International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-7, Issue-6, March 2019

Enhancing Software Reliability Prediction based on Hybrid Fuzzy k-Nearest Neighbor with Glowworm Swarm Optimization (FKNN-GSO) Algorithm

Shailee Lohmor, B. B. Sagar

Abstract: ***Predicting software reliability means gauging the future occurrences of failures in software in order to align the process of the software maintenance. This paper presents a model based on FKNN (Fuzzy k-Nearest Neighbor) and nature inspired Glowworm swarm optimization (GSO) to understand the relationship between the data of software failure time and the nearest n failure time and finally predict the reliability of the software. Glowworm-Swarm Optimization (GSO) is used to search finest combination of weights aimed to obtain maximum regression accuracy and fuzzy k-nearest neighbor (FKNN) to allocate the degree of membership to various software metrics using fuzzy logic concepts. The performance of the proposed model has been compared with the known existing models to evaluate the prediction efficiency of GSO- FKNN.***

Index Terms: Fuzzy Membership function, Glowworm swarm Optimization (GSO), K-Nearest Neighbor (KNN), Software Reliability Prediction (SRE), MAE (Mean-Absolute Error), MSE (Mean-Squared Error)

I. INTRODUCTION

Now a days, we are surrounded by different types of software's who directly or indirectly lays impact on our everyday tasks and decisions. However, with the development of software industry, tremendous growth in the size and costing of the software, large amount of efforts and time in development of software was noted [22]. Since human are unequivocally reliant on the software system in everyday life, any issue related to software or system failures results in diminishing customer satisfaction especially in safety critical system. To avoid such situation, considering the process of forecasting software failures during the development phase was proposed [21]. During research it was observed that majority of faults are found in common modules and deliberate efforts on fault removal from common modules will be beneficial [16]. Later on it was noted that the focused study of the failure dataset can work as a platform to build software reliability prediction model applicable on forthcoming software projects [6]. Software reliability modeling is important since the software is utilized in varied area of numerous applications. To produce reliable, efficient and flexible software products at the resulting stage, it plays a major role in providing path for establishing, handling and preserving the software was discovered [3]. Past data research have demonstrated that the effect of software defects is experienced globally which influence us both financially and on a human side too was noted by P. K. Kapur [12]. Therefore, software reliability prediction plays vital role and has become a major research area in software engineering. S. Chatterjee [18] noted that whenever a functional unit fails to perform its defines function it is termed as failure and a methodology to analyze these failures are termed as software reliability model. The most essential task in software reliability prediction is the prediction of parameter. There are two types of techniques involved in the prediction of parameters namely, LSE (Least Square Error) and MLE (Maximum Likelihood Estimation) were explained [2]. Software Reliability Growth Models (SRGMs) are used in combination with removed faults in order to measure the reliability of the software more efficiently. The parameters which are appropriate to the model are calculated by both least square and first-order differential techniques [24]. In earlier works, recurrent architecture and ensemble model which is based on the concepts of Genetic-Programming (GP) and Group Method of Data Handling (GMDH) has participated in the accurate prediction of software was discovered by Ramakant [17]. Reliability of the software was predicted based on multi-layer Feed Forward Artificial Neural Network (FF-ANN) namely Logistic Growth Curve Model. The FF-ANN was carried out through the presence of network of hidden neurons. By taking advantage of the network's weights, a neuro-genetic approach is also presented for reliability prediction [14]. The approaches used earlier for predicting the software reliability are mostly based on statistics which to a large extent resulted in unsatisfactory performance in terms of prediction. Thus, all the recent research introduced machine learning techniques like Data mining, Naïve bayes algorithm, Support vector machine (SVM), and ANN etc. was noted by Hiroyuki Okamura [7]. In spite of the fact that software faults have been contemplated utilizing these techniques, there are still numerous parts of flaws staying hazy.

Revised Manuscript Received on 30 March 2019.
* Correspondence Author
Shailee Lohmor*, Research Scholar, Research & Development Centre, Bharathiar University, Coimbatore, India.
Dr B B Sagar, Birla Institute of Technology, Mesra-Ranchi , India.

For predicting the efficiency of the software accurately, a support vector regression technique is applied. The parameters of SVR should be selected carefully for calculating the reliability more exactly. The defect datasets are partitioned into training and test subsets during the selection of parameters. However the calculation of SVR parameters is a critical task. Although several approaches have been exploited, they contain some drawbacks were noted by WeiZhao [23] and Cong Jin, [4].

II. LITERATURE REVIEW

The Component-based Software systems (CBSSs) introduced by Kirti Tyagi and Arun Sharma [10] is a Rule-based approach which faced difficulty in estimating the reliability exactly because it was a real-world event. To overcome this, two types of basic soft computing methods was utilized here namely, fuzzy computing and probabilistic computing. These methods were introduced based on four factors. The first factor was the reusability. This was defined as the usage of a component again and again in various applications. The second factor was the operational profile. Here an entire set of activities were contained and also the occurrence for the expectation was included. The third one was the dependability of the component. Here the output of one component was used as the input for the next. The last one was the complexity. It was given by the amount of components and the inter connection between them. The proposed fuzzy computing contains three phases. In the first phase, the classification tables were converted into continuous classifications. Then at the second phase, an inference engine was utilized to handle those classifications. At last, these fuzzy numbers were converted into single real valued numbers in the final phase.

Ahmet Okutan and Olcay Taner Yildiz [1] introduced concept of Bayesian networks to predict defects in the software. In order to calculate the defects in software, they utilized two metrics. Firstly for the Number of Developers-NOD (Number of Children) and secondly for the Source Code Quality-LOCQ (Lack of Coding Quality) was utilized. To identify the incompleteness, LOCQ was used. In NOD, the knowledge about the developer was contained. Using this knowledge, the interconnection between the number of developers and the length of errors which can be affected easily can be identified. The connection between the metrics and the proneness of faults were decided by the Bayesian networks. The expectation of errors to occur at the border was also defined by the Bayesian networks. In Bayesian network, a graph was contained with a combination of vertices (V) and edges (E). Here a metric was denoted by V and the interconnection between the two metrics and defectiveness was denoted by E. In the event that an edge (E) was available from metric m1 towards defectiveness then this would imply that defectiveness was powerful on metric m1. Correspondingly, if an edge (E) was available from metric m2 towards m1 then this would imply that m1 was powerful on metric m2 and so on.

