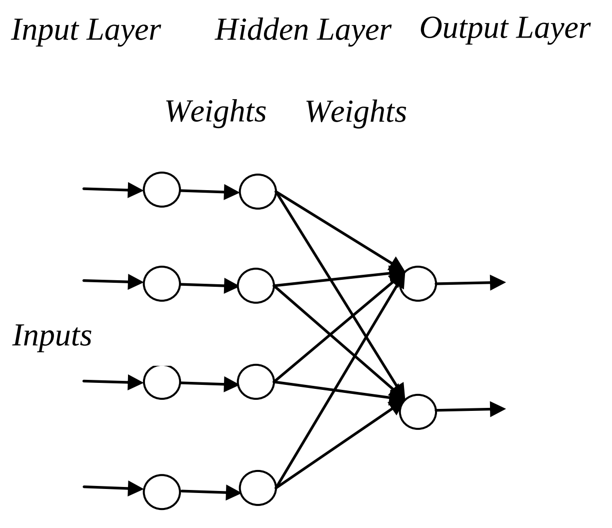

Fig. 1 Neural Network Architecture for Software Development Effort

A. Feed Forward Back Propagation Network

The back propagation (BP) algorithm was developed by Paul Werbos in 1974. Based on LMS algorithm, BP applies a weight correction to the neural network connection weights which is proportional to the partial derivative of the error function [2]. This adjustment to the weights is in the negative direction of the gradient of the error (steepest descent).

The error function is defined as: $E(t) = \frac{1}{2} . |e(t)|^2$ (1)

Where e(t) is the error value, i.e. the difference between the actual output of the plant and the estimated neuro identifier output. The neuro identifier weights are adjusted according to:

$$W(t+1) = W(t) - \eta . \frac{\partial E(t)}{\partial w(t)} \quad (2)$$

where η is the learning rate parameter. A large learning rate might lead to oscillations in the convergence trajectory, while a small learning rate provides a smooth trajectory at the cost of slow convergence speed. Back propagation can be applied in two modes: sequential and batch mode. The former is the online mode of training a neural network, where weight updating is performed after the presentation of each training sample, while in the latter the weight matrices are updated after the presentation of all the training samples that constitute an epoch [1].

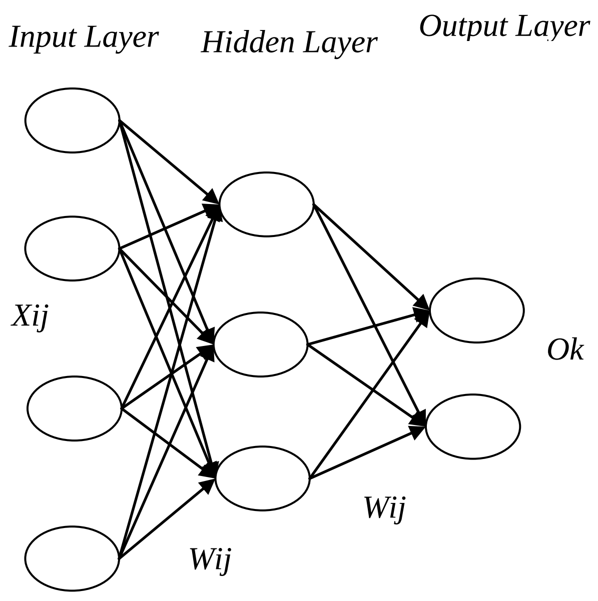

Fig. 2 Multi-Layer Feed Forward Back propagation Neural Network

III. GENETIC ALGORITHM

Genetic Algorithms are search algorithms that are based on concepts of natural selection and natural genetics. Genetic algorithm was developed to simulate some of the processes observed in natural evolution, a process that operates on chromosomes (organic devices for encoding the structure of living being). The genetic algorithm differs from other search methods in that it searches among a population of points, and works with a coding of parameter set, rather than the parameter values themselves. It also uses objective function information without any gradient information. The transition scheme of the genetic algorithm is probabilistic, whereas traditional methods use gradient information. Because of these features of genetic algorithm, they are used as general purpose optimization algorithm. They also provide means to search irregular space and hence are applied to a variety of function optimization, parameter estimation and machine learning applications [6].

