A Novel Weighted Voting Algorithm based on Neural Networks for Fault-Tolerant Systems

Faraneh Zarafshan\textsuperscript{a,b,d}, Gholam Reza Latif-Shabgahi\textsuperscript{c}, Abbas Karimi\textsuperscript{a,b,d}

\textsuperscript{a} Department of Computer and Communication Systems Engineering, Faculty of Engineering, UPM, Malaysia
\textsuperscript{b} Department of Computer Engineering, Faculty of Engineering, Islamic Azad University- Arak Branch, Iran
\textsuperscript{c} Computer and Electrical Department, Faculty of Engineering, Power and Water university of Technology, Tehran, Iran
\textsuperscript{d} Young Researchers Club, Islamic Azad University- Arak Branch, Iran
E-mail: fzarafshan@gmail.com

Abstract—Voting algorithms are used in a wide area of control systems from real-time and safety-critical control systems to pattern recognition, image processing and human organization systems in order to arbitrating among redundant results of processing in redundant hardware modules or software versions. From a point of view, voting algorithms can be categorized to agreement-based voters like plurality and majority or some voters which produce output regardless to agreement existence among the results of redundant variants. In some applications it is necessary to use second type voters including median and weighted average. Although both of median and weighted average voters are the choicest voters for highly available application, weighted average voting is often more trustable than median. Meanwhile median voter simply selects the mid-value of results; weighted average voter assigns weight to each input, based on their pre-determined priority or their differences, so that the share of more trustable inputs will increase rather than the inputs with low probable correctness. This paper introduces a novel weighted average voting algorithm based on neural networks that is capable of improving the rate of system reliability. Our experimental results showed that the neural weighted average voter has increased the reliability 116.63\% in general and 309.82\%, 130.27\% and 9.37\% respectively for large, medium and small errors in comparison with weighted average, and 73.87\% in general and 160.44\%, 83.59\% and 7.52\% respectively for large, medium and small errors in comparison with median voter.

Keywords—voting algorithm; fault tolerance; neural networks; reliability.

I. INTRODUCTION

One of the fault tolerance mechanisms is fault masking which is used in many fault-tolerant systems including safety critical computer systems (e.g. flight control systems, nuclear power plants, health monitoring equipments), highly reliable systems (e.g. file server, call processing applications) and highly available systems (e.g. distributed data base) in order to increase system dependability [3]. Redundancy is one of the main techniques to increase system dependability in which N redundant hardware modules (NMR: N Modular Redundancy) or N redundant software versions (NVP: N-Version Programming) are performing same tasks in parallel and a voting unit -known as a voter- arbitrates among their results to mask the effect of one or more run time errors. Since all modules use identical input and perform same task, it is expected that they produce same outputs. So each module whose result is different with the others can be assumed as faulty module and the output on which the most modules agree, probably is the correct system output. Triple Modular Redundancy (TMR) is the simplest and most applicant approach (for less complexity) form of NMR method (N=3) which is made up of three redundant modules and a voter (see fig. 1).

So far, several voting algorithms are introduced in which each approach is suitable for an application regard to its features. The most popular voting algorithms are Majority, plurality, median, weighted average[9], predictor[10], and maximum likelihood[11].

In a viewpoint, voting algorithms can be categorized into two types, first, agreement based algorithms like plurality and majority in which one of the input values that a particular number of the voter inputs agree is selected as the voter output, Whereas the second type includes voters such as median and weighted average that either produce voter output regardless of the agreement existence or simply select one of the voter inputs as voting result[6]; hence, in highly available applications where making an output in every cycle is essential, using these voters is in priority. Voting on results of redundant sensors and inertial measurement units in interface level of control systems or chemical plants, and reading distributed clocks in distributed calculation nodes for clock synchronization aims are two common examples of second type of voter applications. In such applications,
weighted average voting is often more trustable than a median voter[6], 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. As a result of advantages of weighted average voter and its importance in fault tolerant control systems especially in highly available applications, in this paper we propose a novel weighted average voting algorithm based on neural networks. Our results showed that it has improved system reliability and safety that leads to improvement in system dependability.

