IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 8 Issue 3, March 2021
ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72
www.ijiset.com

Impact of Soft Computing on Software Reliability: A Survey of the Literature

Barka Ndahi [1], Opeyemi Abisoye[2], Sulaimon Bashir[2], Hamazat Aliyu[2]

[1]Department of Mathematical Sciences, University of Maiduguri,
Maiduguri, Borno State, Nigeria

[2]Department of Computer Science, Federal University of Technology,
Minna, Niger State, Nigeria

Abstract

Persistent incidences of software failure in software systems are a great concern in the industry. Soft computing algorithms for software reliability were found to be plagued with less accuracy and efficiency. This study answers the question regarding correlation between lack of optimal soft computing model and software failures in systems. The purpose of this study is to investigate the connection between soft computing with software reliability and the incidences of software failure. Comparative analysis was carried out on the existing methods for improving software reliability in relationship to soft computing from the literatures from 2010 – 2020 in order to justify their strengths and gaps for relevant improvement. Further studies are needed to establish a model that works optimally and prevent software failures.

Keywords: *soft computing, software reliability, computational intelligence, machine learning, software engineering, computer science.*

1. Introduction

Soft computing is a form of techniques and algorithms that deals with situations where there are uncertainties partial truth and ambiguity and helps in forecasting, optimizing and decision making in real life situations [1]. Soft computing includes, but is not limited to, Fuzzy Sets (FZ), Rough Set Theory (RSS), Artificial Neural Network (ANN), Genetic Algorithm (GA), Bayesian Networks (BN), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and others while others are hybrids [2]. Soft computing is used in software engineering to improve software reliability [3].

Soft computing is employed in creating models to predict how faulty a software system is likely to be and based on the knowledge find improvement. Some models work better than others based on criteria and both the good models and the poor ones could be greatly enhanced to boost productivity. Software reliability is very important because quite a number of systems nowadays rely on software and for them to be resilient and reliable, it is of great necessity. It is important to minimize failure as much as possible and soft computing techniques can be used to achieve this.

Optimization is usually started with defining some kind of loss/cost function and ends with minimizing it using one or the other optimization routine. The choice of optimization algorithm can make a difference between getting a good accuracy in hours or days. The applications of optimization are limitless and are widely researched topics in industry as well as academia. Some popular optimization techniques are gradient descent based and non-gradient descent based such as genetic algorithm, swarm algorithm and simulated annealing approach [4].

Software reliability can be defined as the probability of failure-free software operation for a specified period of time in a specified environment. It was noted that whenever a functional unit fails to perform its defined function it is termed as failure and a methodology to analyze these failures are termed as software reliability model [5]. The benefits of implementing software reliability as a part of software development process include the following: software reliability is used for data preservation; it helps to avoid software failure; straightforward in the system upgrade process; and lastly, system efficiency and higher performance gives greater productivity.

Reliability metrics are used to quantitatively express the reliability of the software product. The choice of which metric to adopt depends upon the type of system to which it applies and the requirements of the application domain [6]. Measuring the software reliability is a difficult problem as well as most of the aspects related to software reliability. Even the software sizes have no uniform definition. If measuring the reliability directly is a daunting challenge, something can be measured

that reflects the characteristics related to reliability. Some reliability metrics which can be used to quantify the reliability of the software product are:

- MEAN TIME TO FAILURE (MTTF) is defined as the time interval between the successive failures.
- MEAN TIME TO REPAIR (MTTR): once the failure occur sometime is required to fix the error.
- MEAN TIME BETWEEN FAILURE (MTBF) is MTTF + MTTR.
- RATE OF OCCURRENCE OF FAILURE (ROCOF) is the number of failures occurring in unit time interval.
- PROBABILITY OF FAILURE ON DEMAND (POFOD) is defined as the probability that the system will fail when a service is requested.
- AVAILABILITY (AVAIL) Availability is the probability that the system is available for use at a given time.