Genetic algorithm was used to fine tune the weights of the modification. The initial weight-vector was set by our developer experience. The algorithm was iterated on the basis of previously set scenario. Our assumption was that the genetic algorithm should converge to the suitable weights, which should provide a more accurate estimation. As previously described, a weighted sum of the count of modification group was used as the target function. A genetic algorithm (ga) was used to fine tune the weights of each group. The individuals identified by its chromosome, which is a vector over the real numbers with the same dimension. In the model each chromosome represents a weight-vector, and every element determines the weight of a single modification group. The fitness value is calculated for each individual by evaluating the model with the weights defined in that particular individual. The final goal of the ga is to improve the precision of the model. For classification problems, the F-measure value can give a reliable approximation of the accuracy. The base model was not enhanced with the ga, but it was evaluated using the F-measure. Thus the F-measure was chosen to be the fitness value of the ga. An evolution step starts with the breeding process which consists of two steps, first, the ga selects the two best entities with its fitness value. The crossover operator will apply only this pair. Every call of the crossover operator produces exactly one child. The algorithm repeats the operation to produce more than one child. We used a uniform crossover logic.

During the crossover the algorithm iterates via the elements of the chromosome (vector) and randomly chooses an element from one of the two parents. Every element has the same chance to be copied into the child's chromosome . The chromosome of the children is subject to mutation. A lower limit and an upper limit were determined for the weights of the groups. During the mutation some elements (weights) of the chromosome change. The algorithm gets the half of the distance between the limits and the current selected weight and sets the current value either to the lower or to the upper half point. This way, the two limits are never exceeded. Then, the child is included in the population. The individuals with the worst fitness value are killed (removed from the population) to maintain the size of the population, this way the current evolution step is completed and the algorithm proceeds to the next generation [20].

Selection is based on fitness, i.e. the fitter an individual the greater the chance for this individual to get selected for reproduction and contribute offspring for the next generation.

To generate good offspring, a proficient parent selection mechanism is necessary. This is a process used to determine the number of trials for one particular individual used in reproduction. The chance of selecting one chromosome as a parent should be directly proportional to the number of offspring produced and presented three measures of performance of selection algorithms, bias, spread, and efficiency. Bias defines the absolute difference between actual and expected selection probabilities of individuals. Spread is the range in the possible number of trials that an individual may achieved. Efficiency is related to the overall time complexity of the algorithms. Roulette wheel selection tends to give zero bias but potentially inclines to spread unlimitedly. It can generally be implemented with time complexity of the order of N log N where N is the population size. Stochastic universal sampling (SUS) is another single-phase sampling algorithm with minimum spread, zero bias and the time complexity of SUS is in the order of N [SI. There are other methods can be used such as the ranking scheme [21]. This introduces an alternative to proportional fitness assignment. The chromosomes are selected proportionally to their rank rather than actual evaluation values. It has been shown to help in the avoidance of premature convergence and to speed up the search when the population approaches convergence [22].

Crossover operator takes two chromosomes and swaps part of their genetic information to produce new chromosomes. Although one-point crossover was inspired by biological processes, it has one major drawback in that certain combinations of schema cannot be combined in some situations [23]. A multipoint crossover can be introduced to overcome this problem. As a result, the performance of generating offspring is greatly improved. An example of this operation is depicted in where multiple crossover points are randomly selected. Another approach is the uniform crossover. This generates offspring from the parents based on a randomly generated crossover mask. The operation is demonstrated in the resultant offspring contains a mixture of genes from each parent. The number of effective crossing points is not fixed, but will be averaged at L/2 (where L is the chromosome length).

The preference of which crossover techniques to use is arguable. However, [24] concluded that a two-point crossover seemed to be an optimal number for multipoint crossover. This has since been contradicted by [loll as two-point crossover could perform poorly in a situation where the population has largely converged because of reduced crossover productivity. This low-crossover productivity problem can be resolved by the proposal of reduce-surrogate crossover [25]. Since the uniform crossover exchanges bits rather than segments, it can combine features regardless of their relative locations. This ability may outweigh the disadvantage of' destroying building blocks and make uniform crossover a superior operator for some problems [26]. reports on several experiments for various crossover operators. A general comment is that each of these crossovers is particularly useful for some classes of problems and quite poor for others, except that one-point crossover is indicated as a “loser” experimentally. Some other problem-based crossover techniques have been proposed. described a partially matched crossover (PMX) for the order-based problem. designed an “analogous crossover” for robotic trajectory generation. Therefore, the use of a crossover technique to improve the offspring production, is very much problem oriented. All in all, there is no unified view on this front [27].

