Journal of Physics: Conference Series

PAPER • OPEN ACCESS

Application of majority voting and consensus voting algorithms in N-version software

To cite this article: R Yu Tsarev *et al* 2018 *J. Phys.: Conf. Ser.* **1015** 042059

IOP Conf. Series: Journal of Physics: Conf. Series **1015** (2018) 042059 doi:10.1088/1742-6596/1015/4/042059

Application of majority voting and consensus voting algorithms in N-version software

R Yu Tsarev[1], **M S Durmuş**[2], **I Üstoglu**[3], **V A Morozov**[1]

[1] Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia
[2] Pamukkale University, Kinikli Campus, Denizli, 20070, Turkey
[3] Yıldız Technical University, Davutpasa Mah., Esenler, Istanbul, 34220, Turkey

E-mail: tsarev.sfu@mail.ru

Abstract. N-version programming is one of the most common techniques which is used to improve the reliability of software by building in fault tolerance, redundancy and decreasing common cause failures. N different equivalent software versions are developed by N different and isolated workgroups by considering the same software specifications. The versions solve the same task and return results that have to be compared to determine the correct result. Decisions of N different versions are evaluated by a voting algorithm or the so-called voter. In this paper, two of the most commonly used software voting algorithms such as the majority voting algorithm and the consensus voting algorithm are studied. The distinctive features of N-version programming with majority voting and N-version programming with consensus voting are described. These two algorithms make a decision about the correct result on the base of the agreement matrix. However, if the equivalence relation on the agreement matrix is not satisfied it is impossible to make a decision. It is shown that the agreement matrix can be transformed into an appropriate form by using the Boolean compositions when the equivalence relation is satisfied.

1. Introduction

N-version programming provides a high level of reliability and fault tolerance for software and prevents the software from getting into fatal failures [1]. N-version programming has proven the efficiency in solving a wide range of software engineering problems [2, 3]. N-version programming assumes independent generation of a number of functionally equivalent versions according to the software input-output specification. Software versions in the N-version software implement different methods and algorithms solving the same problem. This approach ensures that an error or a fault of one of the software versions will not lead to disruption of the work of the software in general [4]. This property is very important for the systems which are characterized by high demands on reliability and availability. Thus, N-version execution of software allows one to compensate and mask failures or faults of certain software versions, and thus provide fault tolerance and guarantee the achievement of the objective functions of the software.

N-version programming is based on software redundancy. *N* different software versions are implemented by applying the principle of diversity. When using the N-version approach, there is a possibility that the versions may return different results (outputs). In this case, there arises the problem to determine which results are valid and which are faulty. This problem can be solved by the use of appropriate voting algorithms. It is obvious that the decision of the voting algorithm determines the

result of N-version software.

At present, many voting algorithms have been proposed in the literature. Voting algorithms differ from each other in the work scheme and the dependency to initial data. Selecting the suitable voting algorithm for the given data set is vital. It may be noted that the voting algorithms based on a comparison of the versions' output are efficient and intuitive. However, the application of such algorithms requires partitioning data into subsets where elements are equivalent to each other. Such partition in some cases is difficult due to inconsistency of partitions.

In this study, two fundamental voting algorithms that can be applied in N-version software are discussed. These algorithms are N-version programming with majority voting (NPV-MV) and N-version programming with consensus voting (NVP-CV). For detailed information about these algorithms, the readers can be referred to [3], [5-18].

2. Distinctive features of the voting algorithms

The distinctive features of N-version programming with majority voting and N-version programming with consensus voting are as follows.

Feature 1. The most important point in making the decision on selecting the correct set of outputs relies on the construction and analysis of the agreement matrix. The agreement matrix is a square Boolean matrix with $N \times N$ dimensions (where N is the number of versions). The agreement matrix reflects the equivalence of each output to other outputs. The elements of agreement matrix R are calculated as follows:

$$r_{ij} = \begin{cases} 1, & \text{when } \left|x_i - x_j\right| \le \varepsilon, \\ 0, & \text{when } \left|x_i - x_j\right| > \varepsilon, \end{cases} \tag{1}$$

where i indicates the row and j indicates the column of the agreement matrix; x_i and x_j are the outputs; ε is the tolerance value, checked for equivalence.

Feature 2. The following additional requirements are applied to the agreement matrix. Equivalence relation on agreement matrix R should be satisfied. This relation includes reflexivity, symmetry and transitivity properties, respectively:

$$r_{ii} = 1, \forall i, \tag{2}$$

$$r_{ij} = r_{ji}, \forall i \ne j, \tag{3}$$

$$\text{if } r_{ik} = 1 \text{ and } r_{kj} = 1 \text{ then } r_{ij} = 1, \forall i, j. \tag{4}$$

Performing such requirement is necessary to solve the *inconsistent partitioning problem* [10].

