Journal of Medical Systems (2018) 42: 93
https://doi.org/10.1007/s10916-018-0952-3

Prediction of Software Reliability using Bio Inspired Soft Computing Techniques

Chander Diwaker[1] · Pradeep Tomar[2] · Ramesh C. Poonia[3] · Vijander Singh[3]

Received: 29 August 2017 / Accepted: 27 March 2018 / Published online: 10 April 2018

Abstract
A lot of models have been made for predicting software reliability. The reliability models are restricted to using particular types of methodologies and restricted number of parameters. There are a number of techniques and methodologies that may be used for reliability prediction. There is need to focus on parameters consideration while estimating reliability. The reliability of a system may increase or decreases depending on the selection of different parameters used. Thus there is need to identify factors that heavily affecting the reliability of the system. In present days, reusability is mostly used in the various area of research. Reusability is the basis of Component-Based System (CBS). The cost, time and human skill can be saved using Component-Based Software Engineering (CBSE) concepts. CBSE metrics may be used to assess those techniques which are more suitable for estimating system reliability. Soft computing is used for small as well as large-scale problems where it is difficult to find accurate results due to uncertainty or randomness. Several possibilities are available to apply soft computing techniques in medicine related problems. Clinical science of medicine using fuzzy-logic, neural network methodology significantly while basic science of medicine using neural-networks-genetic algorithm most frequently and preferably. There is unavoidable interest shown by medical scientists to use the various soft computing methodologies in genetics, physiology, radiology, cardiology and neurology discipline. CBSE boost users to reuse the past and existing software for making new products to provide quality with a saving of time, memory space, and money. This paper focused on assessment of commonly used soft computing technique like Genetic Algorithm (GA), Neural-Network (NN), Fuzzy Logic, Support Vector Machine (SVM), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC). This paper presents working of soft computing techniques and assessment of soft computing techniques to predict reliability. The parameter considered while estimating and prediction of reliability are also discussed. This study can be used in estimation and prediction of the reliability of various instruments used in the medical system, software engineering, computer engineering and mechanical engineering also. These concepts can be applied to both software and hardware, to predict the reliability using CBSE.

Keywords Software reliability · Software quality · Soft Computing · Genetic algorithm · CBSE · Optimization Technique · CBSR

This article is part of the Topical Collection on *Systems-Level Quality Improvement*

✉ Vijander Singh
vijan2005@gmail.com

Chander Diwaker
chander_cd@rediffmail.com

Pradeep Tomar
parry.tomar@gmail.com

Ramesh C. Poonia
rameshcpoonia@gmail.com

[1] Department of Computer Science and Engineering, Kurukshetra University, Kurukshetra, India

[2] Department of Computer Science and Engineering, School of ICT, Gautam Buddha University, Greater Noida, UP, India

[3] Amity Institute of Information Technology, Amity University Rajasthan, Jaipur, India

Introduction

Software reliability is defined as running a system without any failure for a particular time interval. Predicting software reliability is a research issue due to many challenges in predicting and estimation of software reliability. To make the system more efficient and less error, less maintenance, there is need of predicting and estimating software reliability by applied recent techniques and methodologies. The system may become more suitable for combining other factors affecting reliability with reusability. The research is going on to reduce complexities and failure rate in the system. It is a difficult task to compute the best cost where there is a large area with a population with the random movement of many components. Component-Based Software Development (CBSD) helps in making reliability model in easy and efficient manner.

It is a difficult task to make a new system in more efficient manner and in very less time interval. Therefore, CBSE may be used to make new software to save time and money by providing quality and high reliability. Goulão (Goulão, 2005) defines CBSE as a section of software engineering that depends on component re-usability. A software component formed a component model by integration of various components for performing a particular function. The components can be separately positioned and compiled without a change in attributes as per composition standard. Thus Component-Based Software Reliability (CBSR) depends on interaction among reusable components. CBS relies on connections of components. If the connectivity of components is complex, then it is difficult to estimate CBSR. The choice of components depends on interfaces between components and reusability of components. These selected components assist in components integration. Soft computing becomes popular in the research area of estimating and predicting reliability. Many optimization techniques have been used that can help in providing best local/global cost to achieve the target.

It is a difficult task to make a new system in more efficient manner and in very less time interval. Therefore, CBSE may be used to make new software to save time and money by providing quality and high reliability. A software component formed a component model by integration of various components for performing a particular function. The components can be separately positioned and compiled without a change in attributes as per composition standard. Thus CBSRe depends on interaction among reusable components. CBS relies on connections of components. If the connectivity of components is complex, then it is difficult to estimate Component-Based Software Reliability (CBSRe). The choice of components depends on interfaces between components and reusability of components. These selected components assist in components integration.

Related work

Various reliability models have been proposed by taking different parameters consideration. Different optimization techniques have been used to estimate reliability which is discussed below in various author's work.

A lot of models have been made for predicting software reliability. The reliability models are restricted to using particular types of methodologies and restricted number of parameters. There are a number of techniques and methodologies that may be used for reliability prediction, but there is a lack of software reliability prediction in case of CBSE. CBSE boost users to reuse the past and existing software for making new products to provide quality with a saving of time, memory space, and money. But to get the optimized result, researchers and practitioner used soft computing techniques for small as well as large-scale problems where it is difficult to find accurate results due to uncertainty or randomness. This study focused on assessment of commonly used soft computing technique like GA, NN, Fuzzy Logic, SVM, ACO, PSO, and ABC. It is a difficult task to compute the best cost where there is a large area with a population with the random movement of many components. CBSD helps in making reliability model in easy and efficient manner. This work presents working of soft computing techniques and assessment of soft computing techniques to predict reliability based on CBS.

Analysis of neural network

Chen and Wang [1] proposed a model based on back-propagation NN for estimating failures of a software system during the maintenance phase. This is a straight and fast method for calculating failure. To represent the application and validation of the proposed method, an illustration of a commercial shop floor control system was utilized. Proposed model lying on BPN improved estimation for software failures. The performance regarding the model was measured by calculating the mean average percentage error. The input data models were enhancement and failure correction records and output was future failure time.

Zheng [2] presented two types of models for calculating software reliability such as parametric and non-parametric. No single parametric model able to achieve the accurate prediction in all cases. This was a non-parametric model for purpose of SRP, based on NN ensembles. The comparison was performed among proposed system using single NN based system and with the parametric not homogenous Poisson process models. The experimental outcome showed that by combing numerous NNs, the predictability of system can be considerably improved. Software execution time was the input of system and predicted failure numbers is the output of the system. This model considered three different rules for the combination module i.e. Mean rule, Median rule, and Weighted mean law. For comparing the different models the term, variable term-prediction was used and performance measures of variable term-prediction were evaluated using relative Error and Absolute Error.

Singh and Kumar [3] proposed software reliability prediction (SRP) model based on feedforward NN. The proposed technique used backpropagation training algorithm for improved reliability prediction. A comparison among considered approach and other SRP models had been shown via utilizing the seven distinct failure data sets composed of standardized software projects for testing validity of the considered method. Software ET was input for the considered method and number of cumulative failures was the output of the system. The effectiveness of proposed model was calculated by parameters such as variable-term-predictability, RMSE, mean average error. In this paper MATLAB, 7.0.1 environment is used for representing execution time and a number of failures.

Bisi and Goyal [4] introduced model by utilized NN for estimating an accumulative number of failures along with two encoding scheme that was exponential function and logarithmic function. Depending on the existing data of software failure, encoding of execution time was performed by utilizing the exponential function and logarithmic function. Performance of considered method had tested on eighteen data sets of software failure. Numerical result illustrated that considered approach provided acceptable outcomes across several software projects. The considered model had a good capability of prediction as compared to other models. The model calculated three parameters like Root Mean Square Error (RMSE), average relative error and Relative RMS.

Arora and Choudhary [5] Applied feed-forward neural networks model in support of software reliability growth prediction. This novel approach used testing and debugging data from several software projects. The NN method showed consistent performance in estimation and predictive performance was comparable with the parametric model. This model calculates two parameters i.e. execution time and faults. The units for execution time and faults were days and cumulative faults at end of every day. The results demonstrated total time for debugging and examining was 46 days, and faults were 266. MATLAB was used for graphical representation.