The paper is organized as follow: In section 2, we briefly review the related works. The details of proposed voting based on neural networks are introduced in section 3. Section 4 describes experimental model that has been used for simulations. In section 5, new voting algorithm has been analyzed and its performance is compared with common weighted average and median voting algorithms in presence of small, medium and large errors. Finally, conclusions and future works are presented.

II. RELATED WORKS

In weighted voting algorithms, the idea is to assign a weight to each voter input. Clearly, the voter input with higher weight is more effective on what the voter output is. There are two main approaches for weight assignments: pre-determined and dynamic[6]. In pre-determined method, weights are assigned to each voter input based on a pre-determined priority. This priority can be based on a priori estimated reliability or failure history records [12] whereas in dynamic approach, weight are calculated based on online history records of modules or inter module’s mechanism or the distances between voter inputs. One of the dynamic weight assignment approaches in common is Lorczak’s distance based algorithm [9.] in which if $x_i$ is the ith input, $w_i$ is the calculated weight for ith input, y is the voter output and N is the number of inputs, then the distance between ith and jth inputs ($d_{ij}$) is:

$$d_{ij} = |x_i - x_j| : i, j = 1, 2, ..., N : i \neq j$$

And weight of each input is calculated by

$$w_i = \frac{1}{1 + \frac{N}{\alpha} \prod_{i=1}^{N} \frac{d_{ij}^2}{d_{ij}^2}}$$

(α is an application-specific tolerance factor to tune input weights). Finally, voter output can be determined by:

$$y = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i}$$

In the current paper, WA denotes above voting algorithm.

Lorczak also has considered other common voting approaches such as median, majority and plurality in his paper[9].

So far, several voting algorithms based on weighted voting are introduced in literature[6],[3],[4],[2] and some papers have used neural networks for voting in some issues (e.g. voting by using neural in human organization system[13] and voting among several neural networks with different sets of learning patterns and parameters in pattern recognition [14]) , however this paper is the first reported use of a complete neural voter in the context of fault tolerance.

III. NEURAL VOTER

In order to achieve the neural weighted average voter we need to teach neural network to assign weights to each voter input based on its previous training and generate appropriate output for each input. The flowchart of design stages of neural voter have been presented in fig. 2.

The first step in training the neural network is to select the network training templates, which are implemented in the form of input vectors and desired targets. Since in TMR system, the voting cycle is performed on three inputs, the training input templates are considered to be in the form of three member vectors. A target output is also dedicated to each set of input templates. To simplify the training process, the range of the neural network inputs and outputs are considered between [0-1]. If the range of the tested data in simulation exceeds this range, we can normalize it between [0-1].

Choose appropriate training templates in form of three member vectors compatible with TMR structure  
Calculate targets of each input template based on Lorczak’s weight assignment  
Choose the type of network and its training parameters  
Train neural network  
Network performance and error  
Non-acceptable  
Change network parameters  
Acceptable  
Evaluate the network and compare the results

Figure 2: flowchart of designing neural weighted average voter
We train the input templates to the network in the form of three different steps, each containing several scenarios with hundreds of training templates, in order to achieve the least training error. Target output of each input template is extracted through Lorczak weight assignment method. The steps comprehensively considered different error rates on different elements with various combinations of faulty modules and errors, so that the network would be able to tolerate failure of every module with any error rate.

In these steps, the following points are regarded:

In first step, all modules are supposed to be as fault-free. To do so, the elements of the input vector are considered equal during training, but the elements of each vector are different from the other training vectors and are increasing periodically during training.

In second step, two first modules are supposed as non-faulty, whereas the third is faulty. We assume that faults cause errors whose symptoms appear to the voter as numerical input values perturbed by varying amounts. Hence, two elements of input training vector are attributed equivalent (non-faulty) and third element gets a different value (faulty). The differences among faulty and non-faulty elements are increasingly changed in form of 4 scenarios from 0.004 to 1 (these values are achieved by experiments). In next scenarios, we replace faulty and non-faulty modules. Totally, we fulfilled 12 scenarios in this stage.