Feature 3. If the equivalence ratio (2)-(4) is not satisfied, the Boolean compositions must be applied to the agreement matrix [19]. Execution of the Boolean compositions should be realized as long as the equivalence relation is not satisfied. In fact, reflexivity and symmetry properties in the agreement matrix are always performed. In the general case, only the one feature of transitivity cannot be realized. The relation, in which only the properties of reflexivity and symmetry are realized, is called *tolerance relation* [19]. In work [19], it is shown that if a valid relation is performed on the agreement matrix and then no more than $N - 1$ of the Boolean matrixes, we get the agreement matrix, on which the equivalence relation (2)-(4) is satisfied. Here, N is the number of versions, and accordingly the number of columns and rows in the agreement matrix.

3. Boolean compositions on the agreement matrix

The aim of the Boolean compositions on the agreement matrix is to transform the agreement matrix into a proper form where the equivalence relation is satisfied. In general, the operation of the Boolean composition which is defined for matrixes is as follows:

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

$$C = A \circ B, \text{where } c_{ij} = \bigoplus_{k=1}^{N} (a_{ik} \otimes b_{kj})$$

where c_{ij} represents the elements of the resulting matrix, $\oplus$ represents a function of logical "or" and $\otimes$ represents a function of logical "and".

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

$$E = R^1 \cup R^2 \cup R^3 \cup ... \cup R^Q, 1 \le Q \le N-1, \quad (5)$$

where *E* is the agreement matrix, on which the equivalence relation is satisfied, *Q* is the number of consequent Boolean compositions, *N* is the number of versions; $R^1 = R$, $R^2 = R \circ R$, ,…

Thus, if the equivalence relation is not satisfied on agreement matrix *R*, then it is necessary to perform one Boolean composition:

$$E^2 = R \cup R \circ R. \quad (6)$$

In case if the equivalence relation is not satisfied on resulting modified agreement matrix E^2, then it is necessary to perform the following Boolean composition:

$$E^3 = R \cup R \circ R \cup R \circ R \circ R = E^2 \cup R^3.$$

The application of Boolean compositions is illustrated by an example in the next section.

4. Application of Boolean compositions

Let us suppose that for some module of N-version software, the number of versions is N = 5, the tolerance value is ε = 0.0005, and the following outputs have been obtained: {1.5531; 1.5533; 1.5544; 1.5546; 1.5537}. The agreement matrix calculated according to formula (1) is shown in Figure 1.

$R =$

	x_1	x_2	x_3	x_4	x_5
x_1	1	1	0	0	0
x_2	1	1	0	0	1
x_3	0	0	1	1	0
x_4	0	0	1	1	0
x_5	0	1	0	0	1

Figure 1. The agreement matrix.

The equivalence relation for the agreement matrix given in Figure 1 is not satisfied, because the property of transitivity is not satisfied (r_{12} = 1 and r_{25} = 1, but $r_{15} \neq 1$). On the basis of agreement matrix *R*, it is not possible to make a decision using NVP-MV or NVP-CV algorithms. Therefore, an operation of Boolean compositions shall be performed according to formula (6). Calculation of R^2 is as follows:

$$r_{11}^2 = (r_{11} \otimes r_{11}) \oplus (r_{12} \otimes r_{21}) \oplus (r_{13} \otimes r_{31}) \oplus (r_{14} \otimes r_{41}) \oplus (r_{15} \otimes r_{51}) = 1;$$
$$r_{12}^2 = (r_{11} \otimes r_{12}) \oplus (r_{12} \otimes r_{22}) \oplus (r_{13} \otimes r_{32}) \oplus (r_{14} \otimes r_{42}) \oplus (r_{15} \otimes r_{52}) = 1;$$
$$r_{13}^2 = (r_{11} \otimes r_{13}) \oplus (r_{12} \otimes r_{23}) \oplus (r_{13} \otimes r_{33}) \oplus (r_{14} \otimes r_{43}) \oplus (r_{15} \otimes r_{53}) = 0;$$
$$r_{14}^2 = (r_{11} \otimes r_{14}) \oplus (r_{12} \otimes r_{24}) \oplus (r_{13} \otimes r_{34}) \oplus (r_{14} \otimes r_{44}) \oplus (r_{15} \otimes r_{54}) = 0;$$
$$r_{15}^2 = (r_{11} \otimes r_{15}) \oplus (r_{12} \otimes r_{25}) \oplus (r_{13} \otimes r_{35}) \oplus (r_{14} \otimes r_{45}) \oplus (r_{15} \otimes r_{55}) = 1;$$
$$r_{21}^2 = (r_{21} \otimes r_{11}) \oplus (r_{22} \otimes r_{21}) \oplus (r_{23} \otimes r_{31}) \oplus (r_{24} \otimes r_{41}) \oplus (r_{25} \otimes r_{51}) = 1;$$
$$r_{22}^2 = (r_{21} \otimes r_{12}) \oplus (r_{22} \otimes r_{22}) \oplus (r_{23} \otimes r_{32}) \oplus (r_{24} \otimes r_{42}) \oplus (r_{25} \otimes r_{52}) = 1;$$
$$r_{23}^2 = (r_{21} \otimes r_{13}) \oplus (r_{22} \otimes r_{23}) \oplus (r_{23} \otimes r_{33}) \oplus (r_{24} \otimes r_{43}) \oplus (r_{25} \otimes r_{53}) = 0;$$