Kumar et al. [6] determined the problem regarding assumptions that always needs, before starting a project that is a critical task. For attaining the objective, from the similar projects failure history was collected and after that, applied for constructing several models of NN for prediction. Proposed method showed how distinctive data sets able to be employed to NN model and attain the optimal solution between them. Parameters that judge the capability of NN model were a relative error, absolute mean error, average bias, normalized average error and rank metric.

Ramasamya and Lakshmanan [7] proposed simple log power SRGM with exponential testing effort function estimated parameters with the artificial neural network. Performance comparison of SRGM was done when parameters were estimated in the traditional way. The performance was also compared with ANN including testing effort and without including testing efforts. Three vital criteria for which result evaluated was measured are named as Akaike Information Criterion (AIC), R^2 (Coefficient of determination) and RMSE. RMSE was used to compute the difference between actual value and the predicted value. AIC computed the better fit and utilizes it to rank the models, and also it gave a penalty to a model having numerous parameters.

Analysis of genetic algorithm

Aljahdali and El-Telbany [8] utilized multi-objective GA in support of SRP by estimating the faults through the software testing procedure via utilizing the software faults historical data. This scheme considered three main approaches that were weighted formula approach, lexicographic approach and the Pareto approaches to deal with multi-objective difficulties. This scheme used three datasets that were Operating System, Real Time Control, and Military. When autoregression model and AR model plus GA algorithm, applied to data sets outcome shows that ensemble model had superior performance than a single model. Parameters of GA were population size, mutation, number of generations, speed, crossover rate, and selection method. The comparison between single model and ensemble models was shown with parameters: share normalized RMSE and Correlation Coefficient (R^2).

Kim et al. [9] proposed an effectual approach named as Real-Valued GA (RGA) for estimating the software reliability. RGA basically used the real-valued operator that quickly converged the optimal value into a continuous domain. Modeling the efficient genetic operators within RGA is easier as compare to Binary GA (BGA) because genotype is same as a phenotype in RGA. RGA is not produced any special like ± infinity and Not a Number (NaN) and also RGA does not need any procedure to encode and decode the chromosomes among bit strings and actual values because of real data operations. The experiment was accomplished on eight data sets which are real and the outcome shows that RGA was better than other existing Genetic approach. Fitness function, selection operators, heuristic crossover and non-uniform mutation were a parameter used in RGA. For comparison among RGA and another genetic approach, MSE was the comparison criteria.

Sharma et al. [10] presented a group of methods that utilize a GA for generating the test data. These methods have different parameters for automatically producing the structural-oriented test data based on internal program structure. Discovered factors were utilized in calculating the fitness function of GA, for choosing the preeminent possible Test method. Test populations were taken as an input by these methods and the evaluation of test cases was performed for that program. This integration helped in enhancing GA within investigation space examination and exploitation area with improved convergence rate. Few terminologies that might be used during implementation of GA: Individual, Search Space, Locus, Chromosome, Trait, Population, Allele, and Genome. GA was implemented with different tools like Ruby, C++, and MATLAB.

Fazel [11] focused on a method that uses GA for predicting software error. The objective of the considered method was to predict error of software with higher accuracy and speed, and moreover to represent a structure that can execute and expand easily. The achieved result of this method showed a preferred performance from the time period for predicting error and recognition or output rate. The outcome showed that a recognition rate of the proposed method was more than 95% in the best condition. For measuring the performance of the proposed method, some subsequent parameters had changed with every simulation i.e. dataset, number of GA generation, the population requirement of education and testing, and changes in the composition operator. Results of detection simulation of software error were provided by applying MATLAB tool.

Analysis of chaos theory

Rotshtein [12] utilized a convenient methodology which is an incorporation of fuzzy logic with chaos theory. Fuzzy sets membership functions incorporated into a logistic map like chaos generator were utilized for creating reliability bifurcations map of the system with the redundancy of components. This approach explained that increase in a number of redundant elements delaying the jiffy of the first bifurcation associated with a loss of reliability and decreases the orbit size. This approach involved the problem statement of redundancy optimization below conditions of chaotic behavior of affecting parameters and GA to solve the problem. A variable called "failure possibility" was modeled through membership function of the system.

Quin [13] implied a novel method for analyzing and resolving software security and reliability modeling. The first section included the study of failure behavior and artificial nature of software system precisely, obtaining the law and characteristics of software failure and secondly section offered a measure to a degree, for the software reliability criteria that can be utilized for testing the indicators of system reliability and security. Finally, through chaos theory SR modeling and analysis, software data failure analysis, reconstructing phase space and forecasting results through actual results were performed. Two error functions were used for analyzing the advantages and disadvantages of SR chaotic model instance results and termed as Relative Error (RE) and Mean Error (ME).

Tong et al. [14] proposed a novel approach based on chaotic time series and Heterogeneous Ensemble Learning (HEEL) for SRP. This scheme firstly performed chaos identification for identifying whether data of software failure was chaotic and after that used various weak learners for constructing a HEEL model that would be trained through previous failure data. Finally, by using a trained model prediction was performed. Two real-world datasets of software failure i.e. Musa data and National Trauma Data Standard (NTDS) data were used for a case study. Predictive performance was calculated by performing a comparison among two SRGMs and two data-driven models. The outcome demonstrated that proposed approach worked best, and had an excellent forecasting and performance. This approach utilizes MSE as the fitness function. Two norms, RMSE, and average relative error were selected for evaluating the prediction performance.

Analysis of differential evolution

Becerra et al. [15] introduced test data generator that used DE for resolving constrained optimization problems and evaluated its performance for various DE models empirically. With the objective of comparing this technique to other approaches, experiments were extended to the Breeder GA, and comparison of various test data generators with DE approach is performed. The outcome introduced DE as an assured solution technique for the real-world problem. The result showed that DE was robust, mainly in constrained optimization, getting good results in a huge range of problems, and needs a relatively less number of function evaluations. Some of the program characteristics were a number of parameters creating an input, the parameter type, the bounds values, number of branches in source code and the higher number of critical conditions.

Nasar and Johri [16] focused on distributing entire testing resource optimally underneath the dynamic situation. A detailed optimization policy established from the optimal control theory was proposed by using Differential Evolution (DE). DE is an enhanced form of GA for quicker optimization. DE has uncomplicated structure, simple to use and robust. This approach proposed that testing of software and debugging should be observed as simultaneous behavior. Numerical analysis showed that objective assigning of testing effort via using DE. Parameters of DE are population size, a number of generation, crossover constant (CR), differentiation constant (F).

Review and analysis of soft computing techniques

Mahadevan and Rebba [17] developed a Bayesian methodology for validation of reliability model by comparing between model prediction and testing data, both of which have uncertainty. A large complex computational sculpt may be decomposed into smaller modules. Propagating sub-module validation data was used to assess overall reliability forecasting model through Bayes network approach where the full-scale experiment is not viable. The statistical sharing of overall presentation function was updated using validation test data for individual sub-modules. The posterior ratio and prior densities at the predicted values of performance function were utilized to validate model forecasting. The concept of Bayes network may be expanded to the case of system level reliability prediction involving multiple limit states. The methodology may be expanding by including correlated measures in in-between nodes and effect of justification at these nodes on the overall metric for the reliability prediction model.

Bai [18] utilized the Bayesian network for modeling software reliability forecast through an operational profile. Owing to the complication of software products and development process, SRM requires possessing the ability to arrange with multiple parameters. A Markov Bayesian network had been applied to make a model for reliability prediction. This model was an extension of Markov Bayesian network that was built for forecasting software reliability with the operational profile. It focused on discrete-time failure data. As a result, the recommended algorithm, the Gibbs sampling-based Algorithm, would be time-consuming. Beta distributions were implemented as priors for the case study. The important property of Beta distribution is that it is conjugated to the binomial distribution. Thus Beta distribution fits very well for the variables that are defined in (0, 1).

Pai and Hong [19] surveyed the possibility of usage of SVM to predict software reliability. Simulated Annealing (SA) was used to choose the constraints of SVM model. Examples were obtained from current literature and were utilized to present the effectiveness of reliability forecasting. The result showed that SVM model utilizing SA provided better calculations than other methods. A relative study of the estimating performance of several models was evaluated. The greater forecasting capability of the model was due to using of SA algorithms in choosing SVM parameters and minimizing the structural risks in SVM-SA models. Hence, the planned model was a promising alternative for predicting software reliability.

