World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:3, 2015

A Neuro-Fuzzy Approach Based Voting Scheme for Fault Tolerant Systems Using Artificial Bee Colony Training

D. Uma Devi, P. Seetha Ramaiah

***Abstract*—**Voting algorithms are extensively used to make decisions in fault tolerant systems where each redundant module gives inconsistent outputs. Popular voting algorithms include majority voting, weighted voting, and inexact majority voters. Each of these techniques suffers from scenarios where agreements do not exist for the given voter inputs. This has been successfully overcome in literature using fuzzy theory. Our previous work concentrated on a neuro-fuzzy algorithm where training using the neuro system substantially improved the prediction result of the voting system. Weight training of Neural Network is sub-optimal. This study proposes to optimize the weights of the Neural Network using Artificial Bee Colony algorithm. Experimental results show the proposed system improves the decision making of the voting algorithms.

***Keywords*—**Voting algorithms, Fault tolerance, Fault masking, Neuro-Fuzzy System (NFS), Artificial Bee Colony (ABC)

I. Introduction

FAULT-TOLERANCE is designed by placing redundant components which duplicate the original module's functions [1]. A fault is isolated and safe operation guaranteed by replacing the problematic module with a normal module in a specific interval [2]. Pedal movement or force applied to it is measured by a sensor. The digitized information is transmitted to 4 independent brake modules, one at each wheel, through a network [3]. So, a pedal sensor's fault redundancy is critical to a vehicle's safety [4].

Usually, a hardware redundancy system, which adds extra hardware with same functions of the original hardware, is classified as static redundancy, dynamic redundancy, and hybrid redundancy based on its architecture and function. Fig. 1 shows a static redundancy system with multiple parallel modules which needs a voter to determine its final output. The voter uses majority or average rule as a fault masking algorithm to isolate a faulty input. But, static redundancy costs more as it requires three parallel modules for majority voter. It is hard to detect faults when two or more modules are faulty [5].

A faulty system for any reason can cause damage during processing some task. Tasks on real-time distributed system must be feasible, reliable, and scalable. Real-time distributed system like nuclear systems, air traffic control systems, robotics, and grids are highly dependent on deadline [6]. A fault, in real-time distributed system, results in system failure if not detected and repaired on time. These systems should function with high availability even when there are hardware or software faults.

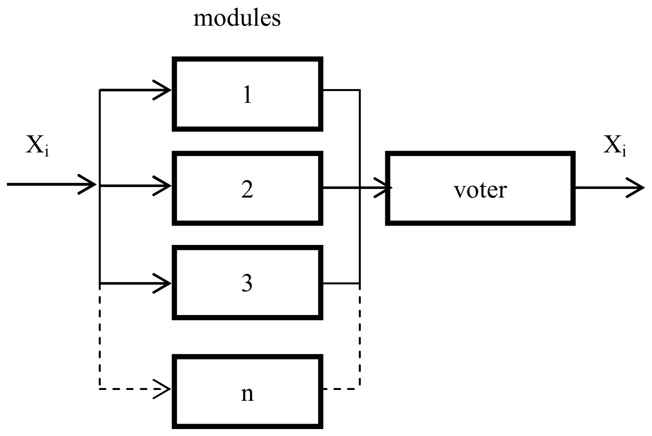

Fig. 1 Static redundancy

Fault-tolerance is important to maintain system dependability. Hardware and software redundancy are effective methods. Hardware fault-tolerance is achieved by applying extra hardware like resource (memory, I/O device), processors, communication links, whereas in software fault tolerance tasks; to handle faults, messages are added to a system. Fault must be detected by applying a fault detector followed by a recovery technique [7].

Fault tolerance increases the dependability of a system. To mask faults (or to switch to an alternate module when errors are detected) and to provide service despite faults [8] are the objectives of a fault-tolerant system. Fault tolerant systems must provide their specified services despite the occurrence of faults in the system's components. Fault tolerance requires the consideration of the issues of:

1. Error Detection,
2. Damage Assessment,
3. Error Recovery,
4. Continued Service

In fault masking, software versions or hardware modules are replicated, and then voting is used to settle among their results, to mask the effect of one or more run-time errors. N-modular redundancy and N-version programming are the well-known fault masking methods. The simplest form of N-

Uma Devi D is with Andhra University, Andhra Pradesh, India (e-mail: umadevi.odm@gmail.com).
P. Seetha Ramaiah is with Andhra University, Andhra Pradesh, India.

modular redundancy approach is Triple Modular Redundancy (TMR).

TMR is a fault tolerant method using three modules or sensors that work together and get the same input signal [9]. The output of TMR is one of the three outputs chosen with voting techniques. Redundant sensors to improve the system safety can be realized by various methods. The simplest method uses two sensors, one primary sensor, and a backup sensor.

Voting is an important operation in realizing ultra-reliable systems based on a multi-channel computation paradigm. Voting is needed even if multiple computation channels have redundant hardware units, diverse program modules executed on same basic hardware, identical hardware and software with diverse data or other possible hardware/program/data redundancy and/or diversity combinations.

Hardware or software voting schemes are appropriate depending on data volume and voting frequency. Low-level voting with high frequency needs hardware voters while high-level voting on results of complex computations can be performed in software without performance degradation or overhead.