$$r_{24}^2 = (r_{21} \otimes r_{14}) \oplus (r_{22} \otimes r_{24}) \oplus (r_{23} \otimes r_{34}) \oplus (r_{24} \otimes r_{44}) \oplus (r_{25} \otimes r_{54}) = 0;$$
$$r_{25}^2 = (r_{21} \otimes r_{15}) \oplus (r_{22} \otimes r_{25}) \oplus (r_{23} \otimes r_{35}) \oplus (r_{24} \otimes r_{45}) \oplus (r_{25} \otimes r_{55}) = 1;$$
$$r_{31}^2 = (r_{31} \otimes r_{11}) \oplus (r_{32} \otimes r_{21}) \oplus (r_{33} \otimes r_{31}) \oplus (r_{34} \otimes r_{41}) \oplus (r_{35} \otimes r_{51}) = 0;$$
$$r_{32}^2 = (r_{31} \otimes r_{12}) \oplus (r_{32} \otimes r_{22}) \oplus (r_{33} \otimes r_{32}) \oplus (r_{34} \otimes r_{42}) \oplus (r_{35} \otimes r_{52}) = 0;$$
$$\dots$$
$$r_{55}^2 = (r_{51} \otimes r_{15}) \oplus (r_{52} \otimes r_{25}) \oplus (r_{53} \otimes r_{35}) \oplus (r_{54} \otimes r_{45}) \oplus (r_{55} \otimes r_{55}) = 1.$$

$$E^2 = R \cup R^2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} \cup \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

On modified agreement matrix E^2, the equivalence relation (2)-(4) is performed, and hence, the decision can be made using the NVP-MV or NVP-CV algorithm.

5. N-version programming with majority voting algorithm (NVP-MV)

Let us assume that there are N different software versions for the use of N-version programming. The output values returned by each version are indicated with $x_1, x_2, \dots, x_N$. After setting tolerance value ε, the algorithm is as follows:

Step 1. Construct agreement matrix R.

The agreement matrix is constructed according to formula (1).

Step 2. Check the equivalence relations on agreement matrix R.

On the agreement matrix, the equivalence relation must be satisfied in accordance with (2)-(4). If the equivalence relation is satisfied, then go to step 4, otherwise, go to step 3.

Step 3. Perform the Boolean compositions.

The Boolean composition (5) is performed until the equivalence ratio (2)-(4) for agreement matrix R will not be satisfied.

Step 4. Define the set of correct outputs.

In each row of the agreement matrix, the number of units is calculated. Y_i indicates the number of units in row i. If there is such row i, which satisfies:

$$Y_i \geq \left\lceil \frac{N+1}{2} \right\rceil, \tag{7}$$

then the set of correct results is generated from those results, which correspond to units in row i. Operator $\lceil\ \rceil$ in (7) 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 2, where A is the set of correct results.

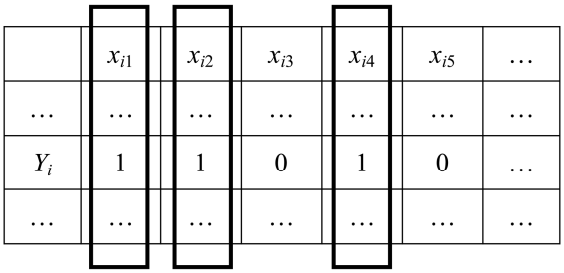

Figure 2. Selecting the correct answers from the agreement matrix.

6. N-version programming with the consensus voting algorithm (NVP-CV)

Let us assume once again that there are N different software versions for the use of N-version programming. The output values returned by each version are indicated with x_1, x_2, …, x_N. After setting tolerance value ε, the algorithm is as follows:

Steps 1-3 of this algorithm are similar to steps 1-3 of the majority voting algorithm (NVP-MV).

