Fuzzy voting algorithms for $N$-version software

R Y Tsarev$^1$, M S Durmuş$^2$, İ Üstoglu$^3$, V A Morozov$^1$, A N Pupkov$^1$

$^1$ Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia
$^2$ Sadenco (Safe, Dependable Engineering & Consultancy), Meltem Mah. 2. Cad. Pümiye Sit. A10 5/10, Muratpasa, Antalya, Turkey
$^3$ Yıldız Technical University, Davutpaşa Mah., Esenler, Istanbul, 34220, Turkey

E-mail: tsarev.sfu@mail.ru

Abstract. The choice of the voting algorithm in $N$-version programming directly affects the evaluation of the results of $N$ software versions and determines the correct result. The result of the voting algorithm is also the outcome of the $N$-version software. Therefore, the choice of the voting algorithm is vital. However, many voting algorithms were already developed and they may be selected for implementation, based on the specifics of the analysis of input data of these algorithms. This article presents a brief overview of major fuzzy voting algorithms.

1. Introduction

The concept of $N$-version programming has many application areas like space systems, railway interlocking systems and nuclear industry where the term “Reliability” is in question. The main idea behind this method is to decrease common cause failures and provide more than one solution (software, hardware, module, etc.) for the same problem. These solutions are developed by isolated workgroups and later integrated together. This integration of different solutions are realized by using another decision-making unit which is also known as “the voter” [1]. The voter receives the output of each version and produces the final decision according to its algorithm.

Recently, there are many algorithms for voting, which differ in schemes of work and the requirements of the original data, see e.g., [2-8]. Each algorithm has its own advantages and disadvantages, and there are certain conditions for its successful application. Among the entire set of voting algorithms in the literature, some of the algorithms have common features. However, in the literature of $N$-version method, there is no general classification of voting algorithms.

This paper briefly describes the Fuzzy Voting Algorithms which can be classified as voting algorithms based on the comparison of the output data. Since, the whole set of the output data (results) of the $N$ different versions are divided into disjoint subsets during the analyses, Fuzzy Voting Algorithms are also regarded as Non-Formalized Voting Algorithms like Classical Voting Algorithms [9-14] or The Voting Algorithms with Minimization [8], [15], [16].

1.1. The Agreement Matrix

In order to describe the relation between $N$-versions, it is a good practice to explain the use of so-called agreement matrix. The agreement matrix is a Boolean matrix with $N \times N$ dimensions (where $N$ is the number of versions). Agreement matrix reflects the equivalence of each output to other outputs. The elements of the agreement matrix are calculated by the following principle:
where \( r_{ij} \) is the element of the agreement matrix \( i \)-row and \( j \)-column, \( x_i \) and \( x_j \) outputs; \( \varepsilon \) is the tolerance value, checked for equivalence.

The following additional requirements are applied to the agreement matrix. Equivalence relation should be executed on the agreement matrix \( R \). This relation includes reflexivity, symmetry and transitivity, respectively (2)-(4):

\[
\begin{align*}
    r_{ii} &= 1, \forall i, \quad (2) \\
    r_{ij} &= r_{ji}, \forall i \neq j, \quad (3) \\
    \text{if } r_{ik} = 1 \text{ and } r_{kj} = 1 \text{ then } r_{ij} = 1, \forall i, j. \quad (4)
\end{align*}
\]

Meeting such requirement is necessary to solve the inconsistent partitioning problem (see e.g., [11]).

If the equivalence ratio (2)-(4) is not performed, the Boolean compositions must be applied to the agreement matrix (see e.g., [17]). Execution of the Boolean compositions should be realized as long as the equivalence relation is not satisfied. In fact, reflexivity and symmetry in the agreement matrix are always performed. In the general case, only the one feature of transitivity cannot be realized. Relation, on which only the features of reflexivity and symmetry are realized, is called tolerance relation (see e.g., [17]). In the work [17], it is shown that if a valid relation is performed on the agreement matrix, and then no more than \( N - 1 \) of the Boolean matrices we get the agreement matrix, on which the equivalence relation (2)-(4) is executed. Here, \( N \) is the number of versions, and accordingly the number of columns and rows in the agreement matrix.