Voting algorithms are used to provide an error masking capability in a wide range of highly dependable commercial and research applications [10]. These applications consist of N-modular redundant hardware systems and differently designed software system which has its base in N-version programming. Depending on the application and type of voter selected the algorithms are carried out in hardware or software. To compare the performances of the voting algorithms, various performance measures can be defined (e.g. Reliability, availability, safety).

Voting algorithms can be grouped from various viewpoints. They may be classified according to [11]:

- The implementation method - Software or Hardware voters
- Type of agreement – Inexact or exact voter
- Output space cardinality size – small space or big output space
- Nature of working environment – asynchronous or synchronous voter
- Depending on the functionality of the voting algorithms—generic, hybrid and purpose-built voting.

Generic voting algorithms simply select one of the variant inputs or amalgamate them to produce a new distinct value of output. Majority voting, plurality voting, median voting or weighted average voting is generic types of voting algorithm. Optimal voting, maximum likelihood voting, predictor voting, smoothing voting or integrated voting are examples of hybrid voting.

Weighted average voting is often more trustable than a median voter since median voter simply selects the mid-value of results whereas a weighted average voter assigns weights which are measures of each input cooperation in making voter output. In case of weighted voting, weights of voting should vary among the different output classes in each classifier [12].

The weight should be high for that specific output class for which the classifier performs well. So, it is a crucial issue to choose the appropriate weights of votes for all the classes per classifier [13]. Weighting problem can be viewed as an optimization problem.

Therefore, it can be solved by taking advantage of artificial intelligence techniques such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). In this work, Artificial Bee Colony (ABC) for classification is implemented. Remaining sections of this paper is organized as follows: Section II discusses the related works in literature. Section III explains the methodology. Section IV discusses the experimental results, and Section V concludes the study.

II. Literature Survey

A new generation of average voter based on parallel algorithms was introduced by [14]. As parallel algorithms have high processing speed and are appropriate for large-scale systems, it is used to achieve an ideal parallel average voting algorithm for applications where input space is large.

A fault-tolerant control system based on majority voting with Kalman filter was presented by [15]. Familiarizing with fault tolerant systems and their requirements at the beginning, Voters, majority voting principle, and Kalman filter equations are described subsequently.

Methods for autonomic management in a voting-based data collection system to handle situations where available network bandwidth may fluctuate and/or device fault parameters change unpredictably was outlined by [16].

A distributed voting strategy for a robust NMR system was proposed by [17]. It is shown that using inexpensive current-based drivers and buffers can eliminate the centralized voter unit and ensure majority voting among N modules in a distributed fashion.

An adaptive fuzzy fault-tolerant voting mechanism was proposed by [18]. The adaptive fuzzy voting mechanism decides correct output and the output solves questions of measured noise and gyro error.

A redundancy-based fault-tolerant methodology to design a reliable, analog system was proposed by [19]. This work's contribution is an innovative analog mean voter. Results verified the concepts and measured the system's reliability when single upset transient occurs.

An extension to fuzzy voting scheme by incorporating Interval Type-2 (IT2) fuzzy logic was proposed by Linda and Manic [20]. The new voter design features robust performance when uncertainty assumptions change dynamically over time. Results demonstrated improved availability, safety, and reliability of the IT2 fuzzy voting scheme.

La-inchua et al. [21] presented a system to detect lane-blocking traffic incidents which are amongst major causes of traffic jam. The proposed system used fuzzy logic to identify traffic status as normal and abnormal. Mean speed and standard deviation of inter-arrival time were used as inputs to the Fuzzy Inference System (FIS), and then, the majority voting was applied to the outputs of FIS to improve detection rate and mean time to detection. Simulation results showed that the proposed lane-blocking detection system is very suitable for real-time implementation.

La-inchua et al. [22] presented a fuzzy logic-based traffic incident detection system to detect a lane-blocking traffic incident that usually causes of traffic congestion. The proposed system used fuzzy logic to identify traffic status as normal and abnormal. It is found that the proposed system that used Discrete Wavelet Transform (DWT) could give higher detection rate when compared with the system without DWT. Furthermore, the majority voting was also applied to the outputs of FIS in order to increase detection rate. The performance of the proposed detection system for lane-blocking traffic incidents would be shown based on the simulation results

Kwiat et al. [23] examined three binary voting algorithms used with computer replication for fault tolerance and separately observe the resultant reliability and security. Random dictator provided good security and majority rule yielded good fault tolerance. The random troika (a subset of 3 replicas) was presented as an effective combination of fault-tolerant and secure computing.

Alahmadi et al. [24] introduced a new hybrid history based voting algorithm for the proposed smart mobile e-health monitoring system. The proposed algorithm had given better and stable performance in the error range expected by the monitoring system.

Zhou and Chen [25] discussed DNS and its security extension and analyzed its lack of security and also proposed a DNS system based on Byzantine fault tolerance. With the method of distributing zone table into several DNS servers, using an improved Byzantine fault-tolerant algorithm and majority voting mechanism, system could provide services continuously, even if partial servers were faulty, and the security of DNS system were enhanced.

Namazi et al. [26] proposed a voter-less fault-tolerant strategy to implement a robust NMR system design. Using a novel logic code division multiple accesses, data could transfer with very low error rates among N modules and fully eliminated the need for a centralized voter unit. This type of dependable strategy is important for future nano-systems in which high defect rate was expected. Experimental results also verified the concept, clarified the design procedure, and measured the system's reliability.