A Time dependent fault detection and fault removal model was designed by Mengmeng Zhu and Hoang Pham [11] for predicting software reliability. The estimation of dependent faults, removal of imperfect defects and the total number of defects, a software reliability model known as NHPP (nonhomogeneous poisson process) was presented. The removal of faults, detection of faults etc. were done based on the programmer's ability, types of faults and programming complexity. The model parameters can be calculated by applying the genetic algorithm (GA). In this approach faults were depended on the faults identified in the previous stage and can be eliminated during detection phase. But in some cases faults were not visible at the testing phase instead it will be visible only at the operation phase. A Fuzzy Analytic Hierarchy based approach for Structure-based software reliability allocation was introduced by Subhashis Chatterjee et.al [19]. The reliability of software was estimated during the design phase of the software. The view point of the user was integrated and a model for software reliability allocation was introduced. In this model, the success of reliability can be guaranteed. For allocation of reliability, in this method, the communication among users, software engineers and programmers was enhanced. With the help of our proposed model, the reliability can be estimated at the planning and design phase itself. To split the problem from higher level to lower level, a hierarchical structure was developed. The entire process was used to determine the parameters of the Hierarchy. A Multi-Objective Optimization based approach proposed by Dr. M. Sangeetha et.al [5] aimed at Software reliability allocation for improving quality considered factors like reliability, cost and schedule were taken into account and an effective software reliability allocation method was developed. Here the reliability of the software was increased by various software reliability models, reliability evaluation, debugger employment, internal quality factors and the estimation of computation results. During the allocation of reliability of the software, a multi objective optimization technique was introduced which was supported by both the effectiveness of resources and schedule planning. In this proposed optimization technique, reliability of the software was increased and the factors like cost and schedule were decreased. Software architecture, data of software failure and project information were the three inputs utilized. The information about the number of modules contained in the software was given in the software architecture. The requirements like reliability goal, budget and schedule were included in the project information. The parameters of the software reliability models can be calculated by the failure data.

III. SOFTWARE RELIABILITY MODEL BASED ON GSO OPTIMIZED FUZZY KNN FOR RELIABILITY PREDICTION

Software-Reliability is defined as the process of sudden occurrence of successful executions of an input state which was selected from the input space under specified operating conditions.

To construct and preserve the quality of software, the process of predicting the software reliability plays a major role. Its main aim is to eliminate the number of failures caused in a system during development stage because these failures will reduce the performance. The estimation of reliability of software is a critical step because the accuracy varies from one application to another and also there will be increased cost. To overcome this type of problems, several types of Software-Reliability-Growth-Models (SRGMs) have been utilized for predicting the reliability of the software. The two common types of SRGMs are the parametric and the non-parametric model. In parametric model, the nonhomogeneous poisson process (NHPP) is utilized while in non-parametric model various machine learning and soft computing techniques are utilized.

The statistical independence between consecutive software failures is considered as assumption in majority of the currently available models for predicting software. However, there is a question mark till now on predicting reliability of software and addressing this problem will be considered as motivation. Hence this research work plans to develop an efficient model for predicting software reliability aimed at the determination of reliability of software.

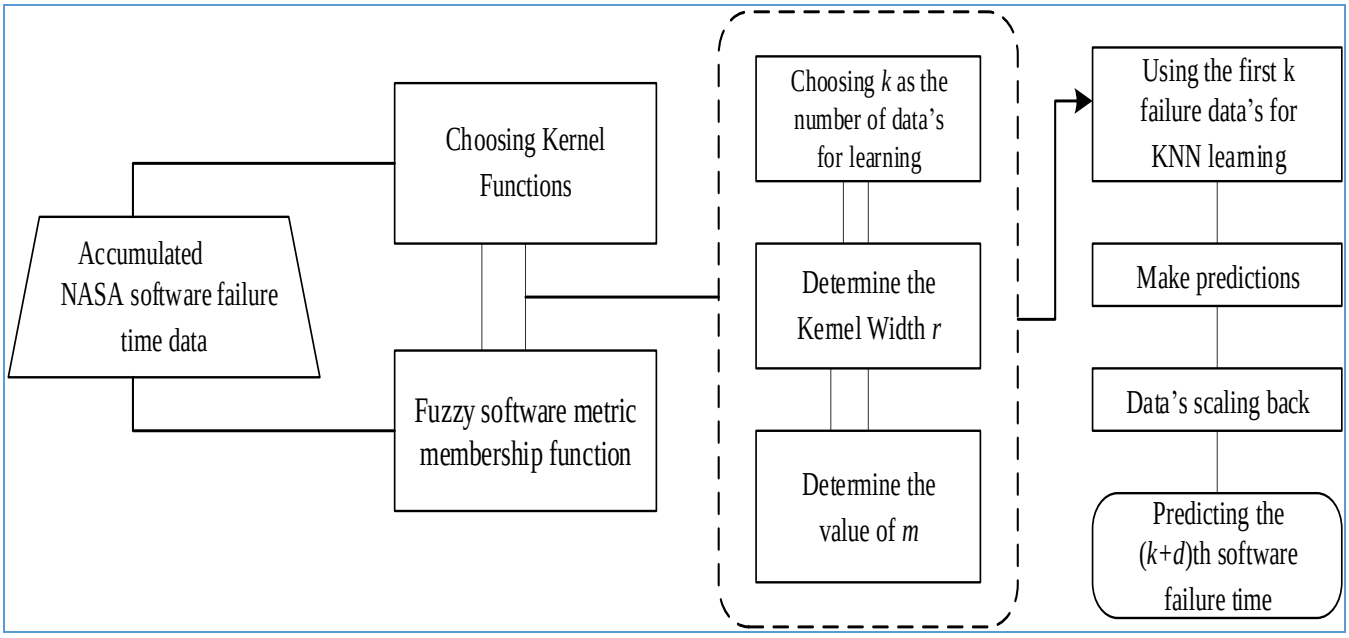


Fig. 1. Software Reliability Model process based on KNN

Software reliability prediction is core task among the processes of software development. Constructing powerful reliability prediction model is a key element to make good reliable software products. Recently, Fuzzy control systems were largely used to solve nonlinear prediction problems in many fields. Existing methods of predicting software reliability suffer from poor performance for accurate failure estimation. In this research work an efficient model for software reliability prediction has been proposed on an adaptive fuzzy k-nearest neighbor (FKNN), where we allocate the degree of membership to various software metrics using fuzzy logic concepts. Assuming the distance of its k-nearest neighbors, the fuzzy strength parameter 'k' and the neighborhood size 'm' are adaptively specified by the Quantum Glowworm Swarm Optimization (QGSO) algorithm. In our proposed model to calculate the reliability 'R' for the software detect data set we used the number of failures, number of test cases and time interval. In existing approach of reliability prediction, reliability 'R' is calculated only by using number of failures and time interval.

