Vol 6 • Number 2 April - Sep 2015 pp. 227-237 **Impact Factor: 2.5** **Available at** www.csjournals.com

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

[1]D. Uma Devi, [2]P. Seetha Ramaiah
Department of CS and SE, Andhra University, Vishakapatnam, Andhra Pradesh 530003, India,
umadevi.odm@gmail.com

**Abstract:** Voting algorithms make decisions in fault tolerant systems where a redundant module provides inconsistent outputs. Popular voting algorithms are majority voting, weighted voting and inexact majority voters. Each technique suffers from areas where agreements are lacking for voter inputs. This was overcome using fuzzy theory in literature. Our earlier work concentrated on a neuro-fuzzy algorithm where training using neuro system improved the voting system's prediction result. Neural Network Weight training is sub-optimal. This paper proposes to optimize the weights of the Neural Network using Hybrid 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), Hybrid Artificial Bee Colony (HABC).

## Introduction

Fault-tolerance is designed through placement of redundant components duplicating the original module's functions [1]. A fault is isolated and safe operation guaranteed by replacing problematic module with a normal one within a specific interval [2]. Pedal movement/force applied is measured by a sensor. This digitized information is transmitted to 4 independent brake modules, one at each wheel, via a network [3]. So, a pedal sensor's fault redundancy is critical to vehicle safety [4].

A fault-tolerant system may tolerate one/more fault-types including -- i) transient, intermittent or permanent hardware faults, ii) software/hardware design errors, iii) operator errors, or iv) externally induced upsets/physical damage.

Hardware Fault-Tolerance: Most fault-tolerant designs were directed to building computers that recover from hardware components random faults automatically. Two approaches to hardware fault recovery were used: 1) fault masking, and 2) dynamic recovery.

- Fault masking: a structural redundancy technique that totally masks faults in a redundant modules set. Many identical modules execute same functions and their outputs are voted to remove errors created by a faulty module. Triple Modular Redundancy (TMR) is a fault masking form where circuitry is triplicated and voted.

- Dynamic recovery: required when only one copy of a computation runs at a time, and involves automated self-repair. As in fault masking, computing system is partitioned into modules backed by spares as protective redundancy.

Software Fault-Tolerance: Efforts to attain software that tolerate software design faults (programming errors) used static/dynamic redundancy approaches similar to those for hardware faults. One approach, N-version programming, uses static redundancy as in independently written programs (versions) that perform same functions and their outputs are voted at special checkpoints.

Voting algorithms were used in many domains like political elections, neural networks (N), sensor networks, and distributed systems. Voting algorithms, also called agreement algorithms, specify how votes are integrated leading to final results. They create consistency and mask out effects of faulty behavior. The votes are input values provided by many sources1, and final result that is either a scalar or in vector form referred to by many phrases like "agreed value", "decision value", "voted value", or simply "final decision" [5].

Voter is a critical component in implementing N-Modular Redundant systems. Voting is a hard problem in itself, for 3 reasons: i) floating point arithmetic is not exact and so voting on floating point values results in inexact voting, ii) variants output (redundant modules) may be highly sensitive to small critical region variations, e.g. around threshold values in a specification, and iii) some problems have many correct solutions (square roots) which confuse voters. Different voting strategies were introduced to handle such problems; examples are predictor voters, maximum likelihood voter, stepwise negotiation voter and word voters. Two traditional voting algorithms are majority and weighted average voters.

An inexact majority voter in its general form produces a correct output if majority of inputs match others; that is, they are within an application-specific intervals of each other. In no majority cases, voter generates an exception flag which is detected by system supervisor to drive system to a safe state. Efficient implementations of majority voter were addressed [6].

Weighted average voter, calculates weighted mean of redundant input values. It is used in applications like clock synchronization in distributed computer systems, pattern recognition and sensor planes where a result is generated in every voting cycle.

Voting algorithms are grouped from various viewpoints. They can be classified according to [7]:

- Implementation method – Software/Hardware voters
- Agreement type – Exact/Inexact voter
- Output space cardinality size – small space/large output space
- Nature of working environment – synchronous/asynchronous voter

From a functionality view point, voting algorithms are classified into 3 categories:

1. Generic voting algorithms (result generated by amalgamation or selection)
2. Hybrid voting algorithms (generic plus additional information on variants)
3. Purpose-built voters

Weighted average voting is more trusty than a median voter as median voter just selects results mid-value whereas a weighted average voter assigns weights which are measures of an input cooperation in making voter output. In weighted voting, weights of voting should vary among different output classes in a classifier [8].

Weight should be high for a specific output class for which a classifier performs well. Hence, it is crucial to choose appropriate weights of votes for all classes per classifier [9]. Weighting problem is an optimization problem.