1.1 Soft Computing Models

Models come in variations such as Artificial Neural Network, Bayesian Network, Fuzzy Logic, Chaos Theory, Evolutionary Computing, Support Vector Machine and some are Hybrids – combination of models. Optimization algorithms are used in minimizing loss functions to get desired output. Some methods include gradient descent, particle swarm optimization, firefly algorithm, genetic algorithm and many more. Combining models and optimization techniques optimally gives better and more capable models. When constructing models like Artificial Neural Networks, the number of neural nets and how they are linked could impact positively and at times even negatively. Using the right number of mini-batches and learning rate contributes in giving models that are efficient and gives output at a rate that is desired with a high level of accuracy.

Soft computing algorithms for software reliability are plagued with less accuracy and efficiency [7] and no single model is universal to all the situations [8]. Recent studies reported that [7] used soft computing algorithm for software reliability and assessment. Results demonstrate that the proposed Particle Swarm Optimization (PSO) algorithm for training the Neural Network to predict software reliability showed that different types of evolutionary computation techniques other than PSO could also be employed to obtain a better and faster learning in estimating and predicting software reliability.

A combined method was proposed to predict software fault using combination of neural network and naive bayes algorithm [9]. Results show that the constructed model has higher prediction accuracy and prediction precision than the other methods. Combination of other learning algorithms to build an efficient prediction model is recommended. Various techniques for soft computing and software reliability improvement approaches have been employed over time to find faults in software systems and various results were arrived at. Results show that creating resource (energy, cost, time) efficient models which output optimal results are highly recommended [10]–[14] and the use of good datasets abreast with advances in technology to validate models [15]. Metrics for reliability should be used to prove and improve robustness [10], [12]; necessary parameters harnessed to achieve results [16]–[18] and experimental results ought to confirm the efficient performance of models [12], [19]. A combination of these ideas could help in producing a robust soft computing model. Ideas:

- Combination of other learning algorithms to build a prediction model is recommended
- Resource (energy, cost, time) efficient models which output optimal (accurate, fast) results are highly recommended
- Using good and up to date datasets and benchmark datasets abreast with advances in computing technology especially in security sector to validate models is highly recommended.

Software reliability is a key part in software quality. Assumptions and abstractions must be made to simplify the problem. Software reliability modeling has matured to the point that meaningful results can be obtained by applying suitable models to the problem.

1.2 Research Questions

Now, the main question that comes to mind is what is the solution? Researchers have different opinions on the current situation and to date are working hard to provide the alternative which could work in the current situation and help the world have a reliable software reliability model.

In the course of this review, the following research questions were asked:

1. What are the analysis of the research published on the challenges experienced related to soft computing and software reliability?
2. What are the results of the search related to the literature that allowed us to understand the series of challenges we face in current times and how they persisted in one form or the other?
3. Which classification/identification is the best possible solution under different scenarios as an outcome of the current situation?
4. Which lessons were learned from the past after mapping literature and what are the possible future scenario and agenda?

2. Methodology

2.1 Literature search

The preliminary search term was comprehensively evaluated to identify the most suitable search terms. Based on the stated objectives, the following terms were used to search the relevant literature in the established academic databases: “soft computing” AND “software reliability” and secondly “computational intelligence” AND “software reliability”. All articles were identified and retrieved from academic online databases via ScienceDirect, IEEE Xplore™, ACM Digital Library and SAGE Journals and total is 223; additional records identified through other sources is 50.