Derasevic et al. [27] presented a fault-tolerant system architecture for control applications that add a node replication scheme with voting on top of a Flexible Time Triggered (FTT)-based system.

III. Methodology

Fuzzy control systems produce actions according to fuzzy rules based on fuzzy logic. The basic units of the fuzzy logic controller are fuzzifier, fuzzy rule base, fuzzy inference engine, and defuzzifier [28]. In fuzzifier, crisp input values are mapped to fuzzy sets using membership functions. Fuzzy rule base contains the IF-THEN rules which specify the behavior of the system. Fuzzy inference engine maps input fuzzy sets to output fuzzy sets using rule base. Defuzzifier maps the fuzzy output sets to crisp output value.

Rule-based fuzzy inference step along with centroid norm for defuzzification are used in this voter. Statistically, selecting the fuzzy parameter values, in this voter, the variation of the fuzzy parameter values is varied by the performance of the voter. A main limitation of this voter is the static selection of fuzzy threshold parameter values [29].

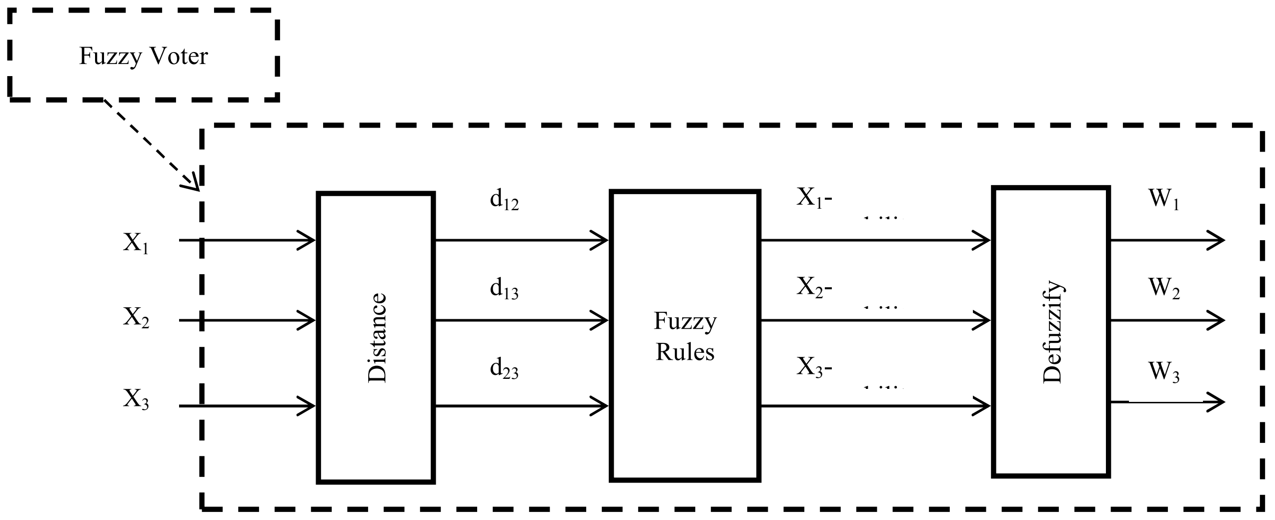

Fig. 2 Input fuzzy voter

There is a need for automatic dynamic selection of values for these parameters for any dynamically varying input dataset. The fuzzy voter computes the voter output as a weighted average and weights are determined by the fuzzy inference engine. The fuzzy voter computes the voter output as a weighted average and weights are determined by the fuzzy inference engine.

A Neural-Fuzzy System (NFS) is designed to realize the process of fuzzy reasoning, where the connection weights of the network relate to the parameters of fuzzy reasoning. Using the backpropagation type learning algorithms, the NFS can identify fuzzy rules and learn membership functions of the fuzzy reasoning. It is easy to start a one-to-one correspondence between the network and the fuzzy system [30]. The NFS architecture has distinct nodes for antecedent clauses, conjunction operators, and consequent clauses. A fuzzy control system can also be termed as an NFS.

The neuro-fuzzy system consists of parts of a conventional fuzzy system except that computation at each stage is completed by a layer of hidden neurons. The NNs learning capacity, in such system, is provided to improve the system knowledge [31]. The proposed neuro-adaptive learning technique benefits from fuzzy modeling procedure to learn from data. To compute the membership function parameters, that allow the associated FIS to track the given input/output data, it is used. The training schemes of the NNs are used for training.

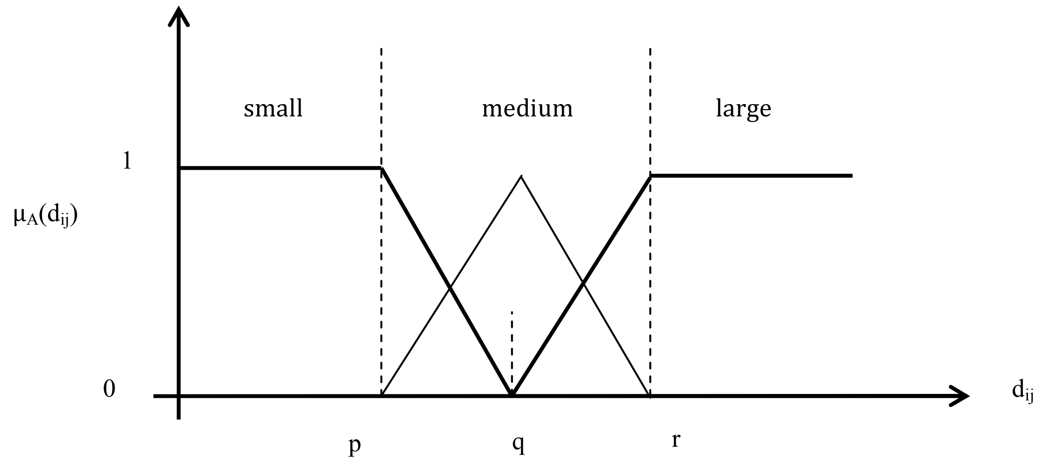

Fig. 3 Representation of membership grades μ A (dij)