To achieve parameter optimization for FKNN, the task of GSO algorithm is to determine the very much discriminative features for reliability prediction are subset from software failure dataset. Further to control the ability of GSO algorithm in the local and global search an adaptive Control Parameter is used including acceleration coefficients and inertia weight.

The creation of population of worms is randomly created by GSO algorithm in which each worm is a candidate subset of features. Each candidate is encoded with its Luciferin value and its fitness is assessed using FKNN. In addition, the proposed model has reduced the computational time by a large margin owing to its parallel implementation. The proposed software reliability prediction model, named GSO-FKNN has been implemented on Matlab working platform and the experimental results compared with existing techniques.

A. Reliability Prediction of SRGMs

SRGM can be utilized to comprehend the idea of how and why software fails and what are the methodologies ought to be utilized to measure the software reliability. This software failure dataset is used to conclude the readiness of the software to finally launch. The approach is aimed to capture the system's test data in order to forecast the number of failures.

The SRGM used during the

fault removal process are discrete time model and continuous time model. The former is based on the number of test cases whereas the latter uses the CPU time as the basic parameter. The basic models that are based on the software fault data are as follows:

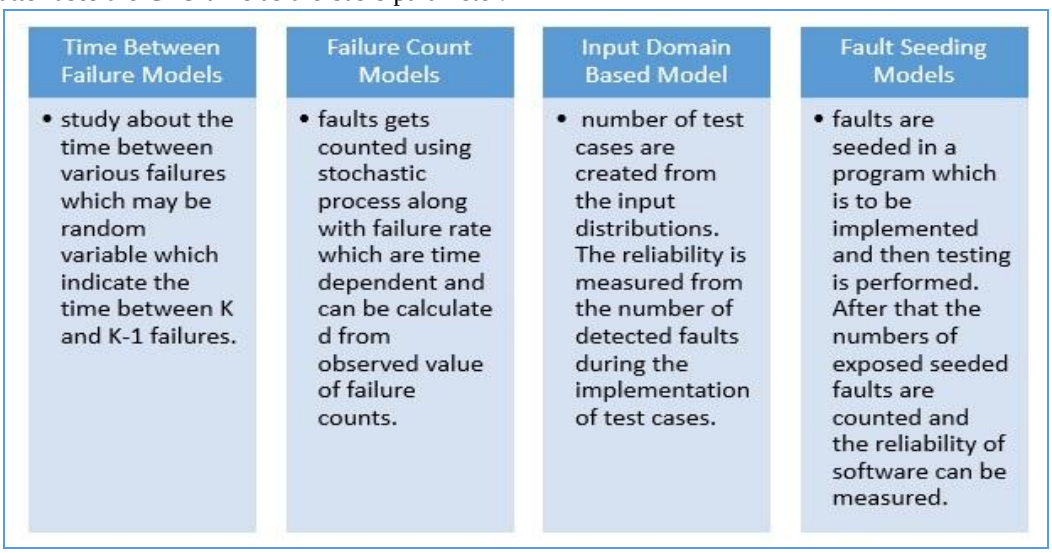


Fig. 2. Types of SRGM

A. Determination of Software Metrics

Majority of software-metrics like the object-oriented metrics or the process-oriented metrics have been used to predict software defect because as per the recent studies object-oriented metrics and process-oriented metrics are efficient in analysing and predicting faults as compared to traditional size-complexity metrics. In the recent researches the Fuzzy inference systems (FIS) are used to make decision making simpler so that the real time problem can be directed to make a decision and act accordingly. The FIS are aimed to reinforce the process of human reasoning by means of Fuzzy logic (If-Then rules). The proposed model incorporates Fuzzy inference system (FIS) to build development of membership function. The dataset for the study is from PROMISE Software Engineering Repository KC2 data set which contains 21 software metrics whereas on the basis of earlier research it was noted that only 13 software measurements holds major role in fault prediction. The method used for creation of fuzzy sets for static code software metrics is depicted in the figure ---

Let Y represent a set of M training patterns $(S_1, S_2, \ldots\ldots, S_j, \ldots\ldots, S_n)$ and S_j represent a feature. The feature S_j have n values $p_{1j}, p_{2j}, \ldots\ldots\ldots, p_{nj}$. $S_{j\min}$ And $S_{j\max}$ represent the minimum and maximum value of feature S_j. When the input feature is quantitative value, then followings steps are done for membership function generation.

Step 1. Sort the values of feature S_j in ascending order.

Step 2. Perform K –means clustering algorithm for clustering the quantitative values of the feature S_j into k clusters $x_1, x_2, \ldots\ldots, x_i, \ldots\ldots, x_k$ Where, $x_{i\min}$ and $x_{i\max}$ denote the minimum and maximum value of i^{th} cluster (x_i)

Step 3. Determine the centres of the cluster $(c_1, c_2, \ldots\ldots, c_i, \ldots\ldots, c_k)$ of k clusters $(x_1, x_2, \ldots\ldots, x_i, \ldots\ldots, x_k)$.

Step 4. Aims to calculate the membership value of two boundary points every cluster.

Substep 4.1. Calculate the difference between adjacent data. For each pair v_i and $v_{i+1} = (i = 1,2,\ldots\ldots, n-1)$ the difference is $diff_i = v_{i+1} - v_i$.

Substep 4.2. The similarity value between adjacent data of quantitative values of feature S_j needs to be determined. The formula mentioned below is used to find how similar the adjacent data is.

$$s_m = \begin{cases} 1 - \dfrac{diff_i}{C * \sigma_i} & for\ diff_i \le C * \sigma_i \\ 0 & otherwise \end{cases}$$

Where, s_m represents the similarity component between adjacent data, σ_i represents Standard derivation of $diff_i$ and C refers to Control Parameter that decides the shape of membership functions.