Mutation is implemented by occasionally altering a random bit in a string before the off springs are inserted into the new population [7]. Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima. It is an operator that introduces diversity in the population whenever the population tends to become homogeneous due to repeated use of reproduction and crossover operators. Mutation may cause the chromosomes of individuals to be different from those of their parent individuals. Mutation in a way is the process of randomly disturbing genetic information. They operate at the bit level; when the bits are being copied from the current string to the new string, there is probability that each bit may become mutated. This probability is usually a quite small value, called as mutation probability pm. A coin toss mechanism is employed; if random number between zero and one is less than the mutation probability, then the bit is inverted, so that zero becomes one and one becomes zero. This helps in introducing a bit of diversity to the population by scattering the occasional points.

This random scattering would result in a better optima, or even modify a part of genetic code that will be beneficial in later operations. On the other hand, it might produce a weak individual that will never be selected for further operations. The need for mutation is to create a point in the neighbourhood of the current point, thereby achieving a local search around the current solution. The mutation is also used to maintain diversity in the population. It can be noticed that all four strings have a 0 in the left most bit position. If the true optimum solution requires 1 in that position, then neither reproduction nor crossover operator described above will be able to create 1 in that position. The inclusion of mutation introduces probability pm of turning 0 into 1. These three operators are simple and straightforward. The reproduction operator selects good strings and the crossover operator recombines good sub-strings from good strings together, hopefully, to create a better sub-string. The mutation operator alters a string locally expecting a better string. Even though none of these claims are guaranteed and/or tested while creating a string, it is expected that if bad strings 8 are created they will be eliminated by the reproduction operator in the next generation and if good strings are created, they will be increasingly emphasized. Further insight into these operators, different ways of implementations and some mathematical foundations of genetic algorithms can be obtained from GA literature. Application of these operators on the current population creates a new population. This new population is used to generate subsequent populations and so on, yielding solutions that are closer to the optimum solution. The values of the objective function of the individuals of the new population are again determined by decoding the strings. These values express the fitness of the solutions of the new generations. This completes one cycle of genetic algorithm called a generation. In each generation if the solution is improved, it is stored as the best solution [28].

Control parameters: We can visualize the functioning of GAs as a balanced combination of exploration of new regions in the search space and exploitation of already sampled regions. The balance, which critically controls the performance of GAs is determined by the right choice of control parameters: the crossover and mutation probabilities and population sizes. The trade-offs that arise are:

- Increasing the crossover probability increases the recombination of building blocks, but it also increases the disruption of good strings.
- Increasing the mutation probability tends to transform the genetic search into a random search, but it also helps reintroduce lost genetic material.
- Increasing the population size increases its diversity and reduces the probability that the GA will prematurely converge to a local optimum, but it also increases the time required for the population to converge to the optimal regions in the search space.

IV. EXPERIMENTAL RESULT

The experimental result following a research and optimization in the form of different data sets, we have treated a datasets in order to obtain the accuracy of effort estimation and more difficult interpretable than models of other techniques[11]. The other techniques are more exactly accurate result in compare the neural network and genetic algorithm.

The result show the accuracy of simplified model (BPNN) remains acceptable compared with optimizes neural network model. We define several input parameters to estimate the accuracy of the soft ware effort estimation of the project. The back propagation neural network (BPNN) and the genetic algorithm simulated using Matlab software. We defined the network of input parameters to calculate the performance of each network.

The two models of compared to the methods of determining the connection weight and bias transformation describes the average prediction the accuracy of each model for different types of datasets. the optimized accuracy of BPNN increase the accuracy for training and testing data sets. It is accurate to compare the prediction accuracy between the training and testing datasets holdout the data compared in other models. This result may predict the accuracy in the software effort estimation datasets.

V. PERFORMANCE CRITERION

There are several measurements used in the literature to evaluate the accuracy of prediction methods in software effort estimation tasks. In this paper we employ the two measurements that are most commonly used in the form of Error and Accuracy of MMRE and PRED (25%) [8].