The variables and fuzzy membership functions are defined as [32]:

Difference between two voter-inputs:

$$d_{ij} = |x_i - x_j|,\ i \neq j. \tag{1}$$

Symmetry: where p, q, and r are real numbers and p < q < r

$$r - q = q - p \tag{2}$$

$$\mu_{\text{small}}(d_{ij}) = \begin{cases} 1 : d_{ij} \le p, \\ \dfrac{q - d_{ij}}{(q-p)} : p < d_{ij} \le q, \\ 0 : q < d_{ij}, \end{cases} \tag{3}$$

$$\mu_{\text{medium}}(d_{ij}) = \begin{cases} 0 : d_{ij} \le p, \\ \dfrac{d_{ij} - p}{(q-p)} : p < d_{ij} \le q, \\ \dfrac{r - d_{ij}}{r-q} : q < d_{ij} < \text{r}, \\ 0 : \text{r} \le \text{d}_{ij}, \end{cases} \tag{4}$$

$$\mu_{\text{large}}(d_{ij}) = \begin{cases} 0 : d_{ij} \le q, \\ \dfrac{d_{ij} - p}{(r-q)} : \text{q} < d_{ij} \le r, \\ 1 : r < d_{ij}. \end{cases} \tag{5}$$

In an m-way voter, there are m(m − 1)/2 fuzzy difference variables. Each difference calculation results in a non-zero membership value being assigned to one or two fuzzy sets defined for that variable.

A. Artificial Bee Colony (ABC)

The ABC algorithm is a swarm based meta-heuristic algorithm based on the foraging behaviour of honey bee colonies. The model is composed of three important elements: employed and unemployed foragers and food sources [33]. The employed and unemployed foragers are the first two elements while the third element is the rich food sources close to their hive. The 2 leading modes of behaviour are also described by the model.

ABC algorithm provides a search process based on population. ABC is classified into three categories—employed bees, onlooker bees, and scout bees. In ABC algorithm, the position of a food source represents a possible solution to an optimization problem, and nectar amount of a food source represents the quality (fitness) of the solution [34]. The number of the employed bees or the onlooker bees is equal to the number of solutions.

Most neural network uses gradient descent method for training, such as backpropagation algorithm. This algorithm has defects such as long time to converge and falling into local minimum easily. ABC algorithm is a global optimization heuristics algorithm, so it can train the weights of neural network to avoid the deficiency of BP algorithm [35]. The main idea of this algorithm is: assume that the network have D parameters including D_1 weights and D_2 thresholds. First of all, the neural network parameter is set to D random non-zero values I_{pi}. Assume the total number of bees is N_s, where population size of onlooker bees is N_e, the population size of scout bees is N_u, the maximum number of iterations is $T_{\max}$, *Limit* is searching number limit.

For neural network training using ABC, there is another crucial parameter that can have a big effect on the results, namely the size of the search space, which here corresponds to the limit on the network weights. It is known that optimizing the initial random weight range for BP can have a big effect on the generalization performance, so it is not surprising that it also has a big effect for ABC too [36].