So, it can be solved by taking advantage of artificial intelligence techniques like GA and Particle Swarm Optimization (PSO). In this study, Artificial Bee Colony (ABC) is used for classification. The remaining sections of this paper are organized as follows: Section 2 discusses related works in literature. Section 3 explains the methodology. Section 4 discusses the experimental results, and section 5 concludes the study.

**Literature Survey**

Danecek and Silhavy [10] presented a fault-tolerant control system based on majority voting with Kalman filter. Being familiar with fault tolerant systems and their requirements at the start, Voters, majority voting principle, and Kalman filter equations are described later.

A dynamic scheduling solution to achieve fault tolerance in many-core architectures was presented by Bolchini et al., [11]. Triple Modular Redundancy (TMR) was to multi-threaded application to mitigate the effects of both permanent and transient faults dynamically, and identify/isolate damaged units. The approach targets best performance and balances use of healthy resources to limit wear-out and aging, which cause permanent damage. Experimental results on synthetic case studies, to validate ability to tolerate faults while optimizing performance and resource usage were reported.

A new approach to Quantum repeaters (QRs) where quantum information is faithfully transmitted via a noisy channel without long distance teleportation, thus eliminating need to establish remote entangled links was investigated by Muralidharan et al., [12]. The approach uses small encoding blocks to fault-tolerantly correct operational and photon loss errors and describes resource requirement for QRs to generate a secure key through optimization. Numerical calculations indicated that quantum memory bits at a repeater station required to generate a secure key has favorable poly-logarithmic scaling with distance across which communication was desired.

An efficient adaptive redundant architecture which uses averaging cell (AVG) principle to improve nano scale circuits and systems reliability was introduced by Aymerich et al., [13]. The architecture, proposed an adaptive structure to cope with nonhomogeneous variability and time-varying effects like degradation and external aggressions, limiting factors in future technologies.

the Methods was developed within the MADNESS project to allow system adaptivity and fault tolerance on NoC-based MPSoCs presented by Meloni et al., [14] involved different layers of system design. The process migration mechanism, can be exploited by run-time manager to cope with permanent faults by migrating processes running on faulty processing element. A fast heuristic determined new mapping of processes to

tiles meaning that the new approach allowed system to react to faults without substantial impact on user experience.

A fault-tolerant control scheme for PWM inverter-fed induction motor-based electric vehicles was proposed by Tabbache et al., [15]. The new strategy deals with power switch (IGBTs) failures mitigation in a reconfigurable induction motor control. In a vehicular context, 4-wire and 4-leg PWM inverter topologies were investigated and performances discussed.

An approach to increase systems life implemented on SRAM-based Field Programmable Gate Arrays (FPGAs), by introduced fault tolerance properties enabling systems to autonomously manage transient/permanent faults occurrence was presented by Bolchini et al., [16]. Based on foreseen mission time and application environment, designer was supported in system implementation able to reconfigure itself, by reloading correct configuration in transient faults, or relocating part of functionality in permanent faults.

A novel technique for improved voting by adaptively varying membership boundaries of a fuzzy voter to achieve realistic consensus among redundant modules inputs of a fault tolerant system was presented by Pathak et al., [17]. It proved that the technique suggested dynamic membership partitioning reduces occurrences of incorrect voter outputs compared to fixed membership partitioning voter implementations. Simulation results for the new voter for Triple Modular Redundancy (TMR) fault tolerant system indicates that algorithm revealed better safety and availability performance compared to the current one.

Many voting algorithms were described to arbitrate results of redundant modules in fault-tolerant systems by Saheb et al., [7]. A fuzzy set theory based voting scheme which softens the inexact majority voter's harsh behavior in the neighborhood of 'voter threshold' and handles uncertainty and multiple error cases in a region defined by fuzzy input variables was proposed. A set of fuzzy rules determines one fuzzy agreeability value for an individual input describing how it matches other inputs. Automatic fuzzy parameter selection is based dynamic fuzzy voter for safety critical systems with limited system knowledge.

Methodology for design/testing of fault tolerant systems implemented in a FPGA platform with different types of diagnostic techniques was presented by Straka et al., [18]. The basic partial dynamic reconfiguration principles are described with their impact on fault tolerance features of digital design implemented in a SRAM-based FPGA. Methodology included a faulty module's detection and localization in a system and its repair and bringing system back to state where it operates correctly. The methodology was verified on ML506 development board with Virtex5 FPGA for different Register-Transfer Level (RTL) components.

Linda and Manic [19] proposed an extension to fuzzy voting scheme by incorporating Interval Type-2 (IT2) fuzzy logic. The new voter design has robust performance when uncertainty assumptions change dynamically with time. Results proved the IT2 fuzzy voting scheme's improved availability, safety and reliability.