Elish and Elish [20] explored the SVM potential in estimating defect-prone units and evaluated its performance with machine learning models and eight statistical data in the perspective of 4 NASA datasets. In all datasets, the overall correctness of SVM was in the range of 84.6–93.3%; its precision was in the range of 84.9–93.6%; its recall was in the range of 99.4–100%; and its F-computation was in the limit of 0.916–0.965. During Consideration of F measure, SVM attained superior F-computation than at least five out of the eight contrasted models in the entire datasets and was not considerably outperformed by any model. The outcome presented the estimation performance of SVM was generally improved than other models. The research may be continued by carrying out further empirical studies with other datasets and to recognize the full perspective and possible drawback of SVM. Another research issue is to consider additional self-governing variables like coupling and cohesion measures.

Kiran and Ravi [21] proposed ensemble models to calculate software reliability efficiently. To calculate software reliability, tree linear and one nonlinear ensemble was formed and tested. In particular, a nonlinear ensemble was trained by using Back Generation NN (BPNN). This approach utilized application of all techniques prediction potential towards information and suitably assigns weights to methods relied upon their performance. Various statistical and smart methods constitute the ensembles. These various techniques applied to forecast software reliability were: (i) Threshold Accepting-based NN (TANN), (ii) Back Propagation NN (BPNN), (iii) Pi–Sigma-Network (PSN), (iv) Multiple Linear Regression (MLR), (v) Tree-Net (vi) Multivariate Adaptive Regression Splines (MARS), (vii) Dynamic Evolving Neuro-FIS (DENFIS) and (viii) Generalized Regression NN (GRNN). Based on the numerical experiment, the not linear ensembles outperformed all further ensembles with constituent statistical plus smart techniques. The ensembles developed can use as feasible alternatives to present methods for prediction of software reliability.

Fenton et al. [22] presented an approach that uses Bayesian networks (BNs) in forecasting software reliability and defects. This scheme allowed the examiner to include fundamental factors in joining quantitative and qualitative computation. While such BN models had proven to be useful, their accuracy was traditionally constrained by two factors a) The inevitable subjectivity resulting from expert elicitation. b) The static discretization necessary in BN inference algorithms. The new scheme to inference using dynamic discretization removed the previous constraints and also makes it much easier to build and modify BN models with continuous nodes. The dynamic discretization algorithm results in significantly more accurate predictions with only minimal increases in computation time.

Doguc and Marque [23] presented a method for building a Bayesian network (BN) to estimate system reliability. This method utilized historical data related to the system to be modeled and in making BN model without the requirement for human intervention. The K2 algorithm was utilized for this purpose which is an efficient relationship rule mining method. This algorithm used a heuristic to provide efficient and accurate results while searching for associations. No human involvement is necessary during the construction of BN and reliability estimation. According to the experimental results, reducing the running time of finding associations from O(2n) to O(n2), the proposed methodology can work efficiently even with substantially large systems. Moreover, the BN models constructed using K2 algorithm were shown to be accurate, especially when more past data related to the system is available. As expected, the outcome showed that when 1000 historical observations on the system are available, the constructed BN is more than 90% accurate. The accuracy of the K2 algorithm can further be improved when the already existing associations between system components are taken into account.

Aljahdali and Sheta [24] proposed an SRGM using fuzzy logic. The model contained a group of linear sub-models connected together smoothly by utilizing fuzzy relationship functions to signify the fuzzy model. A fuzzy on linear regression model was built for calculating the accumulated defects of software engineering purposes. The built model executed based on Takagi-Sugeno technique. The designed fuzzy models were tested utilizing three types of relevance's: operating systems, military, and real-time control purposes. Dataset related to results and analysis were developed by Musa are presented to prove the potential reward of utilizing fuzzy logic in resolving this difficulty.

Moura et al. [25] discussed regression though SVM that was utilized for forecasting reliability time-series and time-to-failure data of engineered components. The data-driven and empirical techniques provided a better choice to manage the complexity of the systems for realistic application of nonlinear reliability prediction. Support Vector Machines Regression (SVR) results were compared with the other data-driven empirical technique's results such as Radial Basis Function (RBF), Infinite Impulse Response Locally Recurrent Neural Network (IIR-LRNN), Box-Jenkins Autoregressive Integrated Moving Average (ARIMA) and the traditional multilayer perception (MLP) models. The achieved SVR showed better results as compared with other learning machines. Further, no shorten parametric consideration was required to calculate future failure times. Finally, SVR showed significant performance in managing wavy and smooth time series data.

Singh and Toora [26] developed Neuro-fuzzy-hybrid algorithm proposed for the component classification. The previous research deals with reusability of software components. The elements used were reuse frequency, regularity, complexity, volume, and coupling. In these data search applications, the design of Neuro-fuzzy hybrid algorithm had exposed its dominance because it contained the advantages of fuzzy plus neural networks. Neuro-fuzzy algorithms are certainly better than Fuzzy algorithm because it inherits adaptability and learning. The developed model of Neuro-fuzzy for software reusability had been simulated in MATLAB. The outcome showed less percentage average error in Neuro-fuzzy algorithms.

Rana and Yadav [27] constructed a fuzzy based software based software quality estimation approach to examine the software criticality. Fuzzy rules were trained to find all risk aspect in software. Software risk analysis was also one of criterion to find the software reliability under risk vector. This approach explained software risk on three levels. These three levels were individual vector risk, subcategory risk, and the aggregate risk. This was a layered approach in which input was software execution time, whereas the outcome of the system was a number of failures. The approach used fuzzy strategies to enhance association mining for predicting software reliability. This exertion was applied at various places as it has a range of applications.

Ziauddin et al. [28] introduced model utilizing fuzzy logic to enhance the exactness of software effort assessment. This model was employed for fuzzify input factor of the COCOMO II model and the outcome was de-fuzzified to get the ensuing attempt. The planned model relied on Fuzzy Logic and COCOMO II. The COCOMO II contained three sets of inputs software attributes: 17 effort multipliers (EMs), 5 scale factors (SFs), one size of KDLOC (SZ) plus one output i.e. is an effort. To symbolize the linguistic conditions inside the model, triangular fuzzy statistics were used. The result of the model was contrasted with COCOMO II with Alaa Sheta Model. This model provided a better result than other models.

Yang et al. [29] introduced a concept to solve large-scale difficult software system utilizing Bayesian Network that supported SRM methods. The task flow oriented software reliability simulation prediction methods were combined together. Bayesian Network can estimate the first reliability for difficult software system by structured learning and parameters learning from software structure and the possible history data. This method can broadly make use of previous information of software architecture, the past data and software task flow to perform the active reliability forecast and find the reliability weaknesses at the same time. Single Train network Control & Management System (TCMS) software was selected as the experiment application to verify its feasibility and validity. The future research directions will focus on modeling of the external abnormal events to simulate fault injection function based on a proposed method to realize other realistic and accurate reliability analysis and estimation.

Khosla and Soni [30] optimized the fuzzy membership functions (MF) to manage the deviation in pendulum angle and velocity using ABC and ACO. With the ABC based fuzzy, the inverted pendulum remained in steady state with less error. The parameters considered were pendulum position, angle, velocity, integrated control. The outcomes showed better performance in case of proposed fuzzy controller relied on ABC algorithm.

Elloumi et al. [31] proposed an approach included combining Fuzzy Logic with ACO (FACO) and PSO (F PSO)for solving TSP. The aim was to find the best path with less time for TSP. The outcome showed that as increase the size of the population, path length and execution time of FACO and FPSO decreases. The outcome showed better execution time of PSO is better than ACO.

Shanmugam and Florence [32] estimated accuracy for software reliability model using ACO. The suitability of the accuracy was analyzed on Poisson and binomial models. Using this approach, space and time complexity was also reduced. Probability, failure rate, number of intervals, fitness function, ants, time interval were used for predicting reliability.

Roopa and Reddy [33] utilized PSO approach to generate design choices in the design phase and assisted in design decision at the time of building CBSS. Fitness value and computational time of PSO and GA for a different number of iterations were compared. The results showed that PSO shows better outcome as compared to GA.