Fig. 3. Creation of Fuzzy Sets

A. k-NN for regression

K-Nearest Neighbor (k-NN) is a Machine-Learning (ML) algorithm used to group the input data on the basis of k nearest neighbors. The algorithm is very simple, effective and non-parametric. K-NN categorizes its data during the testing phase rather than training phase thus is called as lazy learning algorithm. K-NN being a lazy learning method makes its adaptable to the changes however on the other hand it takes larger computation time during testing. K-NN is used for regression and classification.

When K Nearest Neighbour (K-NN) is used for regression problems then the prediction is based on the average of the K-most similar instances. Thus the choice of K is important in building the K-NN model. To obtain estimates of the model parameters that are unknown we use cross validation technique. In order to work on regression we use k-NN to evaluate x_t (testing point) which means weighted average of the closest training points. A kernel function is incorporated to evaluate the closeness of testing point and finally calculate weight of each neighbour. Consider the training dataset be $X = \{x_1, x_2, ..., x_M\}$ which has M training points and all these holds N features. The weighted Euclidean distance is used to capture the closeness of training point with the testing point where number of features is denoted as N, weight w_n which ranges between $0 \le w_n \le 1$ is assigned to nth feature. The formula is expressed as

$$d(X_t, X_i) = \sqrt{\sum_{n=1}^{N} w_n (x_{t,n} - x_{i,n})^2} \quad (1)$$

Summarising we can say that entire approach reflects the weight of the feature which signifies the importance of the feature.

In the proposed approach Glowworm-Swarm Optimization (GSO) is used to search finest combination of weights aimed to obtain maximum regression accuracy. Further, as a kernel function we have used the Gaussian Radial Basis Function (RBF) that decays as an exponential function of the difference between two data points (for e.g., the weighted Euclidean distance in our case). The Gaussian Radial Basis Function (RBF) is expressed as

$$\phi\left(x_t, x_{(i)}\right) = e^{-d\left(x_t, x_{(i)}\right)/\beta} \quad (2)$$

Where β refers to Gaussian Decay Factor and is set to half of the mean distance between training points and their k-nearest neighbors [18].

B. Feature Weighting with GSO

With the aim to maximize the regression accuracy, an optimal feature weight vector is determined referred as Feature Weighting. Root Mean Square Error (used as Error measure) indicates the optimal combination of weights. The RMSE is evaluated using k-Fold Cross Validation (CV) and the steps are depicted in the figure 4.

Considering that a training data set X consists of L mutually exclusive subsets or folds. During each Cross Validation (performed L times) trial, the test set is of L subsets and the training set has $L-1$ subsets.

Step1: Let $I_l = \{i : x_i \in X_l\}, l = 1,2,....,L$, denote the index set of the data points that construct the subset X_l.

Step 2: The Cross Validated -RMSE is calculated as the root square of the average error over all the L CV trials,

Expressed as

$$\varepsilon_{CV} = \sqrt{\frac{1}{U}\sum_{l=1}^{L}\sum_{i \in I_l}\left(\hat{f}_{w, X \setminus X_l}(X_i) - f(X_i)\right)^2}$$

Where U are data points for Cross Validation, weight vector $W = (w_1, w_2, ..., w_N)$, k-NN regression model $\hat{f}_{w, X \setminus X_l}$ built with the w (weight vector) and X (training dataset) excluding the subset X_l, and $f(x_i)$ the true response of x_i.

Fig. 4. RMSE evaluation using k-fold cross validation

C. Optimization of the Weight

To differentiate feature selection from the feature weighting, the problem of optimization of the weights is framed as Minimize $\varepsilon_{CV} = \varepsilon_{CV}(w)$,subject to $0 \le w_n \le 1, n = 1,2,....,N$ (3)

In addressing the problem of weight optimization the proposed approach incorporates Glowworm-swarm optimization (GSO) algorithm [25] as depicted by Eq. (4). The agents i in the algorithm are glowworms that carry a luminescence quantity called luciferin $(l_i(t))$ along with them. These agents are randomly deployed initially in the search space. Using the objective function the fitness of the glowworms is evaluated and broadcasted to the neighbors. The agents identify each other and move using a probabilistic mechanism and comparing luciferin values. The algorithm follow three phases:

Phase 1: Initially the luciferin value of all glowworms remain same. Luciferin is updated on the basis of the function values of their positions where little value is deducted from luciferin as the decay with time.

The luciferin update rule can be written as:

$$l_i(t) = (1-\rho) l_i(t-1) + \gamma \times f\left(x_i(t)\right) \quad (4)$$

Where

q is the luciferin decay constant $(0 < \rho < 1)$,

γ is the luciferin enhancement constant and

$f\left(x_i(t)\right)$ represents the value of the objective function at

agent i's location at iteration t. Phase 2: In phase 2, the glowworms, using the probabilistic mechanism, move towards the brighter glowworms.

The probability of a glowworm i moving towards a neighbor j is given by:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \quad (5)$$

Where $N_i(t)$ behaves agent i's neighbor muster during iteration t. $N_i(t)$ is shown as follows:

where $x_j(t)$ is agent j location calculated at iteration t, the Euclidian distance between glowworms i and j is represented by $\| x_j(t) - x_i(t) \|$ at iteration t, $l_j(t)$ is a measure of the luciferin held by glowworm j at iteration t, $r_d^i(t)$ is the changeable local-decision range associated with glowworm i at iteration t.

$$x_j(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\| x_j(t) - x_i(t) \|} \right) \quad (6)$$

Phase 3: On the basis of the local information, glowworms takes decision of their movements. The Radical sensor range determines the number of peaks and acts as s strong function to calculate the local decision range.

In extreme cases, we may need to find multiple peaks where the sensor range is made a varying parameter, and each agent i is associated with a local-decision domain whose radial range is dynamic in nature. The local-decision domain update rule can be represented as follows:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_i^d(t) + \beta(\eta_t - |N_i(t)|)\}\} \quad (7)$$

Where $r_d^i(t)$ is the decision radius of agent i at iteration t and satisfies $0 < r_d^i < r_s$,

r_s is the sensor radial range,

b is change rate of neighbor-domain and

n_t is threshold for numbers of glowworms within decision range.