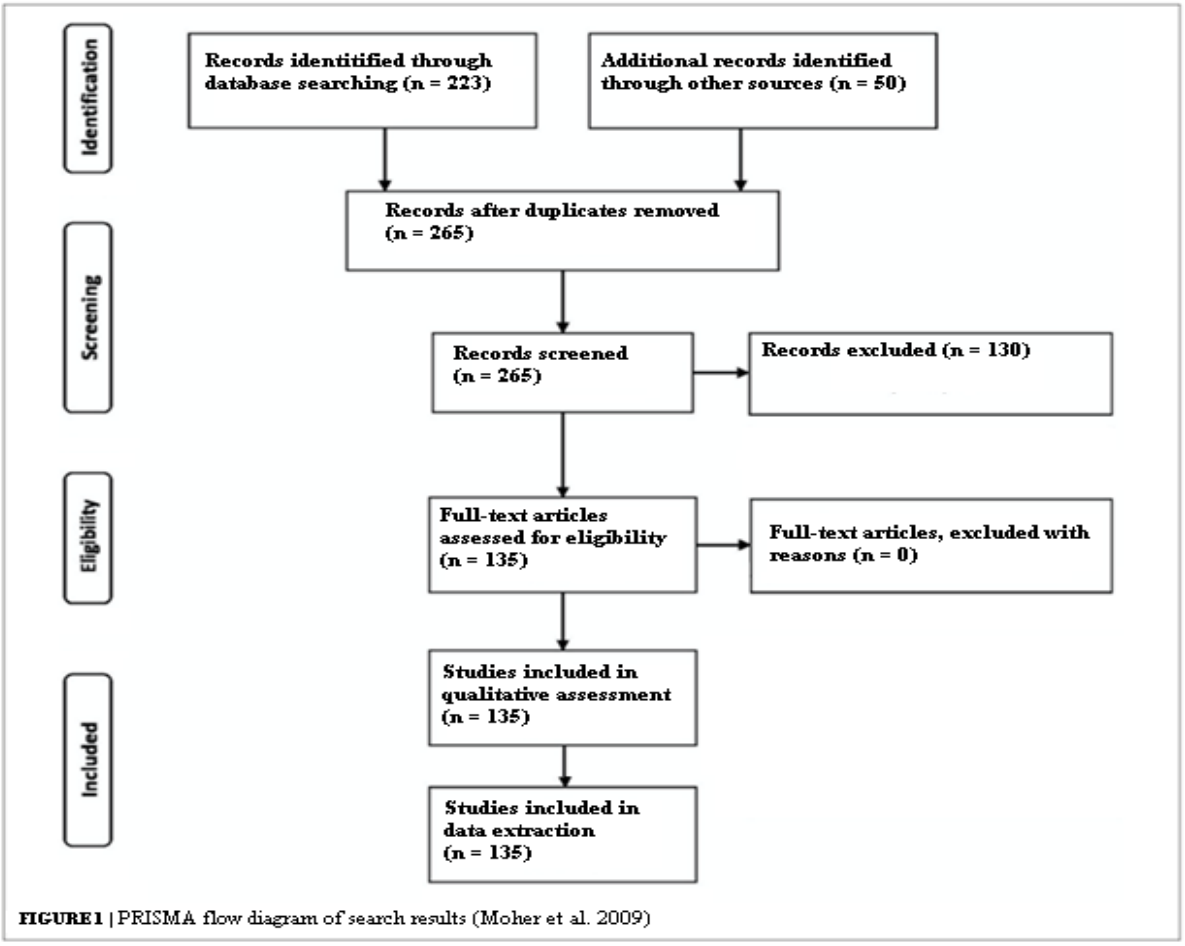

Fig. 1 Search procedure and study selection diagram showing the literature inclusion and exclusion at every stage. (PRISMA statement)

Then search strategy was applied at this stage. The references that are found to match the search terms proposed in our study were scanned to identify studies cited in the articles selected for inclusion in the study. A screening was done by sorting the relevant titles related to the objectives of this paper following the example PRISMA Statement [20]. The screening was based on the abstracts of the selected papers.

The papers that fulfilled the screening criteria were used in our study. Data extraction was done on the sorted papers and subsequently tabulated and Fig. 1 is created. The articles that were returned and identified from the online databases amounted to a total of 265.