Singh et al. [34] present a model to assess the reusability using FL. The parameters considered were modularity, maintainability, flexibility, interface complexity, and adaptability. Different membership functions such as Gaussian membership, trapezoidal and triangular were utilized. 243 fuzzy sets were created and membership functions were classified as Least, Less, Modder, More, Most.

Okutan and Yıldız [35] proposed model relied on the Bayesian network in which the interaction among metrics and defects proneness on multiple datasets. The metrics applied to Promise data repository. It defined two metrics: Numbers of Developer (NOD) and Source code quality. The experiments executed on nine open font Promise data repository dataset shows that result for class (RFC), lack of coding quality (LOCQ) and lines of codes (LOC) were the most valuable metrics while weight method per class (WMC), coupling among objects (CBO) and lack of cohesion of methods (LCOM) were less efficient metrics on defect-proneness. It provided partial outcome on numbers of children (NOC) and depth of inheritance tree (DIT). In future direction, it can include other software and procedure metrics in this model to expose the relationships among them and to perform analysis during defect estimation.

Tyagi and Sharma [36] proposed an ANFIS model for estimating CBSR relied on basic components of soft computing and evaluated its performance with that of plain FIS for different datasets. This was a hybrid method that requires less calculative time than conventional schemes and the previously proposed FIS approach. The RMSE value was calculated for the outcome attained by FIS and the outcome attained by ANFIS with the actual output. Using the ANFIS, first trained FIS, and the rules were created based on training data to produce an output of the trained model. Hence, the ANFIS performed better than the FIS. The only limitation of the model was its complex execution for big data sets.

Lal and Kumar [37] used fuzzy logic to estimate the reliability of various CBSS. Various rules were applied on FIS for designing and inspection of reliability for the CBSS. MATLAB was used for simulation and observing results. The main steps involved were the identification of components, design, and analysis of reliability for CBSS, and compare the reliability of proposed model with the existing model. The result showed that the proposed approach presented a better outcome than the conventional approach of estimating software reliability.

Tyagi and Sharma [38] introduced Heuristic Component Dependency Graphs (HCDGs) to estimate CBSS reliability. This paper also proposes component reliability and CBSS reliability. This algorithm is an extension of ACO named as ACOREL. This algorithm identified the most used path. This path helps in estimating path reliability. A parameter such as reliability of average execution, component time, component path probability, pheromone amount, heuristic information, number of components was used in estimating CBSR.

Diwaker et al. [39] discussed various metrics which were classified on basis of system level and component level. These metrics help in measuring different characteristics of components. The factors affecting CBS system were also discussed. In future, these metrics and factors may be used with soft computing techniques to estimate CBSR in an efficient manner.

Jaiswal and Giri [40] estimated CBSR using the FIS and ANFIS with two different number of relationship function. After comparing the output reliability, values for different input sets were analyzed. FIS and ANFIS models offered better result for 5 membership function as comparing three membership functions. Here, CBSR estimation was performed based on only four factors that are Reusability, Operational profile, Application complexity and Component dependency. But CBSR was exaggerated by other parameters like Software quality, Fault density, Together with functionality, availability, usability, performance, serviceability, capability, installability and maintainability. In future work, other factors may be also added.

Diwaker and Tomar [41] utilized the CBSE metrics such Component efficiency, component dependency and component density for assessing ACO methodology. ACO can be used to determine the component interaction and reusable components in a system that leads to increasing the reliability of the system. MATLAB was used for implementation of ACO. The outcome showed as component efficiency increased, the component density also increased.

Diwaker and Tomar [42] presented an appraisal of PSO with CBSE metrics. **A** new fitness function for PSO was proposed and the results were presented with help of metrics like component interface complexity metric, component Functionality Metric, and average execution time. The subparameters used were interaction among components, reusability, and usage of resources.

Bolisetty and Yalla [43] proposed Hybrid PSO– Cuckoo Search algorithm for building an adaptive software design relied on Process control model. The components were selected by test cases generator. The adaptive architecture was built using PSO - CS on basis of clustering results. The attributes like functional requirements, evaluation, responsibility, modifiability, efficiency, and traceability were considered for reliability estimation.

Diwaker and Tomar [44] focused on the evaluation of PSO, ABC, and ACO utilizing CBSE reusability metrics. The best of above optimization techniques were evaluated using CBSE reusability metrics. LOC, number of function, reusable components in the coding of ACO, ABC and PSO were parameters for evaluating these optimization techniques. MATLAB was used for implementing these optimization techniques.

ACO showed better reusability for component integrity than ABC and PSO.

Preethi and Rajan [45] discussed various techniques of reliability evaluation such as reliability analysis by using path testing designed for a compound CBSS. It utilized the collected failure information in testing stages to estimate failure rate in the operational situation. In a practical environment as a function or service, various exterior factors may also affect when different schemes are applied for reliability forecasting during the development of life-cycle phase. This work focused on software reliability estimation using path testing for CBSS, SVM, virtual sample-based model, methods based on correlated component failures and quality indices formed in all stages of the software lifecycle. These techniques helped to determine current meaningful knowledge to improve techniques used in firmware reliability prediction.

Rizvi et al. [46] focused on fuzzy logic to treat with worries and vagueness engaged in early stage measures. The model used in Early Stage of Reliability Estimation Model in which integrated requirement plus design metric taken as Input to the FIS to forecast software reliability. There were four input variables in requirement phase and five input variables in design Phase. The prophetic accuracy measure was done by means of Pearson Correlation Coefficient which was encouraging and supporting. The MATLAB was utilized for validating the developed representation and computed the various predictive accuracy measures to ensure its prediction efficiency. The result of the ESRP model is better and improved reliability estimation model.

Rizvi et al. [47] proposed a highly planned structure that described the process of quantifying software reliability, before the coding of software start. The fuzzy rules had been utilized to overcome the limitation of partisanship of requirements stage measures. This study highlighted the feeble point of earlier software reliability forecast efforts, and subsequently developed a structured framework that overcomes the inadequacy of earlier study and quantifies the reliability, on basis of the requirements and designs phase measures, prior to the coding starts. It contains eight different phases: conceptualization, identification, association, quantification, corroboration, analysis, assessment, amendment and packaging. In future work, this framework opens fresh avenues for the researchers, doing research on reliability estimation.

Dubey and Jasra [48] proposed a reliability estimation model for CBSS utilizing ANFIS. The model considered the crucial factors that affect the reliability of CBSS. The hybrid NN was used in ANFIS which was trained by utilizing the data sets. This NN guided rule based on FIS. This model depended on adaptive learning and decision creation capability of ANFIS. An evaluation using Mamdani FIS was also proposed. The outcome showed higher efficiency in ANFIS model to determine the consistency of a CBSS than that of FIS. The future work may be to design an model that considers factors include in CBS. These will include internal, external as-well-as development process factors.

Sharma and Gandhi [49] proposed a model for estimating CBSS reliability by utilizing Modified Neuro-Fuzzy Inference System (MNFIS) concepts. This model relied on four factors: Component dependency, Operational profile, Reusability and Fault Density. The performance of the model was compared with FIS. MNFIS show better performance than FIS. An additional factor i.e. fault density was included in this model. The sensitivity analysis was also compared for two models. It was observed that MNFIS was more stable than FIS model.

Research gap and open issues

As discussed in section 2.5, a lot of research has been done to estimate software reliability. The main challenges faced by researchers and practitioners are identifying significant parameters that affects reliability of software and modeling those parameters in well defined manner to predict the software reliability.

Soft computing techniques

Optimization techniques become popular in optimization solution for large problems. Optimization techniques have many applications. There are several commonly used optimization techniques like Genetic Algorithm (GA), Neural Network (NN), Fuzzy logic, Support Vector Machine (SVM) and Swarm Optimization methods like Artificial Bee Colony (ABC), Ant Colony Optimization (ACO) and Particle Swarm Optimization etc.

The working of these optimization techniques mainly based on activities of components in a limited area and these activities are used to compute optimal solution for a particular problem. The working mainly consists of sharing of information by neighbors to achieve the goal. The working of various soft computing technologies described as follows:

Genetic Algorithm (GA)

GA was developed by *Goldberg* [49] which was inspired by *Darwin's* theory of evolution. *Hermawanto* [50] utilized GA for solving mathematical equality problem. It includes chromosome, collection of which is called population. A chromosome is made from genes, the value of genes may be numerical, symbols, binary or characters based on the type of problem. There is a requirement of fitness function which is fooled by Chromosomes to compute validity of the solution. Crossover is a process of creating new chromosomes called offspring which includes the genes combination of their parent. In a generation, some chromosomes are also mutated in their gene. Crossover and mutation processes are managed by the value of crossover and mutation rate. A chromosome with higher fitness rate has a high probability will be selected for next generation. After various generations, the value of chromosome is converged to an assured value which provides the best solution to a problem.