The MMRE values calculated as follows;

$$MRE = \frac{|Actual\ Effort - Estimated\ Effort|}{Actual\ Effort} * 100$$

$$MMRE = \frac{\sum_{i=1}^{N} MRE}{N}$$

MRE=Mean Relative Error.

MMRE= Mean Magnitude of Relative Error.

PRED(25) is the percentage of predictions that fall within 25% of the actual value.

The PRED (25%) values calculated as follows;

$$PRE = MRE \leq .25$$

$$PRED\,(X) = \frac{A}{N}$$

PRE=Predictions.

PRED=Percentage of Predictions.

VI. RESULTS

For experimental analysis, we have chosen 7 random projects from NASA data set. It shows that the proposed model has MMRE less than NN and GA model as shown in Figure 3. The comparisons between the results is shown in table 1.

TABLE 1: COMPARISONS OF RESULTS

DATA SETS	**MMRE**	**PRED (25%)**
NASA	0.5128	80.6452
ALBERCHET	0.749	93.38
DESHARNAIS	0.1672	79.03
TELECOM	0.3668	46.1538
MAXWELL	0.5220	47.7273
HALLMARK	0.1103	88.88

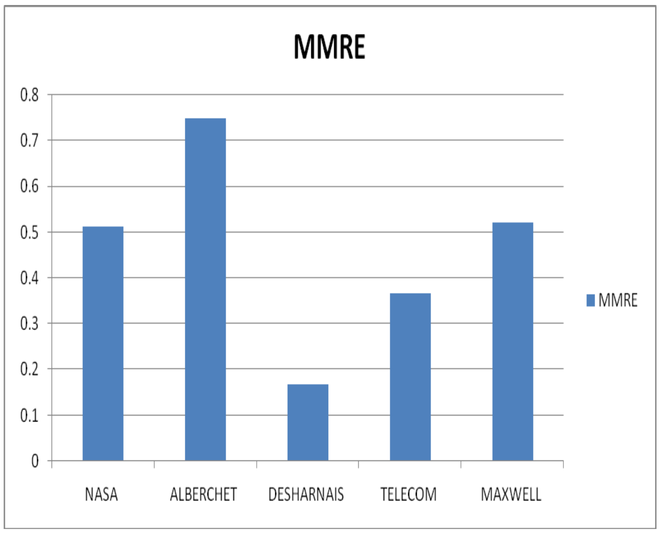

Fig. 3 Comparison of Different Models of MMRE

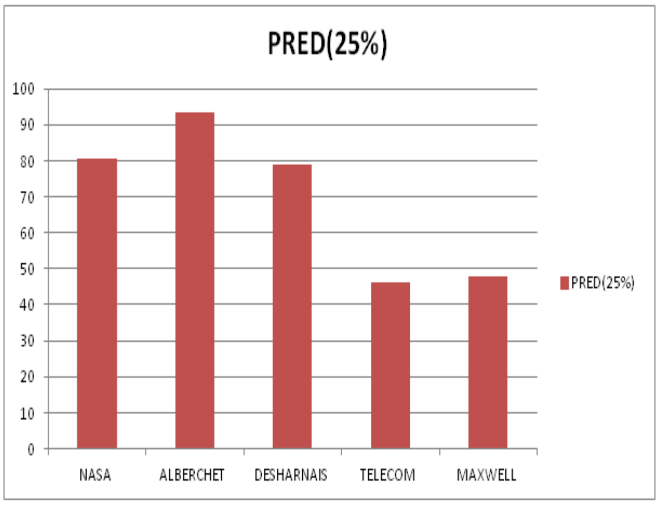

Fig. 4 Comparison of Different Models of PRED

VII. CONCLUSIONS AND FUTURE WORK

The software cost estimation to calculate the accuracy and error for the form of MMRE and PRED (25%) for the given datasets. The all dataset can be calculate the separate values of error measure in the form of MMRE and PRED (25%) for software cost estimation[9].

Accuracy error measures has been obtained successfully by neural network with genetic algorithm. The Genetic Algorithm and Back Propagation algorithm to calculate the Accuracy and Error in the Software Effort Estimation. It can be used to another technique for Radial Basis Function network and Particle Swarm Optimization for calculating Accuracy and Error in the Software Effort Estimations.

REFERENCES

[1] Salman Mohaghegi, Yamilledel Valle, Ganesh K.Venayagamoorthy, And Ronald G. Harley, “A comparison of pso and back propagation for training rbf neural networks for identification of a power system with stat com” IEEE Swarm intelligence symposium., pp. 381-384, Jun.2005.