3. Results and interpretations

3.1 Year

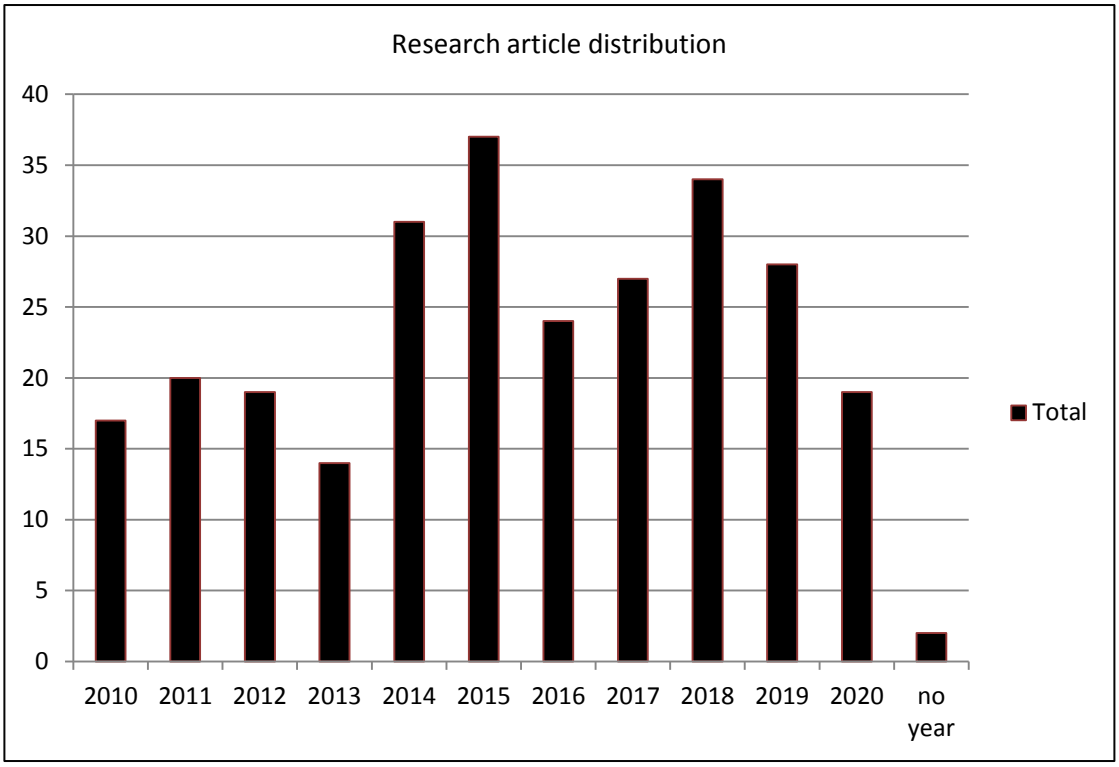

Fig. 2 Research article distribution from 2010 to 2020

This analysis as seen in Fig. 2 reveals that research article distribution was averagely low between 2010 and 2013 then spiked up in 2014 and reached its highest peak in 2015, reduced in 2016 and gradually rose to its second peak in 2018 then gradually descended through 2019 to 2020. This indicates that research in this area has risen and fallen over the years.