Application: strategic asset allocation, cryptography, **TSP and sequence scheduling, Robotics etc.**
Limitations: Selection of fitness function and coding for fitness function is a difficult task. The wrong selection of parameters for mutation and crossover rate results in the wrong output. In most cases, GA provides poor results in small problems.

Fuzzy Logic (FL)

It is a mapping of unknown input statistics information to a scalar statistics data. *Albertos* [51] build a controller using FL. It includes four parts: fuzzifier, inference engine, rules, and defuzzifier. An architecture of the fuzzy system is shown in Fig. 1.

Application: Controlling speed of motors and controlling the temperature in various devices.
Limitation: making fuzzy rules is a difficult task, time consuming, complex for large problems.

Neural Network (NN)

NN can be utilized to detect trends and extract patterns that are too complex. *Awodele and Jegede* [52] discussed the application of NN. A trained NN is considered as an expert. The information in NN is processed in the same manner as in the human brain. NN includes a large number of interconnected processing components called neuron that worked in a parallel manner to solve a particular task. Its operation may be unpredictable. The conventional and NN algorithms are a complement to each other.

The architecture of Neural Networks: Bose had classified NN in feedback and non-feedback network. *Haykin* [53] has classified NN into three types:

- Feed-forward Networks: One-way Communication without a loop. It includes single-layer perception and multi-layer perceptions.
- A feedback network supports bi-direction operation with one or more feedback operation.
- Network Layers: General ANN includes three groups of units: input (raw information), hidden (activities depends on weights of connections among input elements) and output units (activities depends on weights among hidden and output units).

NN consists of the learning process. It concerns with memorization of patterns and network response that includes associative mapping and auto-association. Learning techniques used for adaptive NN is separated in supervised learning and unsupervised learning. The back-propagation algorithm helps in computer error derivative of weights.
Application: NN can be practiced in adaptive learning, Real-Time Operation, Self-Organization and Fault Tolerance through Redundant Information Coding, sales forecasting, industrial process control, customer research, data validation, Risk management, recognition voice,

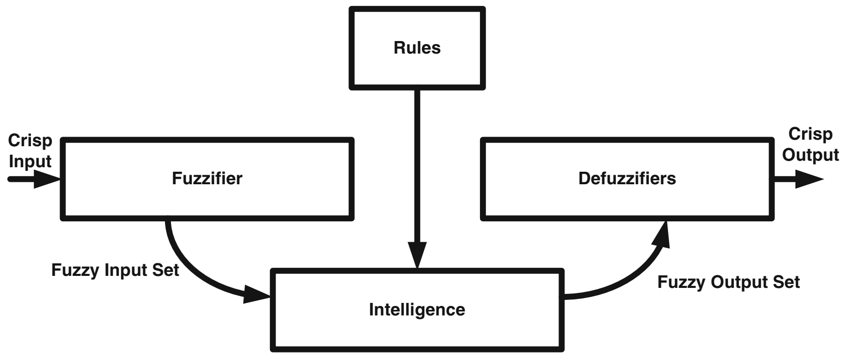

Fig. 1 Fuzzy logic

diagnosis of hepatitis, mine detection undersea, texture analysis, three-dimensional object recognition, recognition of hand-written word, facial recognition, Optical Character Recognition, Stabilizing Controller, Weed identification, Electrical signals occurring from impulses of human's receptor organs and water management etc.
Limitations: Black box characteristics, large computational, difficult of mapping of the solution to cover problem, experiential nature of sculpting development [54].

Particle Swarm Optimization (PSO)

Das et al. [55] discussed working and applications of *PSO* include random movement of components in search space to attain the goal with low cost and high velocity. Each component updates its information depends on the gross foremost velocity of particles.

Figure 2 shows working of PSO. The velocity of various components changes according to their past experience, searching skill, and information nearby. The fitness function plays important role in PSO. The fitness function is chosen as per requirement. It is a difficult procedure due to randomness. Figure 1, shows global best result attained after evaluating the outcomes of executing a number of iterations.

Application: telecommunications, combinatorial optimization, data mining, power systems, constrained problems, signal processing etc.

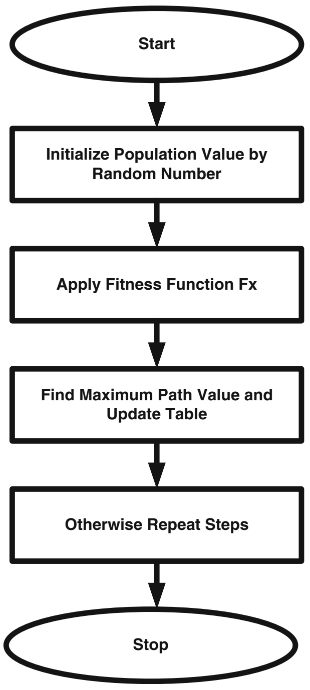
Start
Initialize Population Value by Random Number
Apply Fitness Function Fx
Find Maximum Path Value and Update Table
Otherwise Repeat Steps
Stop

Fig. 2 Working of PSO

Limitation: Less accuracy due to a different direction and random motion of particles. PSO may cause problem in non-coordinate environment

Support Vector Machines (SVM)

SVM consists of the optimal hyperplane for linearly separable patterns. It includes kernel function that helps in pattern recognition as support vectors. *Karamizadeh et al.* [56] discussed advantages and limitations of SVM. Support vectors are data points lie closest to decision surface. SV provides an optimal solution on decision surface. SVMs can maximize margin near separating hyperplane. The decision function is specified by a subset of training samples and support vectors.

Applications: Quadratic programming problem
Limitation: Slow, Expensive, time-consuming. The distribution of Support Vector (SV) is difficult.

Ant Colony Optimization (ACO)

It consists of behavioral actions of ants. *Katiyar et al.* [57] discussed working of ACO. The Fig. 3 shows working of ACO in which quantity of pheromone is kept in the path by ants while moving according to amount and type of food.

This path with maximum pheromone is considered as favorable track and followed by all other ants of the colony. The ants, pheromones, and target are the main constituents of ACO.

Ant System defines the area for the motion of ants and ants updated their information according to pheromone value. It includes three algorithms: Ant-density, Ant-quantity, Ant-cycle.

Applications of ACO: TSP, VRP, GCP, Graph Coloring Problem, Sequential Ordering Problem, Job Scheduling problems, Assignment, image processing, network model problem etc.
Limitations in ACO: It is hard to set and find the value of the objective function, decision parameters/constraints for dynamic and stochastic problems. In multiple objectives, it is a complex task to find a quality solution.

Artificial Bee Colony (ABC)

The main components of ABC are employed, unemployed foragers and food sources. *Karaboga and Akay* [58] discussed the working of ABC is based on cooperative information of individuals that help in making decision quick and in a better

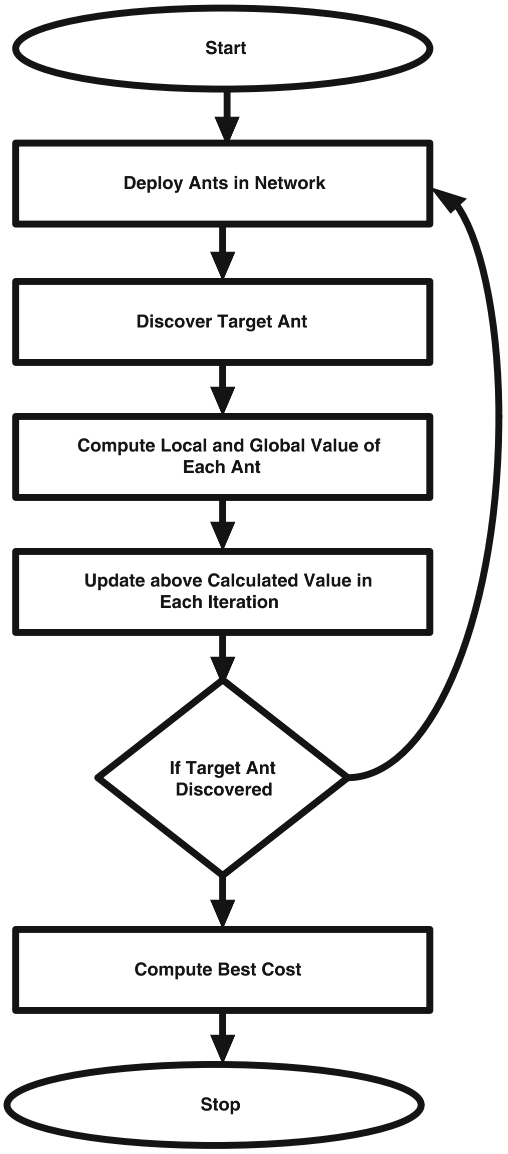

Fig. 3 Working Model of ACO

manner. Many Complex tasks can be assessed using distributing data collection and interaction between workers. Figure 4 shows working with ABC.