D. Forecasting k-NN Formulation

If that the total number of observed failures are n and the execution time is $t_i, i = 1,2,.....,n$, then the general software reliability prediction model can be represented as follows:

$$t_l = (t_{l-m}, t_{l-m+1}, ..., t_{l-1}), \quad (8)$$

Where $t_{l-m}, t_{l-m+1}, ..., t_{l-1}$ is a vector of lagged variables, and the dimensions of the input vector are represented by m or the number of past observations related to the future value.

The KNN regression approach tries to find out the proper representation of software failure time data. The approximating function f plays the vital role in solving the problems of prediction. It provides the auto-correlation between the data and produces estimates. Therefore, to predict reliability of the software the KNN is trained first to understand the relationship between historical reliability metrics and then the failures are predicted.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

For evaluating the performance of the proposed parameter estimation approach for SRGMs, MATLAB 2013 simulation tool is used. In this paper, six classic Software Reliability Models were chosen to calculate prediction accuracy. Out of which four models were compared with other models. The Failure Data have been taken from NASA public software defect datasets set from the promise software engineering repository. The NASA public repository datasets such as CM1, JM1, KC1, KC 2 as data 1,2,3 and 4 respectively is utilized with up to 100 test cases comprising 500 data. All the evaluations done for proposed fuzzy k-nearest neighbor with GSO Algorithm is given based on this dataset.

A. *Performance Evaluation*

For performance evaluation of our proposed work, the models depicted in figure no 5 are used and the processing is made using GSO optimization algorithm.

Type	SRGM	μ(t)	Parameters
Finite failure Poisson	G-O model	$\mu(t) = a(1 - e^{-bt})$	M1
Finite failure Poisson	Delay S shaped model	$\mu(t) = a(1 - (1 + bt)e^{-bt})$	M2
Finite failure binomial	Weibull model	$\mu(t) = a(1 - (e^{-\beta t})a)$	M3
Infinite failure Poisson	M-O model	$\mu(t) = a\ ln\ (1 + bt)$	M4
Binomial	JM model	$\mu(t) = N(1 - \exp(-\phi t))$	M5
Infinite failure	Dunane model	$(t) = \alpha t\beta$	M6

Fig 5. SRGM Models

In this M1, M2, M4, M5, M6 has two parameters and M3 has three parameters respectively. With our approach, three models are accomplished to perform comparison with the existing approaches.

TABLE 1: PARAMETER ESTIMATION ACCURACY OF SRGM MODELS

NASA data	G-O	Delay S shape	Wei-bull	M-O	JM O	Dun-ane
CM1	75.72	73.43	34. 21	39.66	50.32	73.20
JM1	61.33	55.67	23.43	28.03	37.87	56.55
KC1	72.45	59.70	37.90	37.33	53.65	50.43
KC2	73.54	56.13	39.56	37.98	42.10	59.32

As per the result of experiment evaluation, the six SRMs parameter estimation accuracy were determined. All the Models are satisfied with characteristics of SRGM, In general, our experiment gives the better results for all the four models, using novel FKNN-GSO approach.

Thus the above comparisons proved the efficiency of the proposed algorithm with existing results.

B. *Comparison Criteria*

There are several approaches to evaluate the fitting between calculated values of SRGMs and a real data set. In this paper we have used MAE (Mean-Absolute Error), MSE (Mean-Squared Error):

The mean squared error (MSE) is the average value of the squares of the difference between the estimated value and the observed number of software errors:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(m_i - m(t_i))^2 \quad (9)$$

The mean absolute error (MAE) is the average value of the absolute errors:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}(m_i - m(t_i))$$

Where m_i is the observed real failures, and $m(t_i)$ is the estimated failures using a SRGM.

The estimation of MSE for the proposed fuzzy k-nearest neighbor with Glowworm swarm optimization (GSO) Algorithm is done by utilization of NASA public repository datasets with up to 100 test cases comprising of 700 data and is compared with existing FNN as in figure 6. This comparison with ANN and FNN proved the efficiency of proposed fuzzy k-nearest neighbor with GSO algorithm. Compare to the ANN approach, MSE values of proposed approach are smaller than the results of ANN for all four dataset

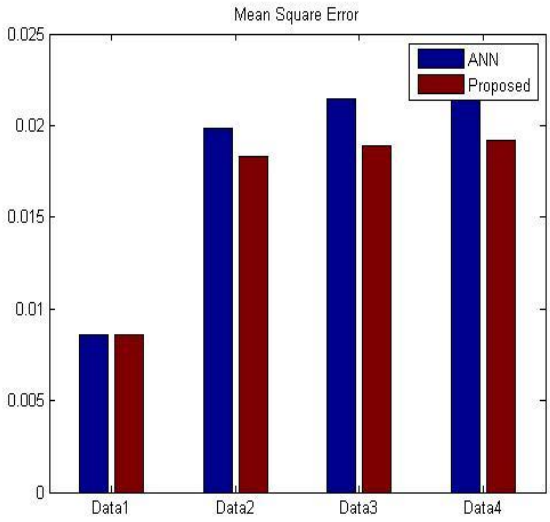


Fig. 6. MSE comparison of NASA datasets

Then simultaneously Mean Absolute Error (MAE) parameter estimation comparison is done for proposed fuzzy k-nearest neighbor with GSO algorithm with existing ANN is illustrated in figure 7.

Compare to the ANN approach, all of the minimum MAE values proposed approach are smaller than the results of ANN for three dataset except dataset3 where the higher values are noted.