In third step, all modules can have different values. For training this condition, we initially assign first elements of input vectors zero and increase them in a similar way of step 2. Meanwhile, we increase next elements starting from non-zero points (for instances: 0.0005 and 0.001 or 0.003 and 0.01), we consider 8 scenarios. If we exchange the elements of input vectors, then we achieve to 6 other scenarios for every 8 scenarios. Totally, we fulfilled 48 scenarios in step 3.

Since we want to train the network with the input and output templates, it is required to use the supervised training method. A neural network which is a kind of Cascade Forward back Propagation with 5 layers and 17 neurons was selected with the trial of different kinds of supervisory networks, number of its layers, the number of neurons in each layer, weights and biases of the network, non-linear functions of each layer and the training templates and with the purpose of getting to an approximately zero error. In this network, one neuron is selected for input layer, 5 in each 3 middle layers and 1 neuron in the output layer, with the Tansig function in all layers. In this paper, we utilize NN-WA as the acronym for the novel proposed algorithm.

**IV. EXPERIMENTAL METHOD**

The experimental voter configuration of this paper is the model introduced in [5, 7-8] for a TMR system (fig. 3). This model has made up of a data generator, a repeater, three saboteurs, a voter and a comparator. The data generator produces an amount for feeding the repeater in each voting cycle. We called this value as the notional correct input which is a measure to find out whether the voter output is correct or incorrect. The repeater creates copies of notional correct result and presents one copy to each saboteur. Each saboteur injects a random fault to the data, similar to real conditions in which random faults may affect each module and make it faulty. The Saboteurs can be programmed so that each generates the faulty data in a particular method. In each cycle, we can inject faults on one, two, or all three redundant data. The uniform distribution is used to produce random injected faults from 0 to 10 i.e. maximum 5% of the total range of the system input data can be faulty. Faulty data out of Saboteurs feed the voter input. Finally, voter uses the voting algorithm in order to arbitrate among the inputs to obtain the result of voting. To find out whether voter result is correct or not, the voter result is compared with the notional correct input. If their difference is more than accuracy threshold, ε -a parameter used in test environment- the voting result is incorrect, otherwise it is correct.

We assume ε =0.5 and voter, comparator, data generator, and repeater are fault-free. So the faults can only appear in the saboteurs.

For the purposes of the results reported here, similar to previous works([5, 7]) the difficult issues associated with ensuring synchronization of the inputs to the voter is ignored.

In order to analyze the voting algorithms, the following parameters are defined:

- **N**: number of total runs of a voter, which is considered 10000 times.
- **Na**: number of agreed results among n voter actions.
- **Nc**: number of voter correct agreements.
- **Nic**: number of voter incorrect agreements.
- **Nc/N**: Percentage of the voter correct answers. A voter which produces more correct results among its total outputs can be interpreted as more reliable voter. Reliability(R) is defined as the ratio of correct voter outputs to the number of voting actions: \( R = \frac{Nc}{N} \). Thus \( R \in [0,1] \) and ideally \( R=1 \) [3,8].
- **Nic/N**: percentage of the voter incorrect answers. Since from the safety (S) view point the smallest number of agreed but incorrect outputs is desirable for a given voter, the safety measure can be defined as: \( S = 1 - \frac{Nic}{N} \). Thus \( S \in [0,1] \) and ideally \( S=1 \). Due to the fact that the incorrect agreements can occasionally produce consequences of catastrophic errors, in safety critical systems the purpose is to increase this parameter to the least possible value[3, 8].
- **Na/N**: percentage of the agreed voter results. This parameter can be assumed as the Availability (A) of the voting algorithm. \( A \in [0,1] \) and ideally \( A=1 \). The main reason of using median and WA voting algorithms in highly available systems is their 100 percent availability. We expect our new algorithm has 100 percent availability as well as median and WA voters, since it has trained based on WA algorithm.
V. Experimental Results

Table 1 indicates the outputs of WA, Median and NN-WA voters for 14 selected inputs where \( x_i \in [0,100] \). The notional correct input has also indicated in the second column of the table. The results of WA, Median and NN-WA under the assumption that \( \varepsilon = 0.5 \) and the corrupted inputs by injecting random errors, \( \varepsilon \in [0,10] \), are presented in next columns. The incorrect answers of voters are highlighted in the table. Meanwhile, WA and Median voters have made incorrect agreements; NN-WA has found agreement.