Step 4. Define the set of correct outputs.

In each row of the agreement matrix, the number of units is calculated. Y_i indicates the number of units in row i. Next, the row in which the Y_i is maximal is selected. The set of correct results is generated from those results which correspond to the units in row i. The main reason of selecting the results of the versions is similar to the one that is illustrated in Figure 2. If the agreement matrix contains more than one row in which the number of units is maximal, then the row can be selected randomly.

It should be noted that NVP-CV algorithm produces a result in any case, even if there are no consistent versions. The algorithm returns the output selected randomly.

7. Conclusion

N-version programming use redundant software components which are developed following design diversity rules [20].Redundant software versions solve the same task in different ways implementing diverse algorithms written in different programming languages by different developer teams. Diversity of the versions brings us slightly or totally different results. Thus, it is necessary to analyze the versions' results and determine the correct one. This analyzing mechanism in N-version software is the voter. Voting algorithm plays a crucial role as far as it determines the result of N-version software in general.

This paper considers two well-known voting algorithms, N-version programming with majority voting (NPV-MV) and N-version programming with consensus voting (NPV-CV) that can be implemented in N-version software. Both these algorithms make a decision about the correct versions' result based on the agreement matrix. The agreement matrix reflects the equivalence of the result of a version to other versions' results. In such case, the equivalence relation on the agreement matrix is satisfied and it is impossible to make a decision. However, applying the Boolean compositions on the agreement matrix it is possible to transform the agreement matrix into a proper form where the equivalence relation is satisfied.

This paper describes the distinctive properties of N-version programming with majority voting and N-version programming with consensus voting. It provides the theoretical basis of the Boolean composition application and an example of the application of Boolean compositions to an agreement matrix. Finally, the steps of the algorithms of N-version programming with majority voting and N-version programming with the consensus voting are given. The authors support it with Boolean composition application to allow one to decide on correct result of versions' outputs.

References

[1] Avizienis A and Chen L 1977 *Proc. Int. Conf. COMPSAC'77* (Chicago, IL) 149–155
[2] Eriş O, Yildirim U, Durmuş M S, Söylemez M T and Kurtulan S 2012 *Proc. CTS 2012* (Sofia, Bulgaria) 177–180
[3] Looker N, Munro M and Xu J 2005 *Proc. COMPSAC 2005* (Edinburgh, Scotland, UK) 66–69
[4] Gruzenkin D V, Chernigovskiy A S and Tsarev R Y 2018 *Advances in Intelligent Systems and Computing* **661** 293–303
[5] Akhil K and Kavindra M 1991 *IEEE Trans. Reliab.* **40(5)** 593–600
[6] Mohamed A and Zulkernine M 2007 *Proc. IEEE Int. Symp. High Assurance Systems Engineering* (Dallas, TX) 267–274
[7] Yacoub S 2003 *Reliab. Eng. Syst. Safe.* **81(2)** 133–145
[8] Ahamad M and Ammar M H 1989 *IEEE Trans. Softw. Eng.* **15(4)** 492–496
[9] Blough D M and Sullivan G F 1990 *Proc. 9th Symp. Reliable Distributed Systems* (Huntsville, AL) 136–145
[10] Brilliant S S, Knight J C and Leveson N G 1989 *IEEE Trans. Softw. Eng.* **15(11)** 1481–1485
[11] Kuncheva L I, Whitaker C J, Shipp C A and Duin R P W 2003 *Pattern Anal. Appl.* **6(1)** 22–31
[12] Lam L and Suen C Y 1997 *IEEE Trans. Syst. Man Cybern. A, Syst. Humans* **27(5)** 553–568
[13] Levitin G 2001 *Reliab. Eng. Syst. Safe.* **73(1)** 91–100
[14] Levitin G and Lisnianski A 2001 *Reliab. Eng. Syst. Safe.* **71(2)** 131–138
[15] McAllister D F, Sun C E and Vouk M A 1990 *IEEE Trans. Reliab.* **39(5)** 524–534
[16] McAllister D and Vouk M 1996 Handbook of Software Reliability Engineering Lyu M. McGraw-Hill 577–603
[17] Parhami B 1994 Voting algorithms *IEEE Trans. Reliab.* **43(4)** 617–629
[18] Xie M and Pham H 2005 Modeling the reliability of threshold weighted voting systems *Reliab. Eng. Syst. Safe.* **87(1)** 53–63
[19] Kim K, Vouk M A and McAllister D F 1998 Fault-tolerant software voters based on fuzzy equivalence relations *Proc. IEEE Aerospace Conf.* (Snowmass at Aspen, CO) pp. 5–19
[20] Dubrova E 2013 Fault-Tolerant Design Springer 157–179