3.2 Distribution based on Journal

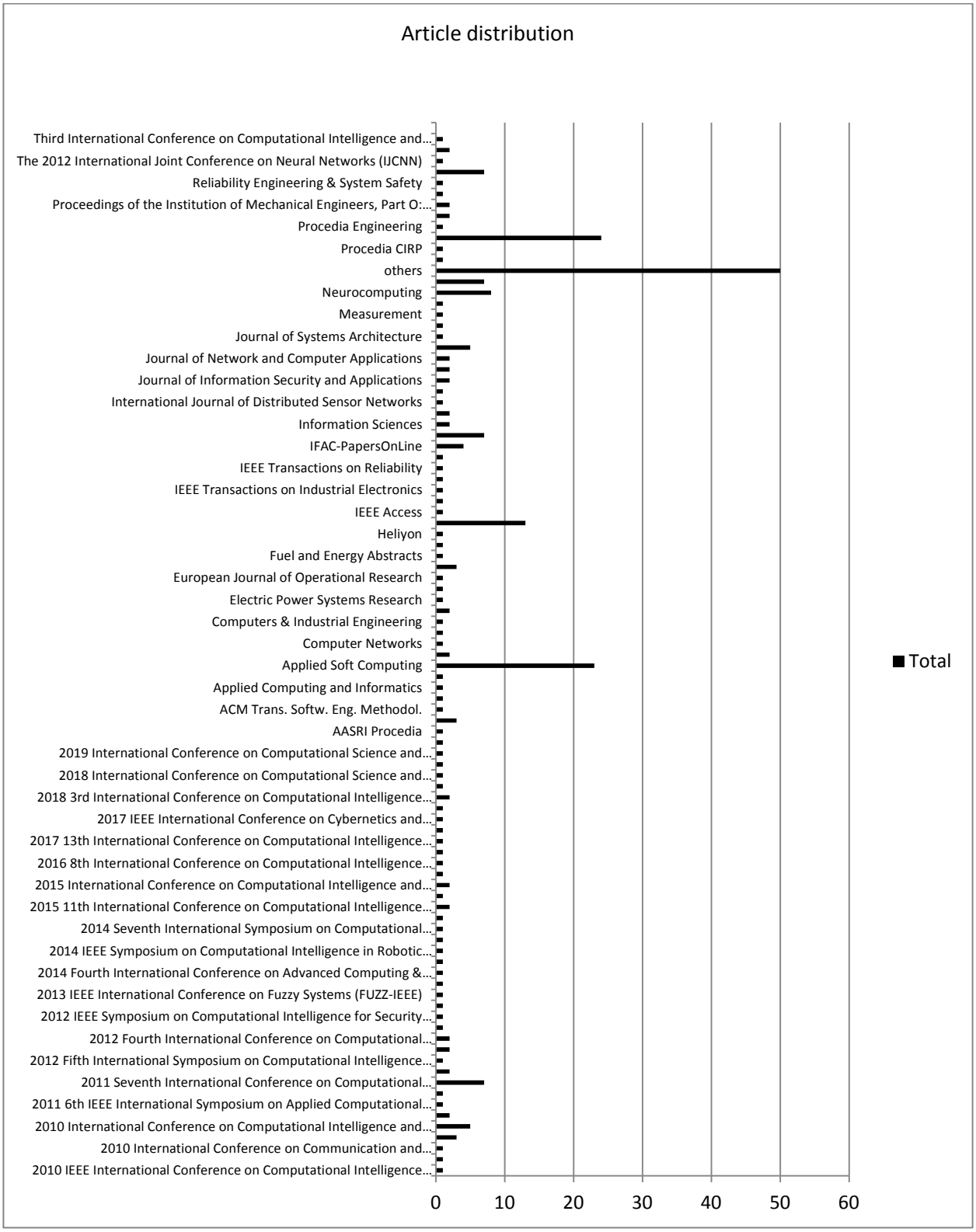

Fig. 3 Article distribution based on journal from 2010 to 2020

Article distributions in these journals reveal that ‘others' had the highest number of publications followed by Procedia Computer Science then Applied Soft Computing with IEEE Access at the fourth place. This indicates that Procedia Computer Science journal gets the second most distributed articles in this subject domain over the years than Applied Soft Computing and the rest.

3.3 Journal popularity comparison

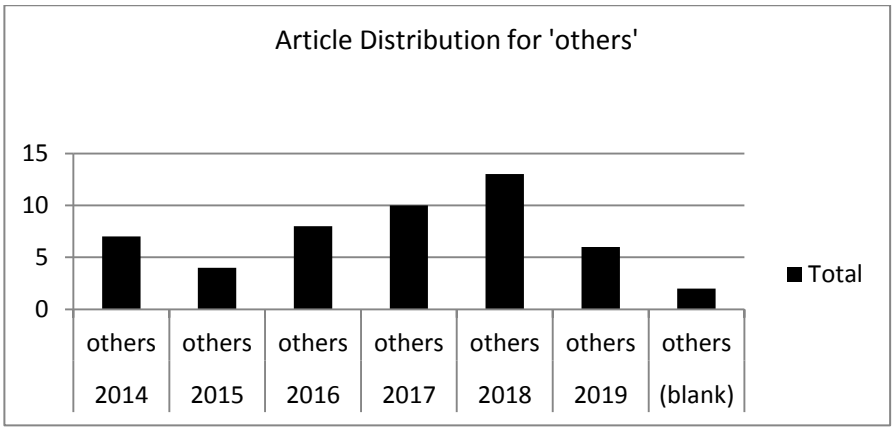

Fig. 4 Article distribution for ‘others’ from 2010 to 2020

Journal popularity comparison for ‘others’ showed that in 2018 distribution was at its peak then it was at its lowest in 2015. Fig. 4 has the summary.

Procedia Computer Science on the other hand showed 2015 had highest distribution while 2013, 2017 and 2019 had the lowest. Fig. 5 demonstrates this summary.

Generally, ‘others’ still had more article distribution over the years.