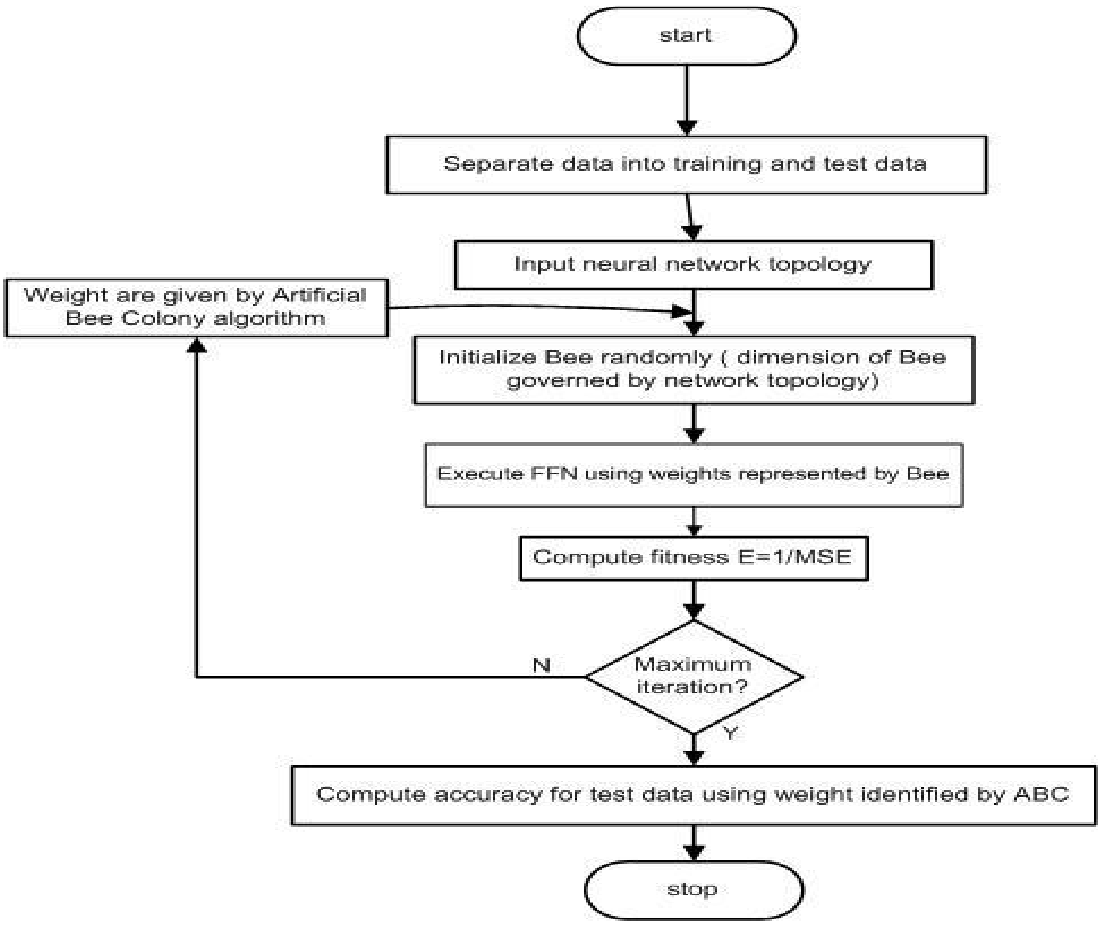
start
Separate data into training and test data
Input neural network topology
Weight are given by Artificial Bee Colony algorithm
Initialize Bee randomly (dimension of Bee governed by network topology)
Execute FFN using weights represented by Bee
Compute fitness E=1/MSE
Maximum iteration?
N
Y
Compute accuracy for test data using weight identified by ABC
stop

Fig. 4 Working of ABC –FFN

The ABC algorithm's pseudo-code is below:

```
Pseudo-code of the ABC algorithm
Initialize the population of solution $X_i$
Evaluate the population
Set cycle to 1
Repeat
Produce new solution $v_i$ in neighborhood of $x_i$ for employed bees and apply greedy selection process $x_i$ and $v_i$
Calculate probability values Pi for solution xi by means of fitness values using (6)
```

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{6}$$

```
Produce new solution $v_i$ for onlooker from solution $x_i$ selected depending on $p_i$ and evaluate them and apply greedy selection process
If an abandoned solution for scout is available, then replace it with a new solution using (7)
```

$$X_i(i) = lb_i + (ub_i - lb_i) * r \tag{7}$$

```
Memorize the best solution so far
Cycle=cycle+1
Until cycle=Max_iterations
```

$B_k = (-1)^k C_{N-1-k}(1)$

IV. Experimental Results

A. Experimental Setup

Details of software voters experimental test harness used are explained. The experimental test harness simulates a TMR system including input data generator, replicator, 3 saboteurs (to inject errors in replicated input data), voter, and a comparator. In every test cycle, the input generator produces one correct result. This simulates redundant module generated identical correct results. Notional correct result copies are presented to all saboteurs, in all cycles. Based on chosen random distributions, saboteurs are programmed to introduce module errors. In a test set, 1, 2 or 3 saboteurs are activated to simulate module result errors in voter inputs. All saboteurs' outputs are subjected to an examined voter, and voter output is compared by cycle notional correct value by comparators.

TABLE I
Safety of Voters

Error Amplitude	Majority voting	Fuzzy based	NN fuzzy based	weight optimized NN fuzzy based system
0.5	1	1	1	1
0.55	1	1	1	1
0.6	0.97	0.98	0.97	0.98
0.65	0.94	0.97	0.95	0.96
0.7	0.93	0.94	0.95	0.96
0.75	0.91	0.93	0.93	0.94
0.8	0.9	0.93	0.92	0.93
0.85	0.89	0.92	0.92	0.93
0.9	0.86	0.89	0.91	0.92
0.95	0.85	0.88	0.91	0.92
1	0.85	0.88	0.9	0.91
1.05	0.84	0.85	0.9	0.91
1.1	0.84	0.85	0.89	0.91
1.15	0.83	0.84	0.86	0.88