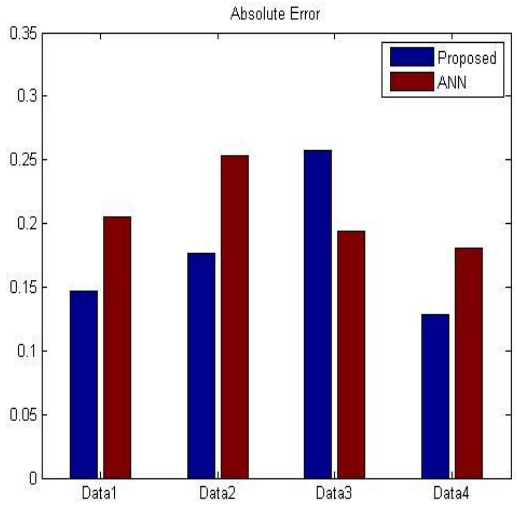
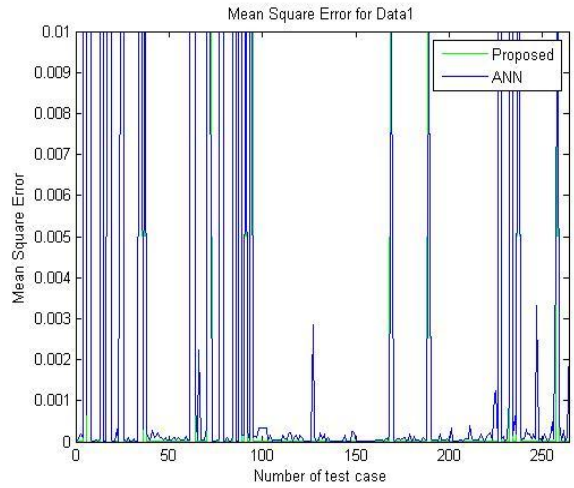
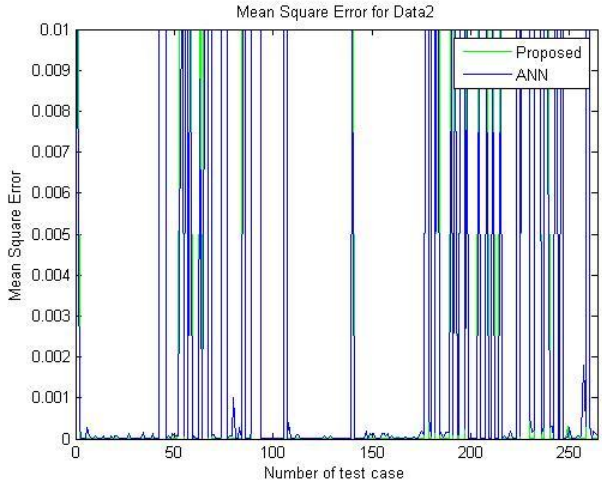
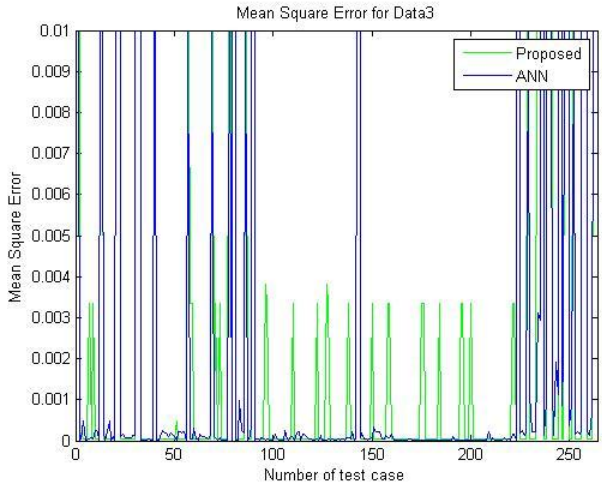


Fig. 7. MAE comparison of NASA datasets

These comparisons proved the efficiency of the proposed algorithm with existing results.







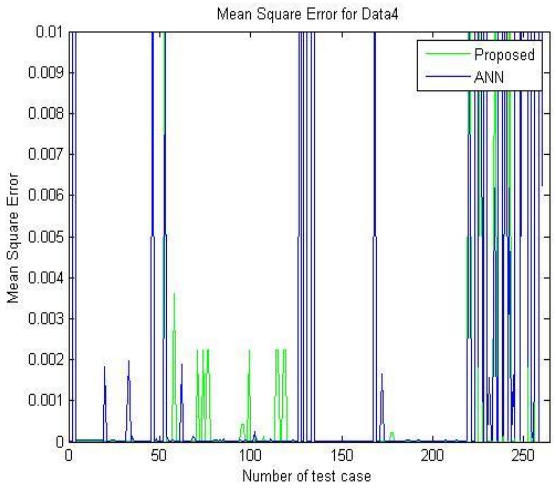
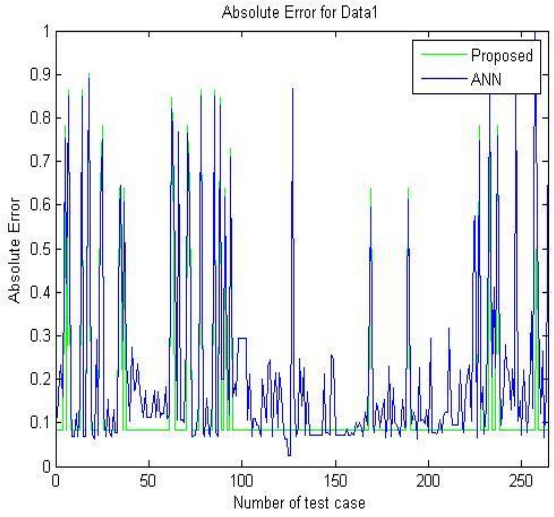


Fig. 8. Mean Square Error (MSE) comparison for proposed dataset with ANN and proposed FKNN-GSO approach

Then simultaneously Mean Absolute Error (MSE) parameter estimation comparison is done for proposed fuzzy k-nearest neighbor with GSO algorithm with existing FNN and is estimated for up to 100 test cases with 500 data as illustrated in figure 9.

Also MSE comparison for existing ANN is given for proposed fuzzy k-nearest neighbor with GSO algorithm and is estimated for up to 100 test cases with 700 data and is illustrated in figure 8. These comparisons proved the efficiency of the proposed algorithm with existing results.