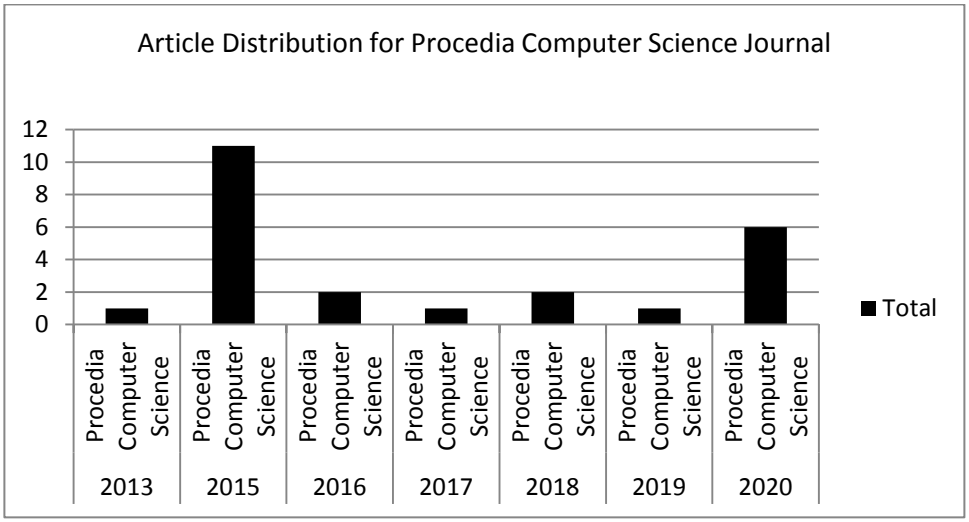

Fig. 5 Article distribution for Procedia Computer Science Journal from 2010 to 2020

3.4 Taxonomy of Soft Computing Approaches to Software Reliability

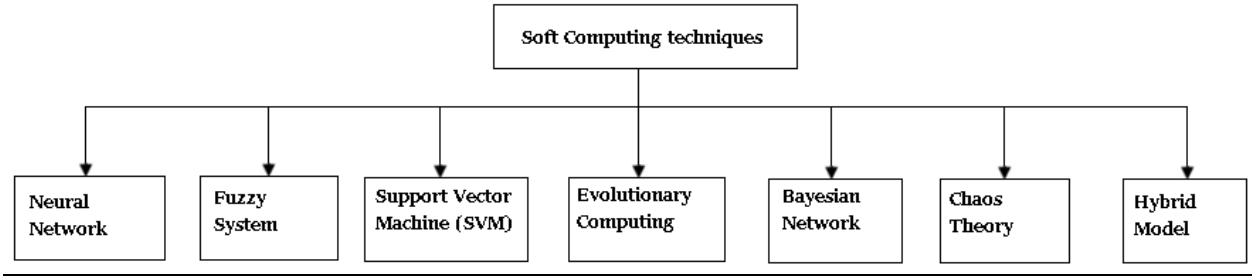

Fig. 6 Soft Computing approaches to Software Reliability

4. Conclusions

The concept of using soft computing to improve software reliability [21], [22] kept being refined with new methods and approaches. Attempts were made to make robust models sometimes based on various combinations of algorithms, optimization techniques, benchmark datasets, efficiency and accuracy from research to research varied with most meeting up to desired expectations [15], [23], [18], [24]. Various systems were considered including those on the web [24], security, database, component level reliability models, system level software reliability models with the help of soft computing techniques such as Artificial Neural Networks, Fuzzy Logic, Support Vector Machines, Evolutionary computation, Bayesian Network, Chaos theory and via Hybrid methods [25], [26]. Creating energy, cost, resource and time effective models which give optimal results were highly recommended [10]–[14]. Using good datasets to validate models also recommended.

Fault prone module; Software Operational Profile (SOP) [27] technique was used in some works and Component Based Software System (CBSS) [13]. Software Reliability Growth Model (SRGM) saw approaches from different angles using varied algorithms [18], [19]. Metrics associated with reliability were employed to prove and improve robustness [10], [12]. Necessary parameters were used and utilized as required to achieve results [16], [17], 18]. Experimental results were used to prove the efficient performance of models [12], [19].

4.1 Highlighting limitations

The limitation of the available research is that not all models proved robust, efficient, resource-effective and were highly accurate in prediction and capable of covering all scenarios giving optimal results. Additionally, some datasets and benchmark datasets became obsolete in relationship to advances in computing technology especially in security sector.