Application: Job shop, Flow Shop and Open shop, planning problems, TSP, spam detection, data mining etc.
Limitation: Mapping waggle dance to the outcome of any job is a complex task. Pre-knowledge of various factors is difficult.

Assessment of soft computing techniques

The overall assessment shows better results in PSO and fuzzy. ACO provides a better result in terms of cost. The best cost

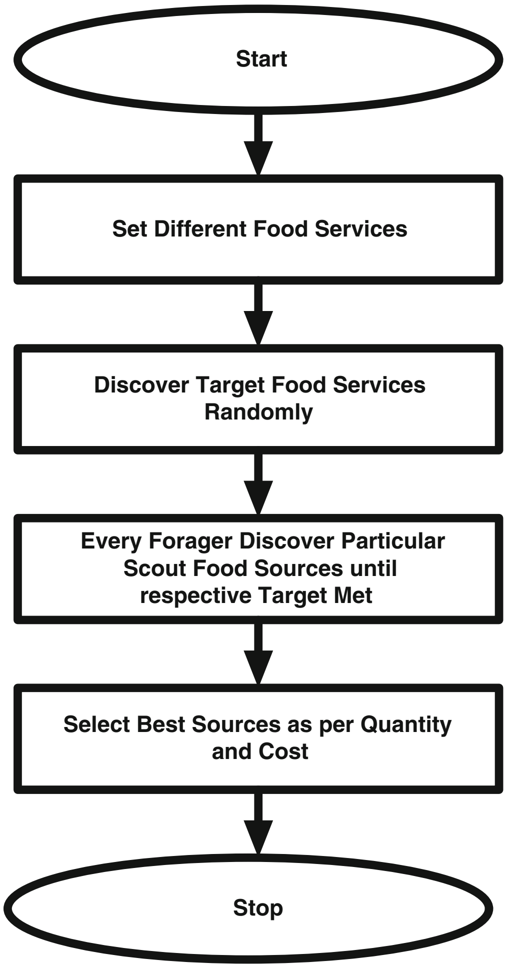
Start
Set Different Food Services
Discover Target Food Services Randomly
Every Forager Discover Particular Scout Food Sources until respective Target Met
Select Best Sources as per Quantity and Cost
Stop

Fig. 4 Working Model of ABC

provides better results with save of time and resources. Table 1 presents the concluded assessment information based on a review of the literature. Various optimization techniques have been used to evaluate software reliability and CBSR as shown in Table 1. In Table 1 SRG is Software reliability growth, SR is Software reliability, and CBSR is Component-Based Software Reliability.

On the basis of a review of the literature, reliability factors identified by Diwaker and Tomar [59] and discussed the factors related to CBSE by Tyagi and Sharma [60]; some important factors have been identified that were used in estimating reliability in many reliability models. These factors can also help in continuing research for software reliability as well as CBSR estimation applications. This survey explained the definition of each factor and justifies its relation along with the reliability of software application. CBSR application lying on the following factors: (a) Reusability (b) Operational profile (c) Portability (d) Functionality (e) Failure rate (f) Application complexity (g) Component dependency (h) Flexibility (i) Repair rate (j) Fault (k) Mean time before failure (l) Security (m) Failure probability of each component (n) Error propagation probability of each component.

Table 1 Assessment of Soft Computing Techniques used for Prediction of Software Reliability

S. No.	Reference/ Year	Techniques Used	Aim	Factors Considered	Area of Research
1	*Chen and Wang* [1], 2002	NN	Estimating the failures of software system during maintenance phase	Mean average percentage error	Failure time.
2	*Bai* [2], 2005	BN	Modeling software reliability through an operational profile	Markov bayesian network, beta distributions	SR
3	*Mahadevan and Rebba* [17], 2005	BN	Validation of Reliability	The posterior ratio and prior densities at the predicted values	SR
4	*Pai and Hong* [19], 2006	SVM	Usage of SVM SA	Structural risks	SR
5	*Fenton et al.* [22], 2008	BN	Forecasting software reliability and defects.	Failure rate	SR
6	*Kiran and Ravi* [21], 2008	NN	Ensemble models to calculate software reliability	Threshold-accepting-based neural network, back-propagation-neural-network, pi–Sigma-network, multiple-linear-regression, Tree-Ne, multivariate-adaptive-regression-splines, dynamic-evolving-NFIS, generalized-regression	SR
7	*Elish and Elish* [20], 2008	SVM	Estimating defect-prone modules and analysis performed with machine learning models	NASA data sets	SR
8	*Doguc and Marque* [21], 2009	Bayesian Network (BN)	Estimating SR	K2 algorithm	SR
9	*Becerra et al.* [15], 2009	Differential Evolution	Resolving constrained optimization problems	Parameter type, bounds values, number of branches in source code and number of critical conditions.	Constrained optimization
10	*Zheng* [2], 2009	NN	Models for calculating software reliability such as parametric and non-parametric.	Mean rule, median rule, and weighted mean law.	SR
11	*Aljahdali and El-Telbany* [8], 2009	GA	Multi-objective in support of SRP by estimating the faults	Share normalized RMSE and correlation coefficient	SRP
12	*Singh and Kumar* [3], 2010	NN	Software reliability prediction based on feedforward NN	Execution time, cumulative failures, variable-term-predictability, RMSE, mean average error	SRP
13	*Aljahdali and Sheta* [24], 2011	Fuzzy Logic	SRGM using fuzzy logic	Execution based on Takagi-Sugeno technique	SR
14	*Moura et al.* [25], 2011	SVM	Regression through SVM	Time-series and time-to-failure data of components	SR
15	*Singh and Toora* [26], 2011	Neuro-Fuzzy	Component Classification	Percentage average error	CBSR
16	*Rotshtein* [12], 2012	Chaos Theory	Applying fuzzy logic to chaos theory	Chaotic behavior failure possibility	SR
17	*Bisi and Goyal* [4], 2012	NN	Estimating accumulative number of failures	Average relative error, RMSE, and relative RMS	SR
18	*Arora and Choudhary* [5], 2013	NN	Testing and debugging data from several software projects	Execution time and faults.	SRG prediction
19	*Nasar and Johri* [16], 2013	Differential Evolution	Distributing total testing resource optimally underneath the dynamic condition	Population size, number of generation, crossover constant, differentiation constant	Constrained optimization
20	*Ziauddin et al.* [28], 2013	Fuzzy Logic	Utilizing fuzzy logic to enhance the exactness of software effort estimation.	Effort multipliers, scale factors, one size of KDLOC, effort.	Software effort estimation

Table 1 (continued)