1.2. The Boolean Compositions on the Agreement Matrix

The aim of the Boolean compositions on the agreement matrix is to transform the agreement matrix into a form where an equivalence relation can be executed. In general, the operation of the Boolean compositions is defined for matrices as follows:

For the given the matrixes \( A \) and \( B \); all elements of the matrix \( A, a_{ij} \) takes the values 0 or 1, and all elements of the matrix \( B, b_{ij} \) takes the values 0 or 1. Then the Boolean composition of the matrixes \( A \) and \( B \) is as follows:

\[
C = A \odot B, \quad \text{where } c_{ij} = \bigoplus_{k=1}^{N} (a_{ik} \otimes b_{kj}), \quad (5)
\]

where \( c_{ij} \) represents the elements of the resulting matrix, \( \oplus \) - function of logical “or”, \( \otimes \) - function of logical “and”.

To achieve the execution of the equivalence relation (2)-(4) in the agreement matrix \( R \), it is necessary to have the consequent implementation of Boolean compositions of \( R \) with itself based on the following principle:

\[
E = R^1 \cup R^2 \cup R^3 \cup \ldots \cup R^Q, 1 \leq Q \leq N - 1 \quad (6)
\]

where \( E \) is the agreement matrix, on which the equivalence relation is executed; \( Q \) – the number of consequent Boolean compositions; \( N \) – the number of versions;

\[
R^1 = R, \quad R^2 = R \circ R, \quad R^2 = R \circ R \circ R, \quad ... \quad (7)
\]

Thus, if the equivalence relation is not executed on the agreement matrix \( R \), then it is necessary to perform one Boolean composition:

\[
E^2 = R \cup R \circ R, \quad (8)
\]

if the equivalence relation is not executed on the resulting modified agreement matrix \( E^2 \), then it is
necessary to perform the following Boolean composition:

\[ E^3 = R \cup R \circ R \cup R \circ R = E^2 \cup R^3. \]  

(9)

In each row of the agreement matrix we calculate the number of units. \( Y_i \) indicates the number of units in \( i \)-row. If there exists such row \( i \), which satisfies:

\[ Y_i \geq \left\lfloor \frac{N + 1}{2} \right\rfloor, \]  

(10)

then the set of correct results is generated from those results, which correspond to units in the row \( i \). Operator \( \left\lfloor \cdot \right\rfloor \) in (10) means “ceiling”, its result is greater than or equal to the argument of the ceiling operator. The principle of selecting the results of the versions is illustrated in Figure 1, where \( A \) is the set of correct results.

\[
\begin{array}{cccccc}
    x_{i1} & x_{i2} & x_{i3} & x_{i4} & x_{i5} & \ldots \\
     \vdots & \vdots & \vdots & \vdots & \vdots & \ldots \\
    Y_j & 1 & 1 & 0 & 1 & 0 & \ldots \\
     \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & \ldots \\
\end{array}
\]

\[ A = \{x_{i1}, x_{i2}, x_{i3}, \ldots\} \]

**Figure 1.** Selecting the correct answers from the agreement matrix.

### 2. Fuzzy Voting Algorithms

Application of the theory of fuzzy logic in voting algorithms in \( N \)-version software is covered in papers [16-23]. Algorithms Fuzzy Consensus Voting (Fuzzy CV) and Fuzzy Majority Voting (Fuzzy MV) belong to the fuzzy ones. Algorithms for this group are characterized by the following distinctive features.

#### 2.1. Feature 1

Construction of the *similarity relation* and the *agreement matrix* is the basis of making the decision. Similarity relation is a matrix of dimension \( N \times N \) (where \( N \) is the number of versions). This means that *similarity measures* are the elements of similarity relations. Elements of similarity relations are calculated as follows:

\[
s(x_i, x_j) = \begin{cases} 
  1 - \frac{x_i - x_j}{\varepsilon}, & \text{when } |x_i - x_j| < \varepsilon, \\
  0, & \text{when } |x_i - x_j| \geq \varepsilon,
\end{cases}
\]  

(11)

where \( s(x_i, x_j) \) is the element of similarity relations in \( i \)-th row and \( j \)-th column (similarity measure of the outputs \( x_i \) and \( x_j \); \( \varepsilon \) – the tolerance value. The similarity measure of two outputs always lies in the range from zero to one inclusive: \( 0 \leq s(x_i, x_j) \leq 1 \).