A novel voter that integrated majority and standard weighted average voters was introduced by Latif-Shabgahi [6]. An experimental framework to evaluate software voting algorithms was provided and the proposed voter's safety/availability behaviour was studied in error scenarios. Experimental results showed that the integrated voter ensured higher availability than majority and weighted average voters in all scenarios. But, it compromised safety behaviour between majority and weighted average voters. It performed faster than the naive weighted average voter.

A novel and comprehensive framework for Reconfigurable Fault Tolerance (RFT) capable of creating and modeling FPGA-based reconfigurable architectures enabling a system to self-adapt its fault-tolerance strategy in accordance with dynamically varying fault rates was presented by Jacobs et al., [20]. The PR-based RFT hardware architecture helped many redundancy-based fault-tolerance modes (DWC, TMR) and additional fault-tolerance features (watchdog timers, check pointing, ABFT and rollback), as also a mechanism for to dynamically switch between modes. A combination of fault-tolerance features enabled use of Commercial-Off-The-Shelf (COTS) SRAM-based FPGAs in harsh environments.

A family of inexact agreement algorithms considering confidence level placed on a node and presence of malicious behavior was introduced by Azadmanesh et al., [5]. Expressions for convergence rate and these algorithms fault tolerance was developed. The effect of weights is shown when agreement process favors nodes with specific trust levels. Difficulties of applying weights to current voting algorithms are also described.

A fuzzy logic-based traffic incident detection system to detect a lane-blocking traffic incident that leads to traffic congestion was presented by La-inchua et al., [21]. The new system used fuzzy logic to identify traffic status as normal/abnormal. It found that the new system using Discrete Wavelet Transform (DWT) ensured higher detection rates compared to a system without DWT. Also, majority voting was applied to FIS outputs to increase detection rate. The proposed detection system's performance for lane-blocking traffic incidents based on the simulation results was shown.

A voter-less fault-tolerant strategy to implement a robust NMR system design was proposed by Namazi et al., [22]. Data, using a novel logic code division multiple accesses, could be transferred with very low error rates among N modules eliminating the need for a centralized voter unit. Such dependable strategy is important for future nano-systems where high defect rate is anticipated. Results verified the concept, clarified design procedure, and measured system reliability.

**Methodology**

Fault-tolerant control (FTC) prevents faults catastrophic consequences by retuning/restructuring control systems to maintain acceptable process operation despite drastic faults. Control actions are generated to satisfy process objectives using process information from sensors and manipulating available actuators. Based on diagnosed failures, the control system is retuned/reconfigured to achieve FTC. Early FTC methods relied on hardware redundancies. As industry transitioned, digital control and computer control, software redundancies and soft sensors became popular for FTC. Using robust control framework and its integration with knowledge-based systems was proposed for FTC. [23]

Fuzzy Logic Controller (FLC) embodies human-like thinking to a control system. FLC can emulate human deductive thinking, ie; a process people use to infer conclusions from what they know. FLC was primarily applied to process control through fuzzy linguistic descriptions. FLC designs controllers for plants with complex dynamics and high nonlinearity model. In a motor control system, FLC converts linguistic control rules to control strategy based on heuristic information/expert knowledge. FLC has a fixed control rules, derived from expert's knowledge. Membership function (MF) of associated input/output linguistic variables is predefined on a common discourse universe. For successful design of FLC's correct input and output scaling factors (gains) selection or tuning of other controller parameters are crucial, which are done through trial and error to achieve best control performance [24]. FLC structure is see in Figure 1 shows 4 functions, each materialized by block.