S. No.	Reference/ Year	Techniques Used	Aim	Factors Considered	Area of Research
21	*Rana and Yadav* [27], 2013	Fuzzy Logic	Fuzzy based software based software quality estimation approach	Vector risk, subcategory risk, aggregate risk.	software quality
22	*Yang et al.* [29], 2013	BN	Solve large-scale difficult software system	Single Train network Control & Management System	SR
23	*Shanmugam and Florence* [32], 2013	ACO	Check accuracy of software reliability model	Probability, failure rate, number of intervals, fitness function, ants, time interval	SR
24	*Roopa and Reddy* [33], 2013	PSO, GA	Generate design choices in design phase	Fitness value and computational time of PSO and GA for each iteration	SR
25	*Khosla and Soni* [30], 2013	ACO, ABC	Control the deviation in pendulum angle and velocity using ABC and ACO.	Pendulum position, angle, velocity, integrated control	SR
26	*Elloumi et al.* [31], 2013	PSO, ACO	Solving Traveling Salesman Problem (TSP) using Fuzzy ACO and Fuzzy PSO	Size of population, path length and execution time	TSP
27	*Quin* [13], 2014	Chaos theory	Analyzing reliability modeling	Software failure, relative error and mean error	SR
28	*Tyagi and Sharma* [36], 2014	ANFIS	Estimating CBSR	RMSE	CBSR
29	*Okutan and Yıldız* [35], 2014	BN	Applied interaction among metrics and defects on multiple data sets.	Numbers of developer and source code quality.	SR
30	*Lal and Kumar* [37], 2014	FIS	Fuzzy logic	Fuzzy Rules	CBSR
31	*Tyagi and Sharma* [38], 2014	ACO	Apply enhanced ACO to estimate CBSR	Heuristic component dependency graphs, Heuristic information, number of components, average Execution, component time, component path probability, pheromone amount	CBSR
32	Singh et al. [34], 2014	Fuzzy Logic	Assess the reusability using fuzzy logic.	Modularity, maintainability, flexibility, interface complexity, adaptability, membership functions	
33	*Jaiswal and Giri* [40], 2015	FIS and ANFIS	Estimated CBSR	Software quality, Fault density, Together with functionality, Usability, Availability, Performance, Serviceability, Capability, Installability and Maintainability	CBSR
34	*Kim et al.* [9], 2015	GA	Real-Valued GA	Fitness function, selection operators, heuristic crossover and non-uniform mutation	SR
35	*Diwaker and Tomar* [42], 2016	PSO	Appraisal of PSO with CBSE metrics	Component interface complexity metric, component functionality metric and average execution time.	CBSE
36	*Bolisetty and Yalla* [43], 2016	PSO– Cuckoo Search Algorithm	PSO–Cuckoo Search algorithm for building an adaptive software design	Functional requirements, evaluation, responsibility, modifiability, efficiency, and traceability	SR
37	*Diwaker and Tomar* [41], 2016	ACO	Assessment of ACO	Component efficiency, component dependency and component density, reusable components, component interaction	CBSR
38	*Kumar et al.* [6], 2016	NN	Employed distinctive data sets to NN	Relative error, absolute mean error, average bias, normalized average error, and rank metric.	Parameter consideration

Table 1 (continued)

S. No.	Reference/ Year	Techniques Used	Aim	Factors Considered	Area of Research
39	*Ramasamy and Lakshmanan* [7], 2016	NN	Simple log power SRGM with exponential testing effort function	Akaike Information Criterion, Coefficient of determination and Root Mean Squared Error	SRGM
40	*Rizvi et al.* [47], 2016	Fuzzy Logic	Designing process of quantifying software reliability	Conceptualization, Identification, Association, Quantification, Corroboration, Analysis, Assessment and amendment, Packaging.	SR
41	*Preethi and Rajan* [45], 2016	SVM	Path testing designed for compound CBSS.	Component failures	CBSR
42	*Rizvi et al.* [46], 2016	Fuzzy Logic	Early Stage of Reliability Estimation Model	Requirements stability, Regularity of specification and documentation, Reviews, review inspection and walkthrough, requirement defect density, requirement change request, scale of new functionality implemented, experience of requirement team, complexity of new functionality, quality of documentation inspected, development staff motivation, and requirements management	SR
43	Diwaker and Tomar [44], 2016	ACO, PSO, ABC	Evaluation of PSO, ABC, and ACO utilizing CBSE reusability metrics	LOC, number of function, reusable components in coding	CBSE
44	*Sharma et al.* [10], 2016	GA	Generating test data by group of methods	Individual, population, search space, chromosome, Trait, allele, locus, and genome.	Generating test data
45	*Fazel* [11], 2016	GA	Predicting software error	Number of GA generation, population requirement of education and testing, and changes in the composition operator	Predicting software error
46	*Tong et al.* [14], 2017	Chaos theory	Predicting reliability using chaotic time series and heterogeneous ensemble learning	Fitness function, RMSE, and average relative error	SR
47	*Dubey and Jasra* [48], 2017	ANFIS	Predicting reliability	Mamdani FIS	CBSR
48	*Sharam and Gandhi* [49]	MNFIS	Estimation of reliability	Component dependency, Operational profile, Reusability and Fault Density	CBSR

On the basis of assessment of various techniques, it has been noticed that PSO and Fuzzy logic can be used for future research for reliability prediction due to providing results with minimum error but in a small space. ACO, GA, NN, and SVM provide good results but these techniques consume less reusability in perspective of fuzzy and PSO and can be used for optimizing problem in large space. The fuzzy logic uses a rule-based approach that can be used to assess any reliability models for a particular set of considered parameters for that reliability model.

Conclusion

Predicting software and CBSR is challenging task and it is also a popular research area. In this paper, the survey of various soft computing techniques has been analyzed with various parameter considerations to emphasize the future aspects of estimation software reliability. After reviewing the literature of applying soft computing techniques for predicting reliability, it is concluded that CBSE becomes more popular as it uses fewer efforts and skills to make new software. The soft computing techniques like PSO and fuzzy logic may be utilized where response fast and output can be considered with less percentage errors. ACO may be used where shortest path's length is computed. These techniques may also help in estimating CBSR. In future, these techniques may be used for building a new model and help in predicting software reliability with the utilization of factors like component interaction, component dependency, complexity, failure rate and reusability etc. this survey helps the researcher for predicting reliability of any kind of tool or instrument that can be hardware based or software based.

References

1. Wang, W. L., Chen, M. H., Heterogeneous software reliability modeling. 13th International Symposium on Software Reliability Engineering(ISSRE): 41–52, 2003.
2. Zheng, J., Predicting software reliability with neural network ensembles. *Expert Syst. Appl.* 36(2):2116–2122, 2009.
3. Singh, Y., and Kumar, P., Application of feed-forward neural networks for software reliability prediction. *ACM SIGSOFT Softw. Eng. Notes* 35(5):1–6, 2010.
4. Bisi, M., and Goyal, N. K., Software reliability prediction using neural network with encoded input. *Int. J. Comput. Appl.* 47(22): 46–52, 2012.
5. Arora, M., and Choudhary, S., Software reliability prediction using neural network. *Int. J. Softw. Web Sci.* 5(2):88–92, 2013.
6. Kumar, D., Kansal, Y., Kapur, P. K., Integrating Neural Networks with Software Reliability. 3rd International Conference on Computing for Sustainable Global Development (INDIACom): 4072–4077, 2016.
7. Ramasamy, S., and Lakshmanan, I., Application of artificial neural network for software reliability growth modeling with testing effort. *Indian J. Sci. Technol.* 9(29):1–7, 2016.
8. Aljahdali, S. H., El-Telbany, M. E, Software reliability prediction using multi-objective genetic algorithm. International Conference on Computer Systems and Applications(AICCSA 2009:293–300, 2009.
9. Kim, T., Lee, K., and Baik, J., An effective approach to estimating the parameters of software reliability growth models using a real-valued genetic algorithm. *J. Syst. Softw.* 102:134–144, 2015.
10. Sharma, A., Rishon, P., and Aggarwal, A., Software testing using genetic algorithms. *Int. J. Comput. Sci. Eng. Surv* 7(2):21–33, 2016.
11. Fazel, F. S., A new method to predict the software fault using improved genetic algorithm. *Bull. la Société R. des Sci. Liège* 85:187–202, 2016.
12. Rotshtein, A., Katielnikov, D., Pustylnik, L., Reliability modeling and optimization using fuzzy logic and chaos theory. Int. J. Qual. Stat. Reliab.: 1–10, 2012.
13. Qian, L., Methodology on qualitative simulation modeling of software reliability based on chaos theory. 5th IEEE International Conference on Software Engineering and Service Science (ICSESS): pp. 99–104, 2014.
14. Tong, H., Han, R., Liu, B., Xu, B., Software reliability prediction using chaos theory and heterogeneous ensemble learning. Risk, Reliab. Saf. Innov. Theory Pract.:2396–2402, 2017.
15. Becerra, R. L., Sagarna, R., Yao, X. An evaluation of differential evolution in software test data generation, IEEE Congress on in Evolutionary Computation(CEC'09): 2850–2857, 2009.
16. Nasar, M., Johri, P., and Chanda, U., A differential evolution approach for software testing effort allocation. *J. Ind. Intell. Inf.* 1(2): 111–115, 2013.
17. Mahadevan, S., and Rebba, R., Validation of reliability computational models using bayes networks. *Reliab. Eng. Syst. Saf.* 87(2): 223–232, 2005.
18. Bai, C. G., Bayesian network-based software reliability prediction with an operational profile. *J. Syst. Soft.* 77(2):103–112, 2005.
19. Pai, P. F., and Hong, W. C., Software reliability forecasting by support vector machines with simulated annealing algorithms. *J. Syst. Softw.* 79(6):747–755, 2006.
20. Elish, K. O., and Elish, M. O., Predicting defect-prone software modules using support vector machines. *J. Syst. Softw.* 81(5):649–660, 2008.
21. Kiran, N. R., and Ravi, V., Software reliability prediction by soft computing techniques. *J. Syst. Softw.* 81(4):576–583, 2008.
22. Fenton, N., Neil, M., and Marquez, D., Using bayesian networks to predict software defects and reliability. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* 222(4):701–712, 2008.
23. Doguc, O., and Ramirez-Marquez, J. E., A generic method for estimating system reliability using bayesian networks. *Reliab. Eng. Syst. Saf.* 94(2):542–550, 2009.
24. Aljahdali, S., Sheta, A. F., Predicting the Reliability of Software Systems Using Fuzzy Logic," 8th International Conference on Information Technology: New Generations (ITNG):36–40, 2011.
25. Moura, M. D. C., Zio, E., Lins, I. D., and Droguett, E., Failure and reliability prediction by support vector machines regression of time series data. *Reliab. Eng. Syst. Saf.* 96(11):1527–1534.
26. Singh, H., and Toora, V. K., Neuro fuzzy logic model for component-based software engineering. *Int. J. Eng. Sci.* 1:303–314, 2011.
27. Rana, S., and Yadav, R. K., A fuzzy improved association mining approach to estimate software quality. *Int. J. Comput. Sci. Mob. e Comput.* 2(6):116–122, 2013.
28. Ziauddin, N., Khan, S., and Nasir, J. A., A fuzzy logic based software cost estimation model. *Int. J. Softw. Eng. It's Appl.* 7(2):7–18, 2013.