The agreement matrix is made on the basis of the similarity relation. The agreement matrix is a Boolean matrix with dimensions \( N \times N \). The agreement matrix should be obtained from the similarity relation by applying the \( \lambda \) cut operation. The \( \lambda \) cut operation transforms the similarity relation \( S \) into the Boolean agreement matrix \( R \), the elements of which are equal to 0 or 1. If there is the similarity relation \( S \) and its elements \( s(x_i, x_j) \), then the elements of the agreement matrix \( r_{ij} \) can be found as follows:

\[
r_{ij} = \begin{cases} 
  1, & \text{when } s(x_i, x_j) \leq \lambda, \\
  0, & \text{when } s(x_i, x_j) > \lambda,
\end{cases}
\]  

(12)
where \( \lambda \) is the value used for the \( \lambda \) cut operation. It should be noted that the choice of the value \( \lambda \) is very important because it determines the voting result (see e.g., [23]).

2.2. Feature 2

The following additional requirement is applied to the similarity relation: on the similarity relation \( S \) the equivalence relation that includes reflexivity, symmetry and transitivity must be performed:

\[
s(x_i, x_j) = 1, \forall i,
\]

\[
s(x_i, x_j) = s(x_j, x_i), \forall i \neq j,
\]

\[
\begin{cases}
  s(x_i, x_k) = s_1 \\
  s(x_k, x_j) = s_2
\end{cases} \Rightarrow s(x_i, x_j) = s : s \geq \min[s_1, s_2].
\]  

2.3. Feature 3

If (13)-(15) is not satisfied, then the fuzzy compositions must be applied to the similarity relation (see e.g., [17]). Execution of fuzzy compositions must be done until the equivalence relation is not satisfied. In fact, reflexivity and symmetry are always satisfied on the similarity relation. In the general case only transitivity feature may not be done. The similarity relation, on which only the properties of reflexivity and symmetry are satisfied, is called fuzzy tolerance relation (see e.g., [17]). It is shown in [17], since the similarity relation is a fuzzy tolerance similarity relation, we can obtain the similarity relation for no more than \( N - 1 \) fuzzy compositions. Here \( N \) is the number of versions and, respectively, the number of columns and rows in the similarity relation matrix.

3. Fuzzy Compositions on the Similarity Relation

The aim of fuzzy compositions is the conversion of a similarity relation to such mode that the equivalence relation could be realized on it. In general, the fuzzy compositions operation is defined by the following:

Assume that the matrixes \( A \) and \( B \) be are given. All elements of the matrix \( A \), \( a_{ij} \) takes values within the range between zero and one \((0 \leq a_{ij} \leq 1)\); and all elements of the matrix \( B \), \( b_{ij} \) takes values within the range between zero and one \((0 \leq b_{ij} \leq 1)\). Then the fuzzy composition of the matrices \( A \) and \( B \) shall be:

The text of your paper should be formatted as follows:

\[
C = A \circ B, \text{where } c_{ij} = \max_k [\min(a_{ik}, b_{kj})], k = 1, \ldots, N,
\]  

where \( c_{ij} \) represents the elements of the resultant matrix.

In order to realize the equivalence relation on the similarity relation \( S \), sequential execution of fuzzy compositions with itself is required:

\[
E = S^1 \circ S^2 \circ S^3 \circ \ldots \circ S^Q, \quad 1 \leq Q \leq N - 1
\]

where \( E \) is the similarity relation on which equivalence relation (3)-(5) is executed; \( Q \) is the quantity of consecutive fuzzy compositions and \( N \) is the number of versions.

For example, if the equivalence relation is not executed on the similarity relation \( S \), execution of one fuzzy composition is required such as:

\[
E^2 = S \circ S.
\]

Additionally, if the equivalence relation is not executed on a received modified matrix, execution of the next fuzzy composition is required:

\[
E^3 = S \circ S \circ S = E^2 S.
\]

An illustration of the use of fuzzy compositions is given in the following example.
3.1. Example 1

Let any $N$-version software module have $N = 4$ number of versions, tolerance value $\varepsilon = 0.04$ and the following set of outputs has been acquired: $\{0.775; 0.781; 0.729; 0.783\}$. Similarity relation, received according to the formula (11), is demonstrated in Figure 2.

$$\begin{array}{|c|c|c|c|c|} 
\hline
 & x_1 & x_2 & x_3 & x_4 \\
\hline
x_1 & 1 & 0.85 & 0 & 0.8 \\
x_2 & 0.85 & 1 & 0 & 0.95 \\
x_3 & 0 & 0 & 1 & 0 \\
x_4 & 0.8 & 0.85 & 0 & 1 \\
\hline
\end{array}$$

Figure 2. Similarity relation.