[2] B.TirimulaRao, B.Sameet, G. KiranSwathi, K. Vikram Gupta, CH. Ravi Teja, S.Sumana, “A novel neural network approach for software cost estimation using functional link artificial neural network (FLANN),” International journal of computer science and network security., vol. 9, no. 6, pp. 126-131, June 2009.

[3] Ivica Kalichanin- Balich, Cuauhtemoc Lopez-Martin, “Applying A feed forward neural network for predicting software Development effort of short-scale projects,” ACIS International conference on software engineering

research, management and applications., pp. 269-275, 2010.

[4] Ali Idri, Taghi M. Khoshgoftaar, Alain Abran, "Can neural networks be easily interpreted in software costestimation," IEEE International conference., vol. 2, pp. 1162-1167, May 2002.

[5] Abbas Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," Information and Software Technology., vol. 44, pp. 911–922, 2002.

[6] CH.V.M.K.Hari, Tegjyot Singh Sethi,B.S.S.Kaushal, Abhishek Sharma, "CPN-A hybrid model for software cost estimation," IEEE Intelligent computational systems., pp. 902-906. Sep 2011.

[7] Venus Marza, Amin Seyyedi, and Luiz Fernando Capretz, "Estimating development time of software projects using a neuro fuzzy approach," World Academy of Science, Engineering and Technology., vol. 46, pp. 575-579, 2008.

[8] Adriano L.I. Oliveira, Petronio L. Braga, Ricardo M.F. Lima, Márcio L. Cornélio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," Information and Software Technology., vol. 52, pp. 1155-1166, 2010.

[9] Vachik S. Dave, KamleshDutta, "Neural Network based Software Effort Estimation & Evaluation criterion MMRE, "International Conference on Computer & Communication Technology., pp. 347-351, 2011.

[10] Petr Musilek, Witold Pedrycz, Nan Sun, Giancarlo Succi, "On the Sensitivity of COCOMO II Software Cost Estimation Model," IEEE Symposium on Software Metrics., 2002.

[11] T. Hasangholipour, and FaribaKhodayar, "A novel optimized neural network model for cost estimation using genetic algorithm," Journal of applied sciences., vol. 10, no. 6, pp. 512-516, 2010.

[12] Tron Foss, Erik Stensrud,Barbara Kitchenham, and Ingunn Myrtveit, "A simulation study of themodel evaluation criterion MMRE," IEEE transactions on software engineering., vol. 29, no. 11, pp. 985-995, Nov 2003.

[13] NicolangeloIannella, Andrew D. Back, "A spiking neural network architecture for nonlinear function approximation", Elsevier neural network., pp. 933-939, Apr 2001.

[14] L. Gerardo Colmenares, and Rafael Perez, "A data reduction method to train, test, and validate neural networks," IEEE proceedings southeastcon.. pp. 277-280, Apr 1998.

[15] Donald F. Specht, "A general neural network," IEEE transactions on neural networks., vol. 2,no. 6, pp. 568-576, Nov 1991.

[16] Putnam, L. H., "A General Empirical Solution to the Macro Software Sizing and stimating Problem", IEEE Transactions on Software Engineering, Vol. 4, No. 4, pp. 345 – 361, 1978.

[17] Srinivasan, K. and Fisher D., "Machine Learning Approaches to Estimating Software Development Effort", IEEE Transactions on Software Engineering, Vol.21, No. 2, 1995.

[18] Iman Attarzadeh and Siew Hock Ow, "A novel soft computing model to increase the accuracy of software development cost estimation," IEEE., pp. 603-607, 2010.

[19] Gerg Balogh, Ádám Zoltán Végh, and Árpád Beszédes, "Prediction of Software Development Modification Effort Enhanced by a Geneticalgorithm,"

[20] J. E. Baker, "Adaptive selection methods for genetic algorithms," in Proc. /st Int. Con$ Genetic Algorithms. J. J. Grefenstette, Ed., Lawrence rrlbaum Associates, pp. 101-111, 1985.

[21] D. Whitley, "The GENITOR algorithm and selection pressure: why rank- based allocation of reproductive trials is best," in Proc. 3rd Int. Conj; genetic Algorithms, J. D. Schaffer, Ed., pp. 116-121, 1989.