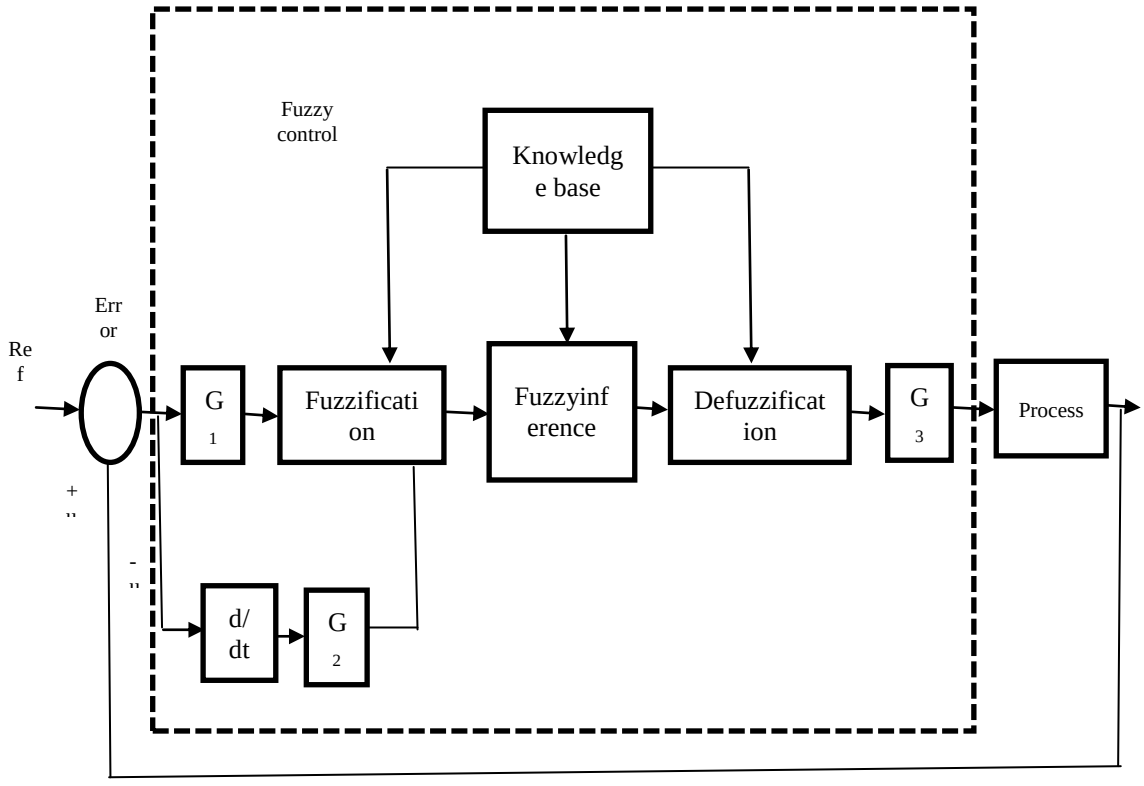


Figure 1. Structure of fuzzy controller

Fuzzy and neuro-fuzzy models for nonlinear dynamic system identification were studied, for forward/recurrent topologies. Several fuzzy and neuro-fuzzy models were developed in many applications like process modelling, identification and fault diagnosis control. NN and fuzzy systems are stand-alone systems. With increased complexity of the process being modelled, difficulty is in developing dependable fuzzy rules and membership functions [25]. Hence another approach, known as neuro-fuzzy approach was developed.

A Neural-Fuzzy System (NFS) realizes the process of fuzzy reasoning, where network connection weights relate to fuzzy reasoning parameters. Using back propagation learning algorithms, NFS identifies fuzzy rules

and learns membership functions of fuzzy reasoning. It is easy to start a one-to-one correspondence between network and fuzzy system [26]. NFS architecture has distinct nodes for antecedent clauses, conjunction operators, and consequent clauses. A fuzzy control system is also called a NFS.

NN with efficient learning algorithms were presented as an alternative to automate or support development of tuning fuzzy systems. NN and fuzzy systems combine to join advantages and cure individual ills. NN introduce computational characteristics of learning in fuzzy systems and receive in turn systems representation interpretation and clarity. Thus, disadvantages of fuzzy systems are compensated by NN capacities. These are complementary justifying their use together [27].

Types of Neuro-Fuzzy Systems: different combinations of techniques are divided, based on the following classes:

- Cooperative Neuro-Fuzzy System: a pre-processing phase where NN mechanisms of learning determine the fuzzy system's sub-blocks.
- Concurrent Neuro-Fuzzy System: In this NN and fuzzy system work continuously together.
- Hybrid Neuro-Fuzzy System: here a NN learns some parameters of fuzzy system (parameters of fuzzy sets, fuzzy rules and weights of rules) iteratively. The majority of research uses neuro-fuzzy term to refer to hybrid neuro-fuzzy system.

A. Hybrid Artificial Bee Colony (HABC)

ABC is a swarm based meta-heuristic algorithm founded on honey bee colonies foraging behaviour. The model has 3 elements: employed and unemployed foragers and food sources [28]. Employed and unemployed foragers are the first 2 elements and the third element is the rich food source close to a hive. The model also describes the 2 leading modes of behaviour.

ABC algorithm ensures a population based search process. ABC is classified into 3 categories—employed bees, onlooker bees, and scout bees. In the algorithm, food source position represents a possible solution to an optimization problem, and nectar of a food source represents solution quality (fitness) [29]. Employed bees or onlooker bees are equal to solutions.

Most NN uses gradient descent method to train like back propagation algorithm. This has defects like long time to converge and falling into local minimum. ABC, a global optimization heuristics algorithm trains NN weights to avoid BP algorithm's deficiency [30]. The algorithm assumes that the network has D parameters including D1 weights and D2 thresholds. First, NN parameter is set to D random non-zero values Ipi. Assume total bees are Ns, where onlooker bees population size is Ne, scout bees population is Nu, maximum iterations are Tmax and Limit is searching number limit.