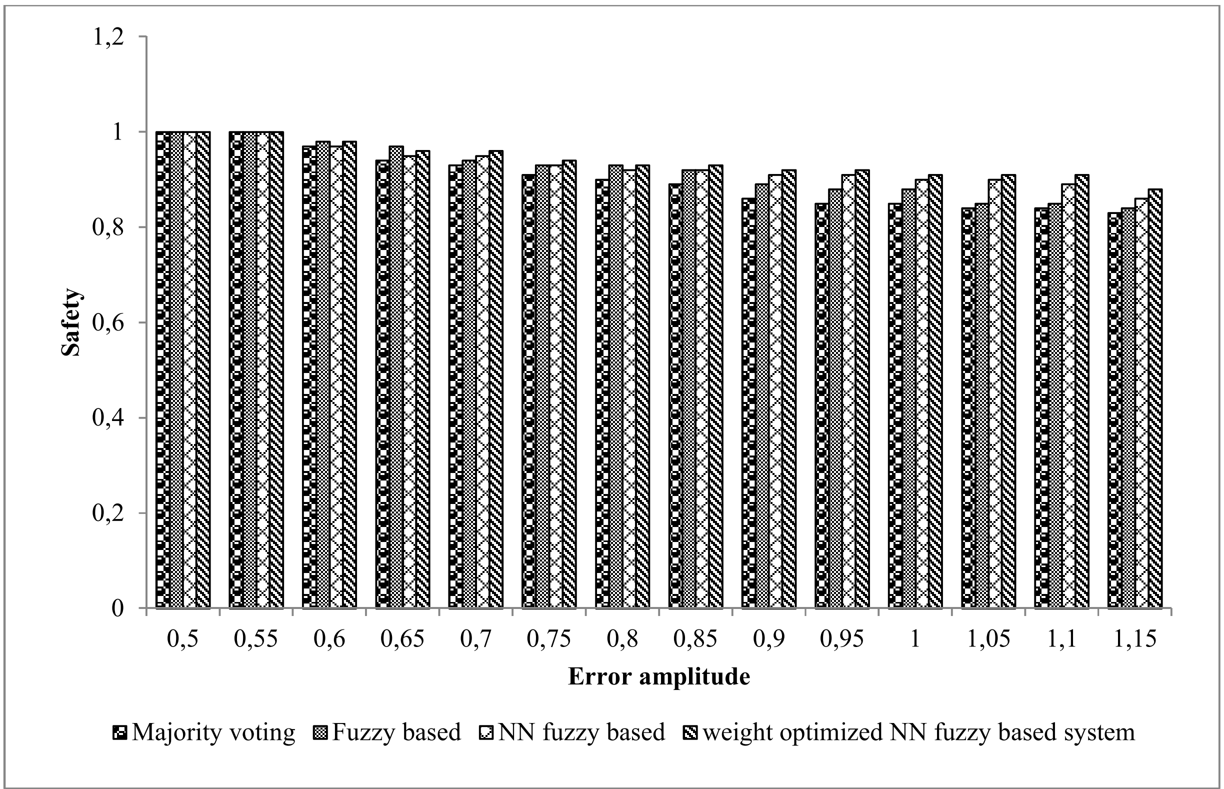

Fig. 5 Safety of Voters

From Fig. 5, the proposed weight optimized neuro-fuzzy voter shows better safety behaviour compared to the standard majority, fuzzy voters and NN fuzzy based. When compared to majority voter, the proposed weight optimized neuro-fuzzy method improves safety by 1.0256% to 8% when the error amplitude is more than 0.6. The proposed weight optimized neuro-fuzzy method improves safety by 1.0363% to 6.8182% when the error amplitude is more than 0.6 when compared to fuzzy voter. The proposed weight optimized neuro-fuzzy method improves safety by 1.0256% to 2.2989% when the error amplitude is more than 0.6 when compared to NN fuzzy voter.

From Fig. 6, the proposed weight optimized neuro-fuzzy voter achieves improved availability compared to the standard majority, fuzzy voters and NN fuzzy based. When compared to majority voter, the proposed weight optimized neuro-fuzzy method has higher availability by 5.5866% to 26.4706%. The proposed weight optimized neuro-fuzzy method improves availability by 1.105% to 18.4397% when compared to fuzzy voter. The proposed weight optimized neuro-fuzzy method improves availability by 1.0471% to 6.7114% when compared to NN fuzzy based voter.