The results of 10000 runs of NN-WA, WA and median algorithms are categorized in table 2. These parameters help us to select an appropriate voter for a given application depending on the nature and requirements of the application. For instance, in a highly available system the concentration is on choosing a voter with maximized value of Na/N, while in a safety critical system the smallest number of incorrect outputs (Nic) is preferred. The highlighted cells present the strength and weakness points of the voters. As we expected, the availability (Na/N) of novel algorithm in same conditions is as much as median and WA voters. The number of correct agreements in NN-WA is respectively 2 and 1.5 times more than Median and WA voters, whereas the number of incorrect agreements is less than the mentioned algorithms, experimental results in table 2 demonstrate that NN-WA voter has improved the percentage of producing correct agreements to 116.63 % and 73.87% for WA and Median voters, respectively; and from the viewpoint of producing incorrect agreements, it has a decrease close to 6.82% and 5.46% for WA and Median algorithms. The rate of Nic/Nic which refers to reliability indicates 132.64% and 84.16% improvement in NN-WA in comparison to WA and Median voters, respectively.

In fig. 4, the rate of Nic/N of NN-WA voter, Median and WA voters are compared in which less rate of Nic/N indicates more safety. Our experimental results show the decrease of Nic/N in NN-WA in comparison to WA voter up to 7.7% for small errors \( \{ \varepsilon \in [0,3]\} \), 6.3% for medium errors \( \{ \varepsilon \in (3,6]\} \) and 5.37% for large errors \( \{ \varepsilon \in (610] \) and in comparison to median voter, 6.39 % for small errors, 4.49% for medium errors and 4.39% for large errors. Consequently, NN-WA voter has priority to WA and Median voters due to its fewer incorrect agreements (more safety) and this position does apply for all three types of errors.

Similarly, the percentages of correct results of the algorithms are illustrated in fig.5. Our experimental results show a significant increase in the correct agreements of NN-WA voter in comparison to WA voter up to 9.37% for small errors, 130% for medium errors and 309% for large errors and in comparison to median voter, 7.52 % for small errors, 83.59% for medium errors and 160.44% for large errors. Hence, NN-WA voter is also better than WA and Median voter, from the viewpoint of making more correct agreements (more reliability), in general case and for all three types of errors.

Plots of fig. 6 show the unreliability of the voting algorithms. Clearly, NN-WA has more reliability (as a cause of reliability=1-unreliability) than WA and Median voting algorithms. According to equality of availability in the novel algorithm, WA and median algorithms, the new proposed algorithm can be used in the situations in which WA and Median algorithms are preferred for some reasons. This guarantees the reliability and safety increasing of the fault-tolerant control system and consequently increasing in system dependability compared to where WA and median are used.

CONCLUSIONS AND FUTURE WORKS

In the current paper, a new voter which is based on neural
networks and has trained by using Lorczak’s weighted average approach has been introduced, implemented and analyzed. Due to the wide applications of TMR in fault-tolerant systems, we utilized this model in designing the new voting algorithm. Note that training neural network is not only limited to our approach and we can design a weighted neural voter by using any other weight assignment formula.

This paper is significant, because of three reasons:

First, in the current paper, the neural networks have been used to design and implement a voting algorithm for fault-masking purposes in fault-tolerant systems. Second, according to applications of weighted average and median voting algorithms in highly available systems in which in some cases high reliability and safety are also considered, new voter is beneficial since it has the most number of correct results and the least number of incorrect results in comparison to weighted average and median voters, based on our experimental results in section 5. Finally, if we use (2) for weight assignments, we need to do several mathematical calculations and clearly extra memory space and time complexity to run the algorithm. Although, training the neural networks is a time consuming procedure, it is performed once, before running the algorithm and has no more time complexity. Also, in voting by using neural voting algorithm, there is no need to do complex mathematical and time consuming calculations.

In future work the reliability of this voter will be examined by using Markov models.

![Figure 5: Nc/N of WA, Median and NN-WA Voter for small, medium and large errors](image)

![Figure 6: the unreliability of NN-WA, Median and WA voters](image)

**REFERENCES**