4.2 Identifying gaps

The following are the summary of gaps found in most of the survey articles.

- Some papers suggested that certain learning algorithms were yet to be combined to build prediction models
- Some analysis suggested that resource (energy, cost, time) efficient models which output optimal (accurate, fast) results ought to be built
- Some paper focuses on datasets and benchmark datasets. Up to date datasets and benchmark datasets abreast with advances in computing technology were not used in some work.

5. Future work

A model that is robust, efficient, cost/resource effective, highly accurate in prediction giving optimal results (including speed) and this model would have its performance proven on metrics, up to date datasets (benchmark) and advances in computing will be developed.

References

[1] S. A. Burney, S. M. Ali, and S. Burney, "A survey of soft computing applications for decision making in supply chain management", In IEEE 3rd International Conference on Engineering Technologies and Social Sciences, 2017, pp. 1–6, https://doi.org/10.1109/ICETSS.2017.8324158.

[2] A. Iftikhar, S. Musa, M. Alam, M. M. Su'ud, and S. M. Ali, "Application of Soft Computing Techniques in Global Software Development: state-of-the-art Review," International Journal of Engineering & Technology, vol. 7, No. 4.15, 2018, pp. 304–310, DOI: 10.14419/ijet.v7i4.15.23015.

[3] P. Dhavakumar, S. Shankar, P. M. Vikram, "Soft computing techniques for enhancing software reliability", International Journal of Latest Trends in Engineering and Technology, 2018 April, pp. 133-140, https://www.ijltet.org.

[4] Ravindra, "Demystifying Optimizations for machine learning", 2018, https://towardsdatascience.com/demystifying-optimizations-for-machine-learning-c6c6405d3eea.

[5] 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, Vol. 5, No. 3, 2014, pp. 399-406.

[6] G. Kaur, and K. Bahl, "Software Reliability, Metrics, Reliability Improvement Using Agile Process", IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1, No. 3, 2014 May.

[7] P. Roy, G. S. Mahapatra, and K. N. Dey, "An Efficient Particle Swarm Optimization-Based Neural Network Approach for Software Reliability Assessment", International Journal of Reliability, Quality and Safety Engineering, Vol. 24, No. 4, DOI: 10.1142/S021853931750019X.

[8] B. Kotaiah and R. A. Khan, "A Survey on Software Reliability Assessment by Using Different Machine Learning Techniques," Int. J. Sci. Eng. Res., 2012.

[9] B. Arasteh, "Software Fault-Prediction using Combination of Neural Network and Naive Bayes Algorithm", Journal of Networking Technology, Vol. 9, No. 3, pp. 94-101, DOI: 10.6025/jnt/2018/9/3/94-101.

[10] S. Chatterjee and B. Maji, "A Mahalanobis distance based algorithm for assigning rank to the predicted fault prone software modules," Appl. Soft Comput., vol. 70, 2018, pp. 764–772, doi: https://doi.org/10.1016/j.asoc.2018.06.032.

[11] F. Colace, V. Loia, and S. Tomasiello, "Revising recurrent neural networks from a granular perspective," Appl. Soft Comput., vol. 82, 2019, p. 105535, doi: https://doi.org/10.1016/j.asoc.2019.105535.

[12] B. K. Dewangan, A. Agarwal, M. Venkatadri, and A. Pasricha, "Self-characteristics based Energy-Efficient Resource Scheduling for Cloud," Procedia Comput. Sci., vol. 152, 2019, pp. 204–211, doi: https://doi.org/10.1016/j.procs.2019.05.044.

[13] S. Kaliraj and A. Bharathi, "Path testing based reliability analysis framework of component based software system," Measurement, vol. 144, 2019, pp. 20–32, doi: https://doi.org/10.1016/j.measurement.2018.11.086.

[14] R. Kaur, S. Arora, P. C. Jha, and S. Madan, "Fuzzy Multi-criteria Approach for Component Selection of Fault Tolerant Software System under Consensus Recovery Block Scheme," Procedia Comput. Sci., vol. 45, 2015, pp. 842–851, doi: https://doi.org/10.1016/j.procs.2015.03.169.