For NN training using ABC, there is another parameter that affects results, namely search space size, which corresponds to limit on network weights. It is known that optimizing initial random weight range for BP can affect generalization performance [31].

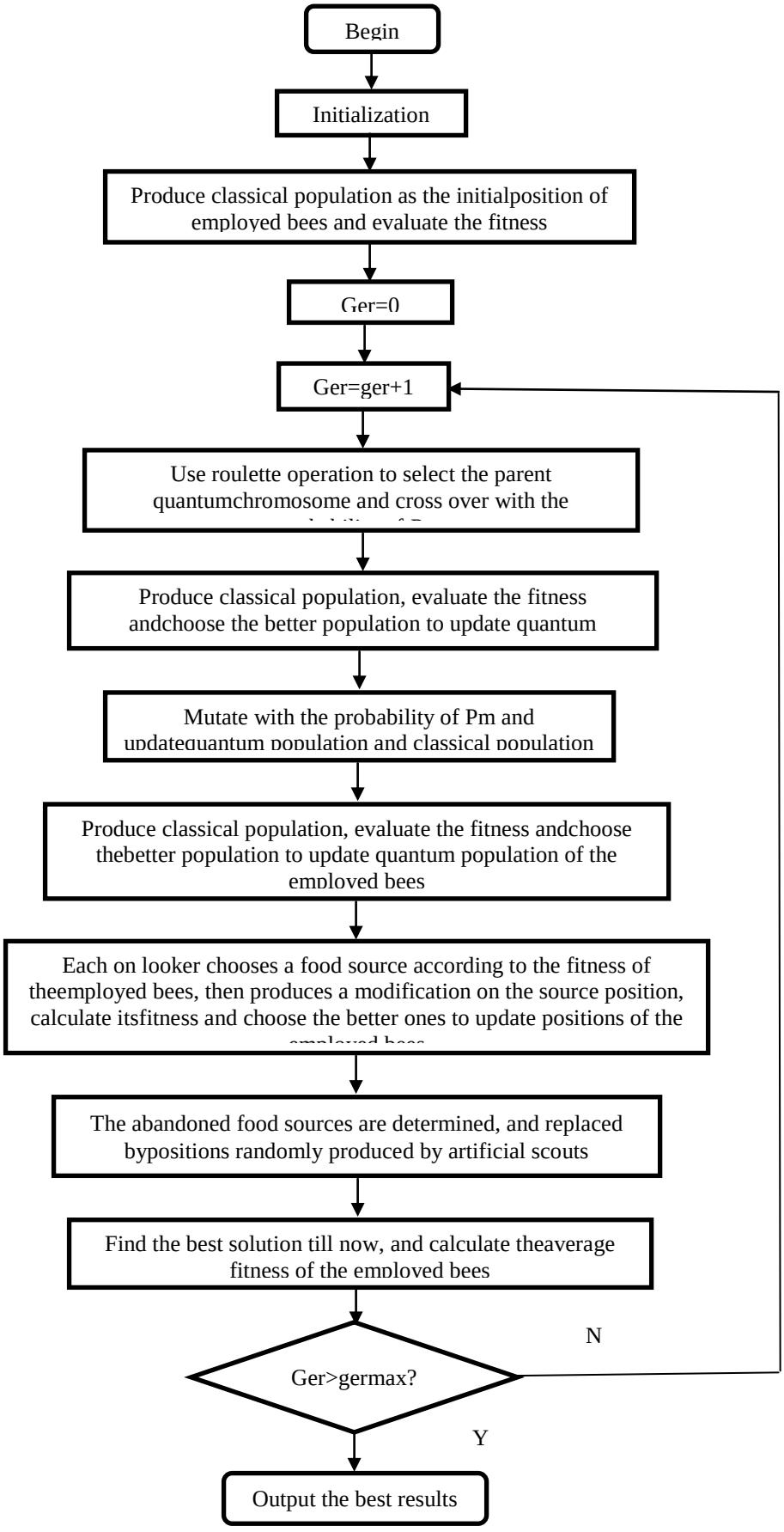


Figure 2. Flowchart of the hybrid ABC

The steps of the proposed HABC algorithm are [32]:

Step1 Initialization phase;

Step 1.1 set system parameters;

Step 1.2 Produce initial population.

Step2 Apply Pareto non-dominated sorting function on population, and update external Pareto archive set by using solutions in first Pareto level front.

Step3 If stopping criterion is satisfied, output non-dominates solutions in external Pareto archive set; otherwise, perform steps 4-7.