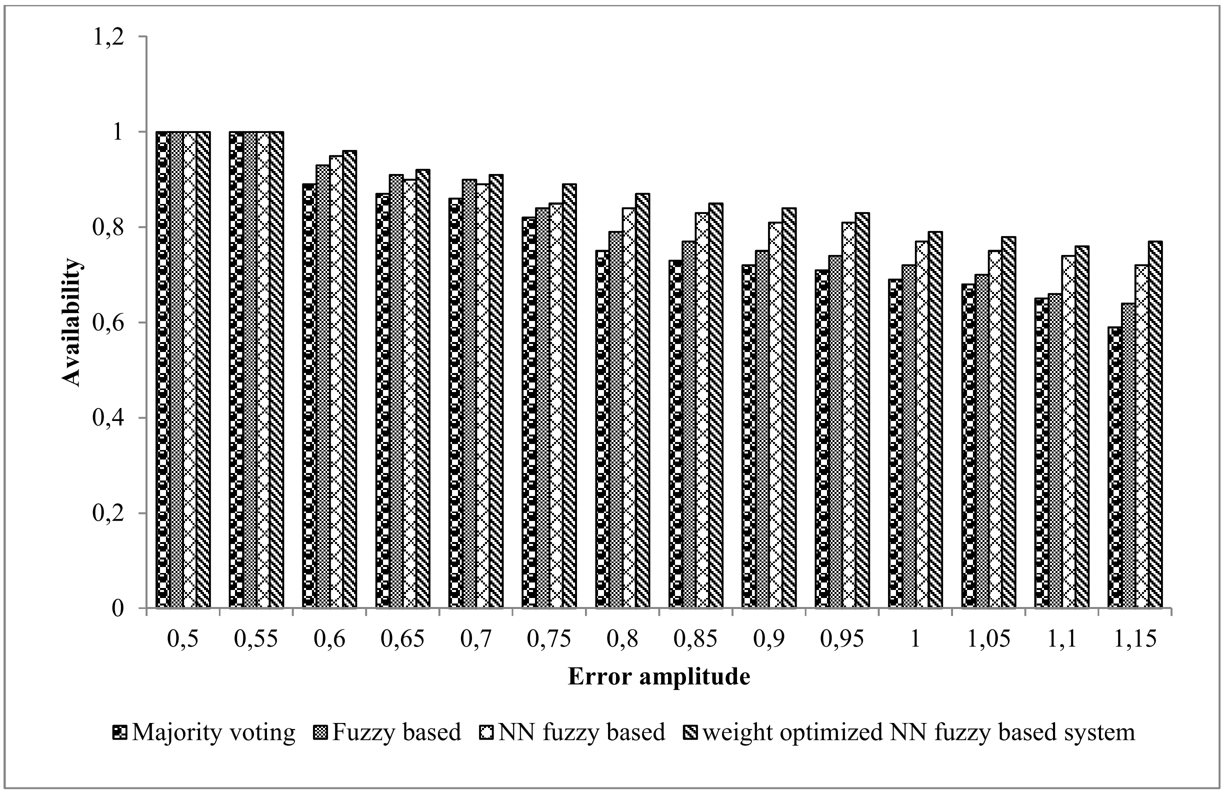

Fig. 6 Availability of Voter

TABLE II
AVAILABILITY OF VOTERS

Error Amplitude	Majority voting	Fuzzy based	NN fuzzy based	weight optimized NN fuzzy based system
0.5	1	1	1	1
0.55	1	1	1	1
0.6	0.89	0.93	0.95	0.96
0.65	0.87	0.91	0.9	0.92
0.7	0.86	0.9	0.89	0.91
0.75	0.82	0.84	0.85	0.89
0.8	0.75	0.79	0.84	0.87
0.85	0.73	0.77	0.83	0.85
0.9	0.72	0.75	0.81	0.84
0.95	0.71	0.74	0.81	0.83
1	0.69	0.72	0.77	0.79
1.05	0.68	0.7	0.75	0.78
1.1	0.65	0.66	0.74	0.76
1.15	0.59	0.64	0.72	0.77

V. CONCLUSION

The neuro-fuzzy approach, symbiotically combining the merits of connectionist and fuzzy approaches, constitutes a key component of soft computing at this phase. To date, there has been no detailed and integrated categorization of the various neuro–fuzzy models used for rule generation. The proposed neuro-fuzzy voter's performance was tested on a refined experimental harness which permitted modeling of various distributions of input signal's noise and errors. The proposed weight optimized NN fuzzy based system gives better safety behavior and improved availability when compared with majority voter, Fuzzy based voter, and NN fuzzy based voter.

REFERENCES

[1] Johnson, B. W. (1988). Design & analysis of fault tolerant digital systems. Addison-Wesley Longman Publishing Co., Inc..
[2] Latif-Shabgahi, G., Bass, J. M., & Bennett, S. (2004). A taxonomy for software voting algorithms used in safety-critical systems. Reliability, IEEE Transactions on, 53(3), 319-328.
[3] Latif-Shabgahi, G., Bennett, S., & Bass, J. M. (2003). Smoothing voter: a novel voting algorithm for handling multiple errors in fault-tolerant control systems. Microprocessors and Microsystems, 27(7), 303-313.
[4] Kwak, S. W., & You, K. H. (2004). Reliability Analysis and Fault Tolerance Strategy of TMR Real-time Control Systems. Journal of Institute of Control, Robotics and Systems, 10(8), 748-754.
[5] Kim, M. H., Lee, S., & Lee, K. C. (2008). Predictive hybrid redundancy using exponential smoothing method for safety critical systems. International Journal of Control Automation and Systems, 6(1), 126.
[6] Girault, A., Lavarenne, C., Sighireanu, M., & Sorel, Y. (2001, April). Generation of fault-tolerant static scheduling for real-time distributed embedded systems with multi-point links. In Parallel and Distributed Processing Symposium, International (Vol. 3, pp. 30125b-30125b). IEEE Computer Society.
[7] Dima, C., Girault, A., Lavarenne, C., & Sorel, Y. (2001). Off-line real-time fault-tolerant scheduling. In Parallel and Distributed Processing, 2001. Proceedings. Ninth Euromicro Workshop on (pp. 410-417). IEEE.
[8] Bala, N. International Journal of Advances In Computing And Information Technology.
[9] Latifi, Z., & Karimi, A. (2014). A TMR Genetic Voting Algorithm for Fault-tolerant Medical Robot. Procedia Computer Science, 42, 301-307.
[10] Saheb, P. B., Subbarao, K. & Dr. S.Phani kumar (2013). A Survey on Voting Algorithms Used In Safety Critical Systems
[11] Das, M. (2010). An approach towards History Based Weighted Average Voting Algorithm using Soft Dynamic Threshold (Doctoral dissertation, Jadavpur University Kolkata).
[12] Zarafshan, F., Latif-Shabgahi, G. R., & Karimi, A. (2010, July). Notice of Retraction A novel weighted voting algorithm based on neural

networks for fault-tolerant systems. In Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on (Vol. 9, pp. 135-139). IEEE.

[13] Zhang, Y., Zhang, H., Cai, J., & Yang, B. (2014, May). A Weighted Voting Classifier Based on Differential Evolution. In Abstract and Applied Analysis (Vol. 2014). Hindawi Publishing Corporation.