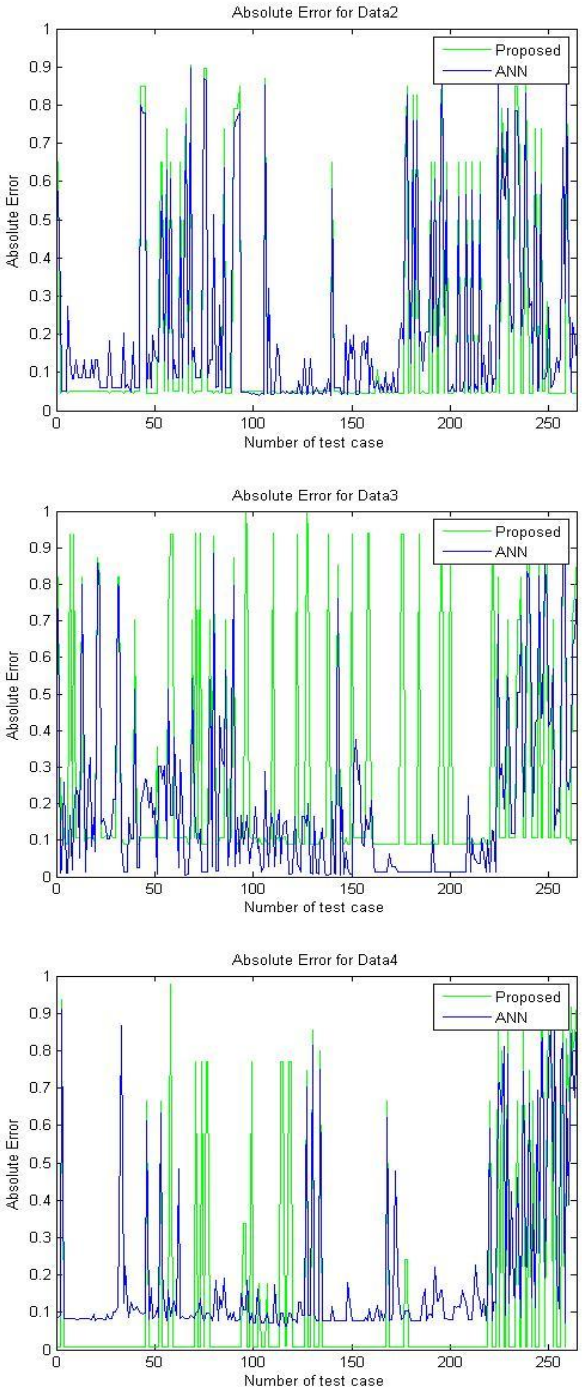


Fig. 9. Mean Absolute Error(MAE) comparison for proposed dataset with ANN and proposed FKNN-GSO approach

V. MODEL COMPARISON

TABLE II. **COMPARATIVE ANALYSIS OF AVERAGE RELATIVE PREDICTION ERROR FOR DATA SETS 1,2,3,4**

Datasets	**SVM**	**FFNN**	**RNN**	RVM	Proposed
Dataset1	2.44	2.58	2.05	2.50	1.11
Dataset2	1.52	3.32	2.97	5.23	2.04
Dataset3	1.24	2.38	3.64	6.26	1.71
Dataset4	1.20	1.51	2.28	4.76	0.87
Average	1.60	2.45	2.74	4.69	1.43

The results derived from modelling the co-relation between the software failure time sequences on the dataset 1,2,3,4 using our proposed FKNN-GSO approach is depicted in the table 2. To maintain the same pattern for the comparative analysis the datasets used to check the correlation is same as cited in [22], [23] and [24]. Park et al. implemented FF-NN and incorporated failure arrangement numbers as info and aggregate time to failure as wanted. The proposed FKNN-GSO approach has given average prediction error lowest as 1.43 as compared to the average prediction error got by as 1.60 utilizing SVM (Support Vector Machine), 2.45 when utilizing feed-forward neural system, 2.74 based on recurrent neural system, and 4.69 when utilizing feed-forward neural network. Thus the proposed FKNN-GSO approach gives much less mean prediction error as compared to other approaches in all the four datasets.

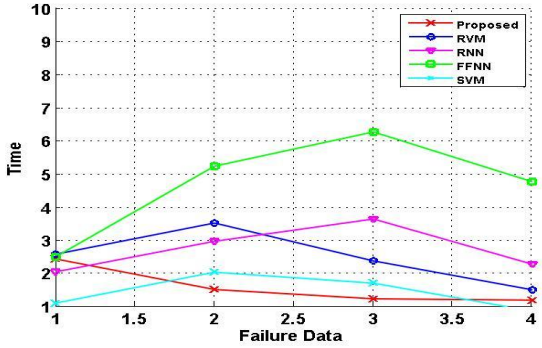


Fig. 10. Comparison with Existing Model

Figure 10 shows the parameter estimation accuracy of six software reliability growth models under NASA public software defect datasets. The performance evaluation given based upon the existing four SRGM models such as SVM, FFNN, RNN and RVM (Relevance Vector Machine) proved the efficiency of our proposed FKNN method as listed in table 2. The performance evaluation for all the four SRGM models are done with the same NASA datasets. Thus with aid of the comparison with currently available methods the efficiency of proposed method is analysed and is stated good for better detecting.

VI. CONCLUSIONS

The quality of software systems is improved with the aid of software defect prediction mostly done by software defect classification and software defect ranking. In this work, a GSO optimized FKNN-based model for software reliability prediction model is introduced. Here the reliability of software is predicted with a FKNN-based model by handling the system failure rate. The predictive accuracy is optimized with the Glowworm-Swarm optimization algorithm aimed to search finest combination of weights to obtain maximum regression accuracy and incorporation of adaptive fuzzy k-nearest neighbor (FKNN) to allocate the degree of membership to various software metrics using fuzzy logic concepts. Models using Euclidean distance metrics ensures the most optimal and reliable prediction results.

The study provides an attractive model for software reliability prediction and hence this technique ensures accurate predictions and reliability of too many real time applications such as health monitoring systems, safety critical systems etc., in Intelligent Environment systems. Although in future further investigation may be aimed at paying attention towards evaluating efficiency of the proposed model on larger datasets. In future, Evolutionary algorithms such as differential evolution, GAs, simulated annealing, chaotic GAs, and PSO can be connected to acquire more proper parameters for kernel functions, and thus accomplish more precise predictive capability in context of software reliability.