Step4 Employed bee phase.

Step 4.1 put each employed bee on a solution in population.

Step 4.2 for each employed bee, perform local search on assigned solution and generate a new neighbouring solution.

Step 4.3 Evaluate new neighbouring solution and record better solution among new solutions and old one as current solution and put it into population. If both solutions are non-dominated with each other, select one as current solution randomly.

Step 4.4 if a solution has not been improved through limit cycles, then corresponding employed bee becomes a scout bee and performs step 6.

Step 4.5 Evaluate solutions corresponding to employed bees, apply Pareto non-dominated sorting algorithm on new population and update external Pareto archive set using solutions in first Pareto level.

Step5 Onlooker bee phase.

Step 5.1 for each onlooker bee, randomly select 3 solutions from population and select the best as food source. If 3 solutions cannot dominate each other, then select a non-dominate solution randomly.

Step 5.2 for each onlooker bee, perform local search for selected food source and carry out greedy selection procedure to record better solution in population.

Step 5.3 Evaluate solutions corresponding to onlooker bees, apply Pareto non-dominated sorting algorithm on new population and update external Pareto archive set using solutions in first Pareto level.

Step6 Scout bee phase.

Step 6.1 Divide scout bees into 2 parts with same number of bees.

Step 6.2 Scout bees in first part randomly choose a food source and perform local search operator in predefined region. After generating a new solution, perform greedy selection procedure.

Step 6.3 Each scout bee in second part randomly select a non-dominate solution in external Pareto archive set and performs many local searches for selected solution. After generating a new solution, perform greedy selection procedure.

Step 6.4 Evaluate solutions corresponding to scout bees, apply Pareto non-dominated sorting algorithm on new population and update external Pareto archive set using solutions in first Pareto level.

Step7 go to step 3.

**Experimental Results**

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 1: Safety of voters

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

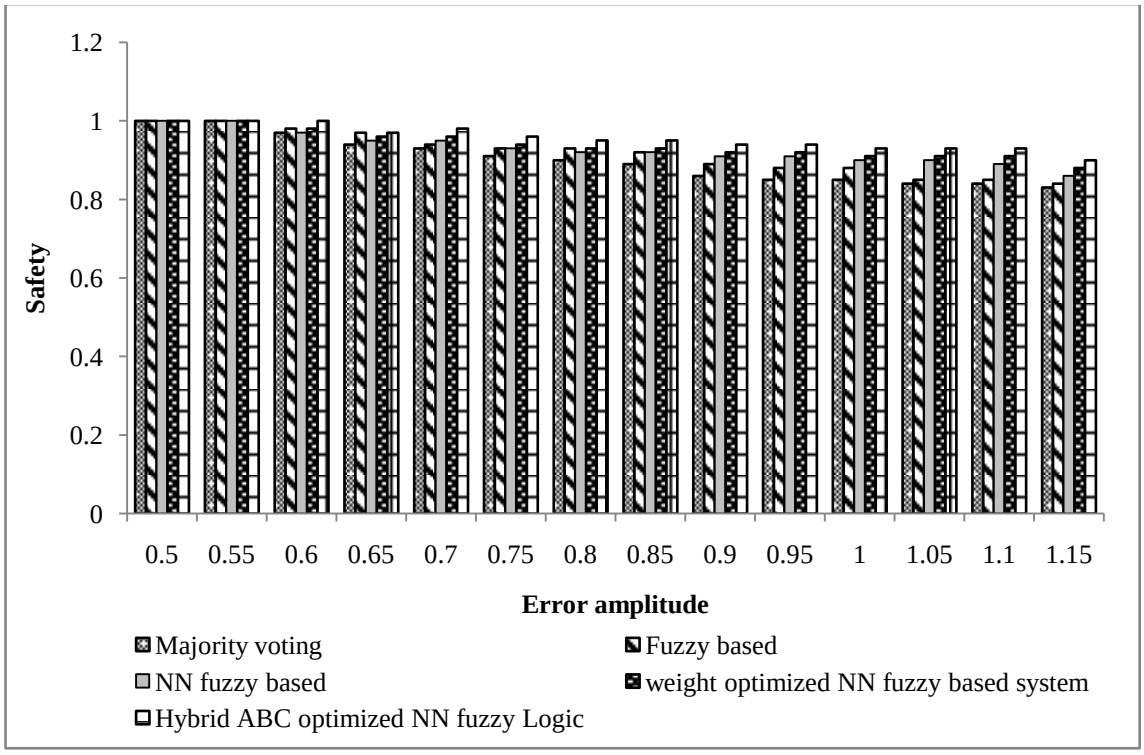


Figure 3. Safety of voters

From the Figure 3, the proposed hybrid optimized neuro-fuzzy voter shows better safety behaviour compared to the standard majority, fuzzy voters, NN fuzzy based and weight optimized NN fuzzy. When compared to NN fuzzy voter, the proposed hybrid optimized neuro-fuzzy method improved safety by 2.0833% to 4.5455% when the error amplitude is more than 0.6. The proposed hybrid optimized neuro-fuzzy method improved safety by 1.0363% to 2.2472% when the error amplitude is more than 0.6 when compared to weight optimized fuzzy based system.