[14] Karimi, A., & Zarafshan, F. (2010, June). An optimal parallel average voting for fault-tolerant control systems. In Networking and Information Technology (ICNIT), 2010 International Conference on (pp. 360-363). IEEE.

[15] Danecek, V., & Silhavy, P. (2011, August). The Fault-tolerant control system based on majority voting with Kalman filter. In Telecommunications and Signal Processing (TSP), 2011 34th International Conference on (pp. 472-477). IEEE.

[16] Ravindran, K., Rabby, M., & Adiththan, A. (2013, January). Autonomic management of replication in voting-based fault-tolerant data collection systems. In Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on (pp. 1-2). IEEE.

[17] Namazi, A., & Nourani, M. (2007, October). Distributed voting for fault-tolerant nanoscale systems. In Computer Design, 2007. ICCD 2007. 25th International Conference on (pp. 568-573). IEEE.

[18] Sui, J., Hua, Z., Yang, L., Tian, Y., & Zhang, Y. (2008, June). Adaptive fuzzy fault-tolerant voting mechanism based on EKF. In Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on (pp. 740-744). IEEE.

[19] Askari, S., Dwivedi, B., Saeed, A., & Nourani, M. (2009, November). Scalable mean voting mechanism for fault tolerant analog circuits. In Design and Test Workshop (IDT), 2009 4th International (pp. 1-6). IEEE.

[20] Linda, O., & Manic, M. (2011). Interval type-2 fuzzy voter design for fault tolerant systems. Information Sciences, 181(14), 2933-2950.

[21] La-inchua, J., Chivapreecha, S., & Thajchayapong, S. (2013, May). A new system for traffic incident detection using fuzzy logic and majority voting. In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2013 10th International Conference on(pp. 1-5). IEEE.

[22] La-inchua, J., Chivapreecha, S., & Thajchayapong, S. (2014, May). Fuzzy logic-based traffic incident detection system with discrete wavelet transform. In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2014 11th International Conference on(pp. 1-6). IEEE.

[23] Kwiat, K., Taylor, A., Zwicker, W., Hill, D., Wetzonis, S., & Ren, S. (2010, November). Analysis of binary voting algorithms for use in fault-tolerant and secure computing. In Computer Engineering and Systems (ICCES), 2010 International Conference on (pp. 269-273). IEEE.

[24] Alahmadi, A., Soh, B., & Alghamdi, S. (2013, April). A hybrid history based weighted voting algorithm for smart mobile e-health monitoring systems. InIntelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on (pp. 402-407). IEEE.

[25] Zhou, W., & Chen, L. (2011, August). A Research and Design of Byzantine Fault Tolerant DNS. In Internet Technology and Applications (iTAP), 2011 International Conference on (pp. 1-4). IEEE.

[26] Namazi, A., Nourani, M., & Saquib, M. (2010). A fault-tolerant interconnect mechanism for NMR nanoarchitectures. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 18(10), 1433-1446.

[27] Derasevic, S., Proenza, J., & Barranco, M. (2014, September). Using FTT-ethernet for the coordinated dispatching of tasks and messages for node replication. In Emerging Technology and Factory Automation (ETFA), 2014 IEEE (pp. 1-4). IEEE.

[28] Öğüt, D. (2003). A behavior based robot control system using neuro-fuzzy approach (Doctoral dissertation, Middle East Technical University).

[29] Singamsetty, P., & Panchumarthy, S. (2012). Automatic fuzzy parameter selection in dynamic fuzzy voter for safety critical systems. International Journal of Fuzzy System Applications (IJFSA), 2(2), 68-90.

[30] Mitra, S., & Hayashi, Y. (2000). Neuro-fuzzy rule generation: survey in soft computing framework. Neural Networks, IEEE Transactions on, 11(3), 748-768.

[31] Alsaade, F. (2010). Neuro-Fuzzy Logic Decision in a Multimodal Biometrics Fusion System. Scientific Journal of King Faisal University (Basic and Applied Sciences), 11(2), 14.

[32] Pathak, A., Agarwal, T., & Mohan, A. (2015). A Novel Fuzzy Membership Partitioning for Improved Voting in Fault Tolerant System. Journal of Intelligent Learning Systems and Applications, 7(01), 1.

[33] Bolaji, A. L. A., Khader, A. T., Al-Betar, M. A., & Awadallah, M. A. (2013). Artificial bee colony algorithm, its variants and applications: a survey. Journal of Theoretical & Applied Information Technology, 47(2).

[34] Zhao, Z., Yang, J., Che, H., Sun, H., & Yang, H. (2013). Application of artificial bee colony algorithm to select architecture of a optimal neural network for the prediction of rolling force in hot strip rolling process. Journal of Chemical & Pharmaceutical Research, 5(9).

[35] Jin, F., & Shu, G. (2012, September). Back propagation neural network based on artificial bee colony algorithm. In Strategic Technology (IFOST), 2012 7th International Forum on (pp. 1-4). IEEE.

[36] Bullinaria, J. A., & AlYahya, K. (2014). Artificial Bee Colony training of neural networks. In Nature Inspired Cooperative Strategies for Optimization (NICSO 2013) (pp. 191-201). Springer International Publishing.

Uma Devi D is with Andhra University. She is currently pursuing his doctorate in India.

P. Seetha Ramaiah is currently working in Andhra University, India.