REFERENCES

1. Ahmet Okutan and Olcav Yildiz. “Programming defect prediction utilizing Bayesian systems”. Observational software engineering, Springer, volume 19, no 1, pp 154-81, 2014.
2. Chao Jung Hsu, Chin-Yu Huang and Jun-Ru Chang. “Upgrading Software Reliability Modeling and Prediction through the introduction time-variable fault reduction factor”. Connected mathematical modeling, Elsevier, volume 35, no 1, pp 506-21, 2011
3. Chin-Yu Huang and Michael R. Lyu. “Estimation and Analysis of some generalized change-point software reliability models”. IEEE Transactions on Reliability, volume 60, no 2, pp 498-514, 2011
4. Cong Jin, Shu-Wei Jin. “Programming reliability prediction show in view of help vector relapse with enhanced estimation of dispersion calculations”,Applied soft computing, Elsevier, volume 15, pp 113-20, 2014.
5. Dr. M Sangeetha, Dr C Arumugam, Dr K M Senthil Kumar, S Hari Shankar, “An effective approach to help multi target improvement in software reliability assignment for enhancing quality”, Procedia computer science, Elsevier, volume 47, pp 118-27, 2015
6. Hai Hu, Chang-Hai Jiang, Kai-Yuan Cai, W Eric Wong and Aditya O Mathur, “Improving programming unwavering quality appraisals utilizing changed versatile testing”. Information and software technology, Elsevier, volume 55, no 2, pp 288-300, 2013
7. Hiroyuki Okamura, Tadashi Dohi and Shunji Osaki, “Programming unwavering quality development models with typical disappointment time conveyances”. Reliability engineering and system safety, Elsevier, volume 116, pp 135-41, 2013
8. Hoang Pham, “A summed up blame discovery programming dependability show subject to arbitrary working conditions”. Vietnam journal of computer science, Springer, pp 1-6, 2016
9. Karunanithi N, Whitley D and Malaiya Y K. “Expectation of programming unwavering quality utilizing connectionist models”. IEEE Transactions on programming engineering, volume 18(7), pp 563-574, 1992
10. Kirti Tyagi and Arun Sharma, A govern based approach for assessing the unwavering quality of segment based frameworks”, Advances in engineering software, Elsevier, volume 54, pp 24-29, 2012.
11. Mengmeng Zhu and Hoang Phan, “ A software reliability demonstrate with time subordinate blame discovery and blame evacuation”, Journal of computer science, Springer, volume 3, no 2, pp 71-79, 2016.
12. P K Kapur, H Pham, Sameer Anand and Kalpana Yadav, “A Unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation”, IEEE Trans on Relaibility, volume 60, no 1 , pp 331-40,2011.
13. Park J Y, Lee S U and Park J H. “Neural system demonstrating for programming unwavering quality expectation from disappointment time information”. Journal of electrical engineering and information sciences, volume4(4) : 533-538, 1999
14. Pratik Roy, G.S Mahapatra and K N Dey, “Neuro-hereditary approach on calculated model based programming unwavering quality expectation”, Expert systems with applications, Elsevier, volume 42, no 10, pp 4709-18, 2015
15. Pratik Roy, G S Mahapatra, Pooja Rani, S K Pandey and K N Dey, “Powerful feedforward and intermittent neural system based dynamic Q weighted blend models for programming unwavering quality expectation”, Applied soft computing, Elsevier, volume 22, pp 629-37, 2014
16. R Peng, Y F Li, W J Zhang and Q P Hu, “ Testing exertion subordinate programming unwavering quality model for blemished troubleshooting process thinking about both discovery and rectification”, Reliability engineering and system safety, Elsevier, volume 126, pp 37-43, 2014
17. Ramakanta Mohanty, V Ravi and M R Patra, “Crossover keen framework for foreseeing programming unwavering quality”, Applied soft computing, Elsevier, volume 13, no 1 , pp 189-200, 2013.
18. S Chatterjee and J B Singh, “ A NHPP based programming unwavering quality model and ideal discharge arrangement with logistic-exponential test scope under blemished investigating”, International journal of system assurance engineering and management, Springer, volume 5, no 3, pp 399-406, 2014
19. Subhashis Chatterjee, Jeetendra B Singhand Arunava Roy, “ A structure based programming unwavering quality designation utilizing fluffy logical chain of command process”, Int Journal of systems science, volume 46, no 3 , pp 513-25, 2015.
20. Tian L and Noore A, “Dynamic programming unwavering quality expectation : An approach in view of help vector machines”. Global Journal of Reliability , Quality and safety engineering, 12(4), pp 309-321, 2005.
21. Tracy Hall, Sarah Beecham, David Bowes, David Gray, and Steve Counsell, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering", IEEE Transactions on Software Engineering, volume. 38, no. 6, pp. 1276-304, 2012.
22. 22.Vahid Garousi, Ahmet Coskuncay , Aysu Betin-Can and Onur Demirors, "A Survey of Software Engineering ctices in Turkey", Journal of Systems and Software, Elsevier, volume 108, pp. 148-77, 2015.
23. WeiZhao, Tao, Ding Zhuo Shu and Enrico Zio, "A dynamic molecule channel bolster vector relapse strategy for unwavering quality forecast", Reliability Engineering and System Safety, Elsevier, volume 119, pp. 109-16, 2013.
24. Yubo Yuan, Houying Zhu, Bo Liu and Feilong Cao, “Software dependability displaying with expelled mistakes and exacerbated diminished rate, Mathematical and Computer Modeling”, Elsevier, vol. 55, no. 3, pp. 697-709, 2012.
25. .K.N. Krishnanand and D. Ghose, “Detection of Multiple Source Locations Using a Glowworm Metaphor with Applications to Collective Robotics”, Proceedings of IEEE Swarm Intelligence Symposium, Pasadena, California, June 2005, 84– 91.

AUTHORS PROFILE

Shailee Lohmor is pursuing her PhD in the area of software reliability and artificial neural network with emphasis on predicting software reliability from the Bharathiar University, Coimbatore. She is currently working as an Assistant Professor in the Department of Business Analytics, New Delhi Institute of Management. Her areas of interest include software reliability, business analytics, predictive analysis, and optimization techniques.

B.B. Sagar is currently working as an Assistant Professor in the Department of CSE, Birla Institute of Technology, Mesra Ranchi and posted at the BIT Noida Campus. He received his MCA from UPTU and PhD (Computer Science) from SHIATS Allahabad. His research interests are in software reliability, network reliability, parallel computing and distributed system. He is reviewer of various reputed SCI and Scopus international journals and conferences like Elsevier, Inderscience, IEEE and Springer. He has published around 30 research papers in journal and conferences of international repute. Recently, he chaired many IEEE international conference. He has been joined as a professional member of IEEE Computer Society. He is a Fellow of IETE and life member of Vijnana Bharti. He invited in various international summits and conferences as an invited special guest organised by Govt. of India and others.