[22] Z. Michalewicz, Genetic Algorithms, Data Structures and Evolution Program, 2nd Ed. Berlin: Springer-Verlag, 1994.

[23] K. DeJong "The analysis and behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1975.

[24] L. Booker, "Improving search in genetic algorithms," in Genetic Algorithms and Stimulated Annealing, L. Davis, Ed. New York Pitman, pp. 61-73, 1987.

[25] G. Syswerda, "Uniform crossover in genetic algorithms," in Proc. 3rd Int. Conf Genetic Algorithms, , pp. 2-9, 1989.

[26] K. F. Man, Member, IEEE, K. S. Tang, and S. Kwong, Genetic Algorithms: Concepts and Applications," IEEE Transactions on industrial electronics, vol. 43, no. 5, october 1996.

[27] Tom V. Mathew, "Genetic Algorithm," Indian Institute of Technology

[28] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB8QFjAA&url=http%3A%2F%2Fstudent.bus.olemiss.edu%2Ffiles%2Fconlon%2Fothers%2Fothers%2FTemp%2FBus669_CompInfo%2FChapter4%2FGenetic%2520Algorithm.ppt&ei=FZpAVKqqC9KTuASxmYHADg&usg=AFQjCNFcIFIb83w9fTKsYlMuV8nUPEtUCA&bvm=bv.77648437,d.c2E&cad=rja

[29] http://www.roderik.net/education/cir/evolutionary-computing/

[30] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0CCgQFjAB&url=http%3A%2F%2Fwww.researchgate.net%2Fpublication%2F31594088_Fuzzy_logic_and_neural_nets_in_intelligent_systems%2Flinks%2F0912f510902e2baec6000000&ei=cZpAVPb9H5SDuwTOilHoBw&usg=AFQjCNEXn_0l7RtT2xfGRn9fL1t-1Q4ijw&bvm=bv.77648437,d.c2E

[31] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0CC8QFjAC&url=http%3A%2F%2Ftjmcs.com%2Fincludes%2Ffiles%2Farticles%2FVol4_Iss3_411%2520-%2520417_Artificial_Neural_Network_Based_Mod.pdf&ei=cZpAVPb9H5SDuwTOiIHoBw&usg=AFQjCNFIZ53gKCrLRgeKDJ17y2tv4YOt9g&bvm=bv.77648437,d.c2E

[32] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB8QFjAA&url=http%3A%2F%2Fwww.ijarcsse.com%2Fdocs%2Fpapers%2FVolume_3%2F5_May2013%2FV3I5-0383.pdf&ei=n5pAVJe9KZKhugSJr4KICA&usg=AFQjCNEVbpgcVZ0_04svP8vpNejoVPUOyg&bvm=bv.77648437,d.c2E

[33] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB8QFjAA&url=http%3A%2F%2Fwenku.baidu.com%2Fview%2F6f0a0df8f705cc17552709e7.html&ei=pppAVObqCNWMuAT3noJY&usg=AFQjCNGZoUFnDCKJS71jtYHtn_RMTNSEww&bvm=bv.77648437,d.c2E

[34] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB8QFjAA&url=http%3A%2F%2Fwww.civil.iitb.ac.in%2Ftvm%2F2701_dga%2F2701-ga-notes%2Fgadoc%2Fgadoc.html&ei=75pAVPq9H8-1uASWrIFw&usg=AFQjCNF6z14rA48bBt-mGvF2Zx3dD7LKFA&bvm=bv.77648437,d.c2E

[35] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0CCYQFjAB&url=http%3A%2F%2Fwww.academia.edu%2F4815323%2FPerformance_Analysis_of_GA_and_PSO_over_Economic_Load_Dispatch_Problem&ei=95pAVN2VBsGcugTaioLADw&usg=AFQjCNHlWOuqukkCB_s-KtTou5gR7vIpXQ&bvm=bv.77648437,d.c2E

[36] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB8QFjAA&url=http%3A%2F%2Fwww.inf.u-szeged.hu%2F~beszedes%2Fresearch%2Fcr.pdf&ei=_5pAVMn5H9CbuQTWjIHAAg&usg=AFQjCNG860hwl_Ee1jXuNPUm7pq3Tn5dGw&bvm=bv.77648437,d.c2E