29. Yang, S., Lu, M., Ge, L., Bayesian network-based software reliability prediction by dynamic simulation. 7th IEEE International Conference on Software Security and Reliability (SERE):13–20, 2013.
30. Khosla, A., and Soni, L. G. M. K., Comparison of ABC and ant colony algorithm based fuzzy controller for an inverted pendulum. *Int. J. Innov. Technol. Explore. Eng.* 3(2):123–234, 2013.
31. Elloumi, W., Baklouti, N., Abraham, A., Alimi, A. M., Hybridization of fuzzy PSO and fuzzy ACO applied to TSP, 13th International Conference on Hybrid Intelligent Systems (HIS): 105–110, 2013.
32. Shanmugam, L., and Florence, L., Enhancement and comparison of ant colony optimization for software reliability models. *J. Comput. Sci.* 9(9):1232–1240, 2013.
33. Roopa, Y. M., and Reddy, A. R. M., Particle swarm optimization approach for component-based software architecture. *Int. J.* 3(12): 557–561, 2013.
34. Singh, C., Pratap, A., Singhal, A., An estimation of software reusability using fuzzy logic technique. International Conference on Signal Propagation and Computer Technology (ICSPCT):250–256, 2014.
35. Okutan, A., and Yıldız, O. T., Software defect prediction using bayesian networks. *Empir. Softw. Eng.* 19(1):154–181, 2014.
36. Tyagi, K., and Sharma, A., An adaptive neuro-fuzzy model for estimating the reliability of component-based software systems. *Appl. Comput. Informatics* 10(1):38–51, 2014.
37. Lal, R., and Kumar, N., Design and analysis of reliability for component-based software system by using soft computing approaches. *Int. J. Emerg. Technol. Adv. Eng.* 4(6):929–932, 2014.
38. Tyagi, K., and Sharma, A., A heuristic model for estimating component-based software system reliability using ant colony optimization. *World Appl. Sci. J.* 31(11):1983–1991, 2014.
39. Diwaker, C., Rani, S., Tomar, P., Metrics used in component-based software engineering. IJITKM Spec. Issue: 46–50, 2014.
40. Jaiswal, G. P., and Giri, R. N., Software reliability estimation of component-based software system using fuzzy logic. *Int. J. Comput. Sci. Inf. Secure.* 127(7):16–20, 2015.
41. Diwaker, C., and Tomar, P., Assessment of Ant Colony using Component-Based Software Engineering Metrics. *Indian J. Sci. Technol.* 9(44):1–5, 2016.
42. Diwaker, C., Tomar, P., Optimization and appraisal of PSO for CBS using CBSE metrics. 3rd International Conference on Computing for Sustainable Global Development (INDIACom:1024–1028, 2016.
43. Bolisetty, P. K., and Yalla, P., An efficient component-based software architecture model using hybrid PSO–CS algorithm. *Int. J. Intelligen t Eng. Syst.* 9(3):46–52, 2016.
44. Diwaker, C., and Tomar, P., Evaluation of swarm optimization techniques using CBSE reusability metrics. IJCTA 2(22):189–197, 2016.
45. Preethi, W., Rajan, M. R. B., Survey on Different Strategies for Software Reliability Prediction. International Conference on Circuit, Power and Computing Technologies (ICCPCT): 1–3, 2016.
46. Rizvi, S. W. A., Khan, R. A., and Singh, V. K., Software reliability prediction using fuzzy inference system: early stage perspective. *Int. J. Comput. Appl.* 145(10):16–23, 2016.
47. Rizvi, S. W. A., Singh, V. K., and Khan, R. A., Fuzzy logic based software reliability quantification framework: early stage perspective (FL SRQF). *Procedia Comput. Sci.* 89:359–368, 2016.
48. Dubey, S. K., Jasra, B., Reliability assessment of component-based software systems using fuzzy and ANFIS techniques. Int. J. Syst. Assur. Eng. Manag. 8(2):1319–1326, 2017.
49. Sharma, R. K., and Gandhi, P., Estimate reliability of component-based software sys-tem using modified neuro-fuzzy model. *Int. J. Eng. Technol.* 6(2):45–49, 2017.
50. Hermawanto, D., Genetic algorithm for solving simple mathematical equality problem. 1–9, 2013.
51. Albertos, P., Sala, A., Fuzzy logic controllers: advantages and drawbacks. *8th International Congress of Automatic Control*, 3: 833–844, 1998.
52. Awodele, O., Jegede, O., Neural networks and its application in engineering, Proceedings of Informing Science & IT Education Conference (InSITE):83–95, 2009.
53. Haykin, S. S., Neural Networks and Learning Machines. NJ: Pearson Upper Saddle River, 2009.
54. Tu, J. V., Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *J. Clin. Epidemiol.* 49(11):1225–1231, 1996.
55. Das, S., Abraham, A., Konar, A., Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. *Adv. Comput. Intell. Ind. Syst*: 1–38, 2008.
56. Karamizadeh, S., Abdullah, S. M., Halimi, M., Shayan, J., Javad, R. M., Advantage and Drawback of Support Vector Machine Functionality. International Conference on Computer, Communications, and Control Technology (I4CT: 63–65, 2014.
57. Katiyar, S., Ibraheem, N., and Ansari, A. Q., Ant colony optimization: a tutorial review. *MR Int. J. Eng. Technol.* 7(2):35–41, 2015.
58. Karaboga, D., and Akay, B., A comparative study of artificial bee colony algorithm. *Appl. Math. Comput.* 214(1):108–132, 2009.
59. Diwaker, C., and Tomar, P., Identification of factors and techniques to design and develop component-based reliability model. *Int. J. Sci. Res. Comput. Sci. Eng.* 5(3):107–114, 2017.
60. Tyagi, K., and Sharma, A., Significant factors for reliability estimation of component-based software systems. *J. Softw. Eng. Appl.* 7(11):934–943, 2014.