[15] A. I. Abubakar, H. Chiroma, S. A. Muaz, and L. B. Ila, "A Review of the Advances in Cyber Security Benchmark Datasets for Evaluating Data-Driven Based Intrusion Detection Systems," Procedia Comput. Sci., vol. 62, 2015, pp. 221–227, doi: https://doi.org/10.1016/j.procs.2015.08.443.

[16] N. R. Barraza, "A Parametric Empirical Bayes Model to Predict Software Reliability Growth," Procedia Comput. Sci., vol. 62, 2015, pp. 360–369, doi: https://doi.org/10.1016/j.procs.2015.08.416.

[17] C. Jin and S.-W. Jin, "Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization," Appl. Soft Comput., vol. 40, 2016, pp. 283–291, doi: https://doi.org/10.1016/j.asoc.2015.11.041.

[18] I. Lakshmanan and S. Ramasamy, "An Artificial Neural-Network Approach to Software Reliability Growth Modeling," Procedia Comput. Sci., vol. 57, 2015, pp. 695–702, doi: https://doi.org/10.1016/j.procs.2015.07.450.

[19] P. Roy, G. S. Mahapatra, and K. N. Dey, "Neuro-genetic approach on logistic model based software reliability prediction," Expert Syst. Appl., vol. 42, no. 10, 2015, pp. 4709–4718, doi: https://doi.org/10.1016/j.eswa.2015.01.043.

[20] Moher D, Liberati A, Tetzlaff J, Altman D., "PRISMA 2009 Flow Diagram," The PRISMA statement, 2009.

[21] H. Xiao, M. Cao, and R. Peng, "Artificial neural network based software fault detection and correction prediction models considering testing effort," Appl. Soft Comput., vol. 94, 2020, p. 106491, doi: https://doi.org/10.1016/j.asoc.2020.106491.

[22] C. Yue, "Picture fuzzy normalized projection and extended VIKOR approach to software reliability assessment," Appl. Soft Comput., vol. 88, 2020, p. 106056, doi: https://doi.org/10.1016/j.asoc.2019.106056.

[23] B. Efe, "An integrated fuzzy multi criteria group decision making approach for ERP system selection," Appl. Soft Comput., vol. 38, 2016, pp. 106–117, doi: https://doi.org/10.1016/j.asoc.2015.09.037.

[24] N. Padhy, R. P. Singh, and S. C. Satapathy, "Software reusability metrics estimation: Algorithms, models and optimization techniques," Comput. Electr. Eng., vol. 69, 2018, pp. 653–668, doi: https://doi.org/10.1016/j.compeleceng.2017.11.022.

[25] P. Rani and G. S. Mahapatra, "A novel approach of NPSO on dynamic weighted NHPP model for software reliability analysis with additional fault introduction parameter," Heliyon, vol. 5, no. 7, 2019, p. e02082, doi: https://doi.org/10.1016/j.heliyon.2019.e02082.

[26] Y. N. Rustambekovich, S. M. Gulyamov, N. B. Usmanova, and D. A. Mirzaev, "Challenging the ways to determine the faults in software: Technique based on associative interconnections," Procedia Comput. Sci., vol. 120, 2017, pp. 641–648, doi: https://doi.org/10.1016/j.procs.2017.11.290.

[27] Amrita and D. K. Yadav, "A Novel Method for Allocating Software Test Cases," Procedia Comput. Sci., vol. 57, 2015, pp. 131–138, doi: https://doi.org/10.1016/j.procs.2015.07.389.

Barka Ndahi studied B.Eng&Tech and M.Eng in Information Science and Computers in Russia between 2007 to 2013. Worked with Dept. of Mass Education in Abuja, Sharon Education and Training in Abuja and currently working with Dept. of Mathematical Sciences in University of Maiduguri, Nigeria. Research interests lies in Soft Computing, Software Reliability and Computer Science. A member of Nigeria Computer Society.

Opeyemi Abisoye studied BSc and MSc in Ilorin. A doctor in the area of Computer Science currently working with Federal University of Technology, Minna, Nigeria.

Sulaimon Bashir completed PhD program in Scotland. Head of Department of Computer Science in Federal University of Technology, Minna, Nigeria.