Table 2: Availability of voters

| Error Amplitude | Majority voting | Fuzzy based | NN fuzzy based | weight optimized NN fuzzy based system | Hybrid ABC optimized NN fuzzy Logic |
|---|---|---|---|---|---|
| 0.5 | 1 | 1 | 1 | 1 | 1 |
| 0.55 | 1 | 1 | 1 | 1 | 1 |
| 0.6 | 0.89 | 0.93 | 0.95 | 0.96 | 0.97 |
| 0.65 | 0.87 | 0.91 | 0.9 | 0.92 | 0.94 |
| 0.7 | 0.86 | 0.9 | 0.89 | 0.91 | 0.92 |
| 0.75 | 0.82 | 0.84 | 0.85 | 0.89 | 0.91 |
| 0.8 | 0.75 | 0.79 | 0.84 | 0.87 | 0.88 |
| 0.85 | 0.73 | 0.77 | 0.83 | 0.85 | 0.87 |
| 0.9 | 0.72 | 0.75 | 0.81 | 0.84 | 0.85 |
| 0.95 | 0.71 | 0.74 | 0.81 | 0.83 | 0.84 |
| 1 | 0.69 | 0.72 | 0.77 | 0.79 | 0.81 |
| 1.05 | 0.68 | 0.7 | 0.75 | 0.78 | 0.79 |
| 1.1 | 0.65 | 0.66 | 0.74 | 0.76 | 0.77 |
| 1.15 | 0.59 | 0.64 | 0.72 | 0.77 | 0.78 |

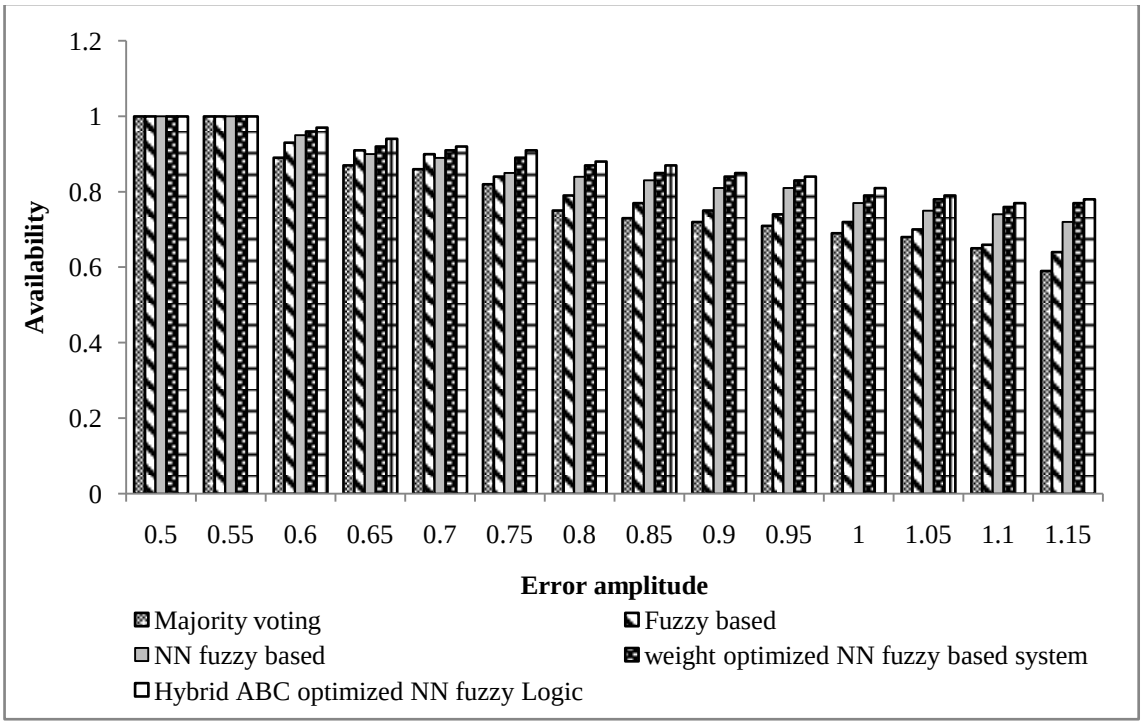


Figure 4. Availability of voters

From the Figure 4 it is observed that when compared to NN fuzzy voter, the proposed hybrid optimized neuro-fuzzy method averagely improved availability by 4.6478% when the error amplitude is more than 0.6. The proposed hybrid optimized neuro-fuzzy method averagely improved availability by 1.5571% when the error amplitude is more than 0.6 when compared to weight optimized fuzzy based system.