It is observable that equivalence relation is not executed on a similarity relation $S$ while the transitive property is not fulfilled:

$$s(x_1, x_2) = 0.85 \rightarrow s(x_1, x_4) = 0.8$$

$$s(x_2, x_4) = 0.95 \rightarrow s(x_1, x_4) = 0.8$$

$s(x_1, x_4) < \min \{s(x_1, x_2), s(x_1, x_4)\}$, as long as $0.8 < 0.85$.

A decision cannot be made upon the basis of similarity relation $S$ using fuzzy algorithms. Let us commit the operation of fuzzy compositions according to the formula (18). Elements of the resultant matrix $E^2$ are introduced in Table 1 and Figure 3.

| $i$ | $j$ | $e_{ij}^2$ |
|-----|-----|----------|
| 1   | 1   | 0.04     |
| 1   | 2   | 0.85     |
| 1   | 3   | 0.05     |
| 1   | 4   | 0.04     |
| 2   | 1   | 0.05     |
| 2   | 2   | 0.85     |
| 2   | 3   | 0.05     |
| 2   | 4   | 0.04     |
| 3   | 1   | 0.05     |
| 3   | 2   | 0.85     |
| 3   | 3   | 0.05     |
| 3   | 4   | 0.04     |
| 4   | 1   | 0.05     |
| 4   | 2   | 0.85     |
| 4   | 3   | 0.05     |
| 4   | 4   | 0.04     |

Figure 3. Elements of the resultant matrix.

Table 1. Elements of the resultant matrix

Equivalence relation (13)-(15) is executed on the similarity relation $E^2$. This means that decision may be made on the basis of $E^2$ and with the help of Fuzzy MV or Fuzzy CV algorithms.
4. Fuzzy Majority Voting (Fuzzy MV)
Let us assume that there is some module of $N$-version software realized by $N$ versions. The output values returned by each version are indicated with $x_1, x_2, \ldots, x_N$. After setting the tolerance value as $\epsilon$, Fuzzy MV algorithm consists of the following steps:

**Step 1.** Develop the similarity relation.
Similarity relation $S$ is developed according to (11).

**Step 2.** Check the equivalence relation for the similarity relation.
The equivalence relation (13)-(15) is executed on the similarity relation $S$. If the equivalence relation is executed, proceed to the Step 4, otherwise proceed to the Step 3.

**Step 3.** Execute the fuzzy compositions.
The fuzzy compositions (18) are executed on the similarity relation $S$ until the equivalence relation is satisfied.

**Step 4.** Develop the agreement matrix.
The agreement matrix $R$ is developed according to (12).

**Step 5.** Determine the set of correct outputs.

This step implies the consideration of an achieved agreement matrix $R$. Number of items is calculated in each line of the matrix $R$. The number of items in the $i$-line are marked with $Y_i$. If there is such a line $i$ for which (10) is executed, then the set of correct version results is formed of those results to which the items in the line $i$ corresponds. The principle of the selection of the version results is illustrated in Figure 1. It should be noted that if the agreement matrix does not contain the lines which satisfy the requirements (10), decision cannot be made with the help of the Fuzzy MV algorithm.

5. Fuzzy Consensus Voting (Fuzzy CV)
Assume that there is some module of $N$-version software realized by $N$ versions. The output values returned by each version are indicated with $x_1, x_2, \ldots, x_N$. After setting the tolerance value as $\epsilon$, Fuzzy CV algorithm consists of the following steps:

**Step 1.** Develop the similarity relation.
The similarity relation $S$ is developed according to (11).

**Step 2.** Check the equivalence relation for the similarity relation.
The equivalence relation (13)-(15) is executed on the similarity relation $S$. If the equivalence relation is executed, proceed to the Step 4, otherwise proceed to the Step 3.

**Step 3.** Execute the fuzzy compositions.
The fuzzy compositions (18) are executed on the similarity relation $S$ until the equivalence relation is satisfied.

**Step 4.** Develop the agreement matrix.
The agreement matrix $R$ is developed according to (12).

**Step 5.** Determine the set of correct outputs.

The number of items is calculated in each line of the matrix. The number of items in the $i$-line is marked with $Y_i$. Then, the line with maximum $Y_i$ value is selected. The set of correct version results is formed of those results to which the items in the line $i$ corresponds. The principle of the selection of the version results is illustrated in Figure 1. If the agreement matrix contains more than one line in which the number of its items is maximal, a line should be selected randomly. It should be pointed out that, the Fuzzy CV algorithm always produces a result anyway because, and the algorithm will return the randomly selected result of version even if there are no agreed versions.
6. Conclusion
This paper briefly explains the use of fuzzy voting algorithms in $N$-version programming. Even if there are many studies and different type of algorithms for the $N$-version programming. Both, the choice of the voting algorithm and the number of versions ($N$) are still important points worth stressing.

7. References
[1] Avizienis A and Chen L 1977 On the implementation of $N$-version programming for software fault-tolerance during program execution Proc. COMPSAC’77 (Chicago, IL) p 149–155
[2] Armounsh A, Salewski F and Kowalewski S 2008 Recovery block with backup voting: A new pattern with extended representation for safety critical embedded systems Proc. ICIT 2008 (Bhubaneswar, India) p 232-237
[3] Durmuş M S, Eriş O, Yıldırım U and Söylemez M T 2015 A new bitwise voting strategy for safety-critical systems with binary decisions Turk. J. Electr. Eng. Co. 23 (5) 1507-1521
[4] Ionescu T B and Scheuermann W 2016 Improving the reliability of decision-support systems for nuclear emergency management by leveraging software design diversity Journal of Computing and Information Technology 24 (1) 45-63
[5] Latif-Shabgahi G, Bass J M. and Bennett S 2004 A taxonomy for software voting algorithms used in safety-critical systems IEEE Trans. on Reliability 53 (3) 319-328
[6] Tsarev R Y, Durmuş M S, Üstoglu I and Morozov V A 2018 Application of majority voting and consensus voting algorithms in $N$-version software J. Phys. Conf. Ser. 1015 (4) art 042059
[7] Tsarev R Y, Durmuş M S, Üstoglu I and Morozov V A 2018 Classification of voting algorithms for $N$-version software J. Phys. Conf. Ser. 1015 (4) art 042060
[8] Yacoub S 2003 Analyzing the behavior and reliability of voting systems comprising tri-state units using enumerated simulation Reliab. Eng. Syst. Safe. 81 (2) 133-145
[9] Ahamad M and Ammar M H 1989 Performance characterization of quorum-consensus algorithms for replicated data IEEE Trans. on Soft. Eng. 15 (4) 492-496
[10] Blough D M and Sullivan O F 1990 A comparison of voting strategies for fault-tolerant distributed systems Proc. 9th Symp. Reliable Distributed Systems (Huntsville, AL) p 136-145
[11] Brilliant S S, Knight J C and Leveson N G 1989 The consistent comparison problem in $N$-version software IEEE Trans. on Soft. Eng. 15 (11) 1481-1485
[12] Kuncheva L I, Whitaker C J, Sh27ipp C A and Duin R P W 2003 Limits on the majority vote accuracy in classifier fusion Pattern Analys. & App. 6 (1) 22-31
[13] Lam L and Suen C Y 1997 Application of majority voting to pattern recognition: An analysis of its behavior and performance IEEE Trans. Syst. Man Cybern. A, Syst. Humans 27 (5) 553-568
[14] Leveson N G, Cha S S, Knight J C and Shimeall T J 1990 of self checks and voting in software error detection: An empirical study IEEE Trans. on Soft. Eng. 16 (4) 432-443
[15] Lorczak P R, Caglayan A K and Eckhardt D E 1989 A theoretical investigation of generalized voters for redundant systems Proc. FTCS (Chicago, IL) p 444-451
[16] Parhami B 1994 Voting algorithms IEEE Trans. on Reliability 43 (4) 617-629
[17] Kim K, Vouk M A and McAllister D F 1998 Fault-tolerant software voters based on fuzzy equivalence relation Proc. IEEE Aerospace Conference (Snowmass at Aspen, CO) p 5-19
[18] Bezdeck J C and Douglas Harris J 1978 Fuzzy partitions and relations; an axiomatic basis for clustering Fuzzy Sets Syst. 1 (2) 111-127
[19] Clic J C and Yuan B 1995 Fuzzy Sets and Fuzzy Logic: Theory and Applications (NJ: Prentice Hall)
[20] Hsu H M and Chen C T 1996 Aggregation of fuzzy opinions under group decision making Fuzzy Sets Syst. 79 (3) 279-285
[21] Manic M and Frincke D 2001 Towards the fault tolerant software: Fuzzy extension of crisp equivalence voters Proc. IECON'2001 (Denver, CO) p 84-89
[22] Ross T J 1995 Fuzzy Logic with Engineering Applications (New York: McGraw-Hill) p 87-96
[23] Wang P 2017 A voting strategy for $N$-version program based on fuzzy clustering Proc. ICEMI’07 (Xi’an, China) p 2666-2669