**Conclusion**

The neuro-fuzzy approach, symbiotically combining connectionist and fuzzy approaches merits, is a key component of soft computing at this phase. To date, there is no detailed/integrated categorization of various

neuro–fuzzy models used for rule generation. The new neuro-fuzzy voter's performance was tested on a refined experimental harness which allowed modelling of various input signal's distribution of noise and errors. The new weight optimized NN fuzzy based system ensured better safety behaviour and improved availability compared to 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]. Azadmanesh, A., Farahani, A., & Najjar, L. (2008). Fault Tolerant Weighted Voting Algorithms. International Journal of Network Security, Vol.7, No.2, PP.240-248.

[6]. Latif-Shabgahi, S. (2011). An Integrated Voting Algorithm for Fault Tolerant Systems. International Conference on Software and Computer Applications IPCSIT vol.9, IACSIT Press, Singapore

[7]. Saheb, P. B., Mr. Subbarao, K., & Dr. Kumar, S. P. (2013). A Survey on Voting Algorithms Used In Safety Critical Systems. International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 2 Issue 7, Page No. 2272-2275

[8]. 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.

[9]. 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.

[10]. 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.

[11]. Bolchini, C., Miele, A., & Sciuto, D. (2012, March). An adaptive approach for online fault management in many-core architectures. In Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012 (pp. 1429-1432). IEEE.

[12]. Muralidharan, S., Kim, J., Lütkenhaus, N., Lukin, M. D., & Jiang, L. (2014). Ultrafast and fault-tolerant quantum communication across long distances. Physical review letters, 112(25), 250501.

[13]. Aymerich, N., Cotofana, S. D., & Rubio, A. (2012). Adaptive fault-tolerant architecture for unreliable technologies with heterogeneous variability. Nanotechnology, IEEE Transactions on, 11(4), 818-829.

[14]. Meloni, P., Tuveri, G., Raffo, L., Cannella, E., Stefanov, T., Derin, O., ... & Sami, M. (2012, September). System adaptivity and fault-tolerance in noc-based mpsocs: the madness project approach. In Digital System Design (DSD), 2012 15th Euromicro Conference on (pp. 517-524). IEEE.

[15]. Tabbache, B., Benbouzid, M., Kheloui, A., Bourgeot, J. M., & Mamoune, A. (2013, November). PWM inverter-fed induction motor-based electrical vehicles fault-tolerant control. In Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE (pp. 8204-8209). IEEE.

[16]. Bolchini, C., Miele, A., & Sandionigi, C. (2013). Autonomous Fault-Tolerant Systems onto SRAM-based FPGA Platforms. Journal of Electronic Testing, 29(6), 779-793.

[17]. 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.

[18]. Straka, M., Kastil, J., Kotasek, Z., & Miculka, L. (2013). Fault tolerant system design and SEU injection based testing. Microprocessors and Microsystems, 37(2), 155-173.

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

[20]. Jacobs, A., Cieslewski, G., George, A. D., Gordon-Ross, A., & Lam, H. (2012). Reconfigurable fault tolerance: A comprehensive framework for reliable and adaptive FPGA-based space computing. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 5(4), 21.

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

[22]. 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.

[23]. MacGregor, J., & Cinar, A. (2012). Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods. Computers & Chemical Engineering, 47, 111 -120.

[24]. Narayana, G. S., Kumar, C. N., & Rambabu, C. A Comparative Analysis of PI Controller and Fuzzy Logic Controller for Hybrid Active Power Filter Using Dual Instantaneous Power Theory. International Journal of Engineering Research & Development, 4, 29-39.

[25]. Jiménez, V. P. G., Jabrane, Y., Armada, A. G., Said, B. A. E., & Ouahman, A. A. (2011). High power amplifier pre-distorter based on neural-fuzzy systems for OFDM signals. Broadcasting, IEEE Transactions on, 57(1), 149-158.

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

[27]. Vieira, J., Dias, F. M., & Mota, A. (2004, April). Neuro-fuzzy systems: A survey. In 5th WSEAS NNA International Conference on Neural Networks and Applications, Udine, Italia.

[28]. 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).

[29]. 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).

[30]. 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.

[31]. 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.

[32]. Li, J. Q., Pan, Q. K., Xie, S. X., & Wang, S. (2011). A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. Int J Comput Commun Control, 6(2), 286-296.