# Conditional Probability Voting Algorithm Based on Heterogeneity of Mimic Defense System

V Rahul

IITH

July 2, 2021

# About the paper

## Authors

- Shuai Wei
- Huihua Zhang
- Wenjian Zhang
- Hong Yu

## Institute

- PLA Strategic Support Force Information Engineering University
- Wuxi Xinwu Confidential Technology Service Center

## Date of Publishing

- October 15, 2020

# Abstract


1. In recent years network attacks have been increasing rapidly, and it is difficult to defend against these attacks, especially attacks at unknown vulnerabilities or backdoors.
2. As a novel method, Mimic defense architecture has been proposed to solve these cyberspace security problems by using the principle of dynamic heterogeneous redundant variants.
3. Choosing appropriate variants and voting algorithm according to heterogeneities of these variants become the key issue of designing mimic defense architecture.
4. This paper analyzes the system failure probability and scalability of 3 different voting algorithms- MHA, MVA, CPVA, and decide which one is the best.

# Variants

Mimic defense system can be considered a restrict version of N-variant systems, because it adopts the basic idea of running multiple variants of the same program in parallel.

1. Variants are usually composed of a series of components, such as CPU, operating system, middleware, application, etc.
2. Each component is composed of several modules. For example, the application can be divided into module 1, module 2,. . ., module M, etc.
3. A module is atomic, and its implementations are different from each other.
4. Each variant can be represented by a module implementation vector $z^i = ({g_1}^i {g_2}^i ... {g_N}^i)$, where N is the number of modules contained in a variant and each module may have several implementations.

# Variants

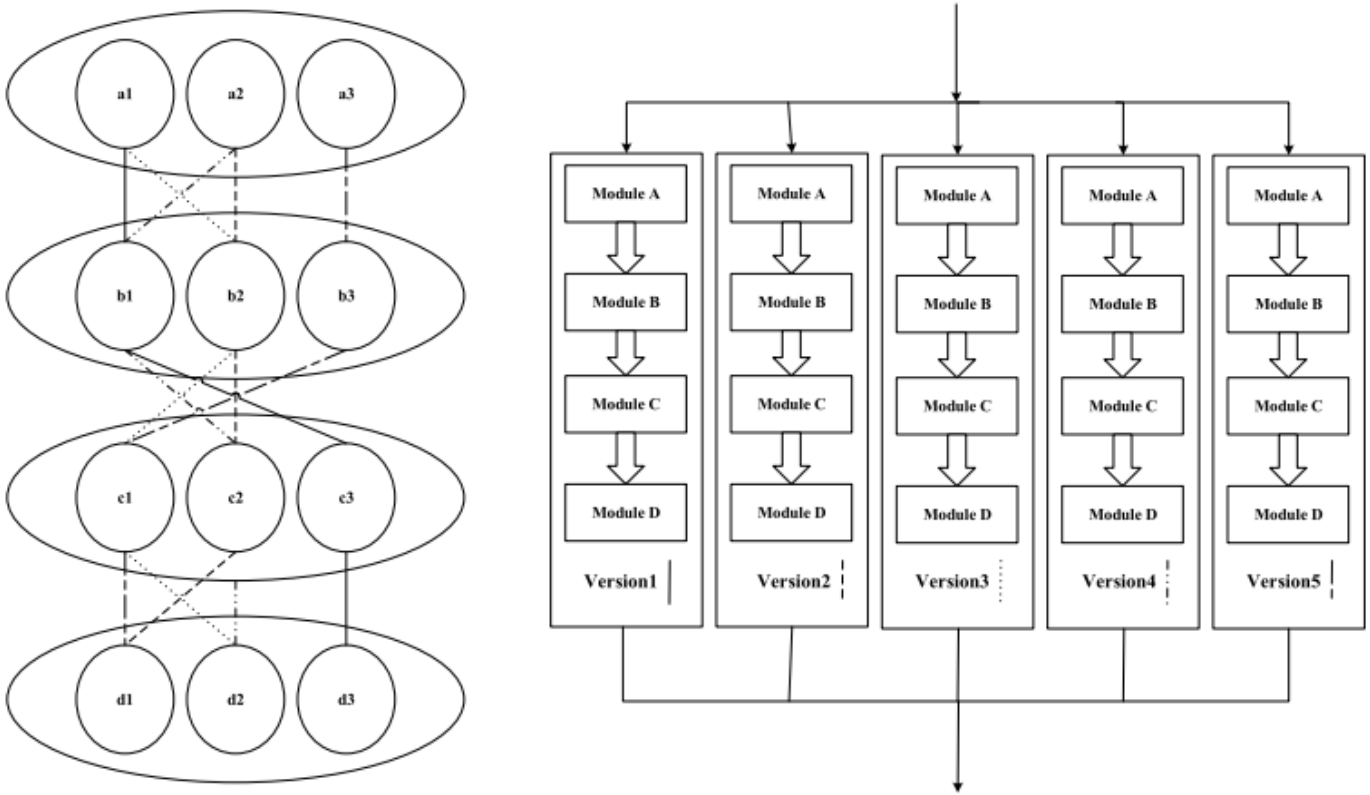


Figure: A typical mimic defense instance

# Variants

The mimic defense system shown in above figure can be described by a matrix as shown below.

$$\begin{pmatrix} z^1 \\ z^2 \\ z^3 \\ z^4 \\ z^5 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_3 & d_3 \\ a_2 & b_2 & c_2 & d_1 \\ a_1 & b_2 & c_1 & d_2 \\ a_2 & b_1 & c_2 & d_2 \\ a_3 & b_3 & c_1 & d_1 \end{pmatrix}$$

# Heterogeneous Variants

1. Mimic defense system requires the variants to be heterogeneous to each other, not just applications, but also including CPU, OS, middleware and so on.
2. For a large system which is common in mimic defense system, it is hard to realize totally heterogeneous.

There are mainly three kinds of algorithms to choose variants:

1. maximum heterogeneous algorithm (MHA)
2. optimal mean distance algorithm (OMDA)
3. random seeds scheduling method

# Heterogeneous Variants

## 2-level similarity

Suppose there are three variants (1, 2, 3), the 2-level similarities are referred to the similarities for 1&2,1&3,1&2. The sum of similarities is lower, the system is considered to be safer.

1. As the number of working variants grows, 2-level similarity become less important.
2. Heterogeneity will reach max when there are 3 variants in the mimic system, and the heterogeneity will drop as variants number increase more than 3.

# Classic Model

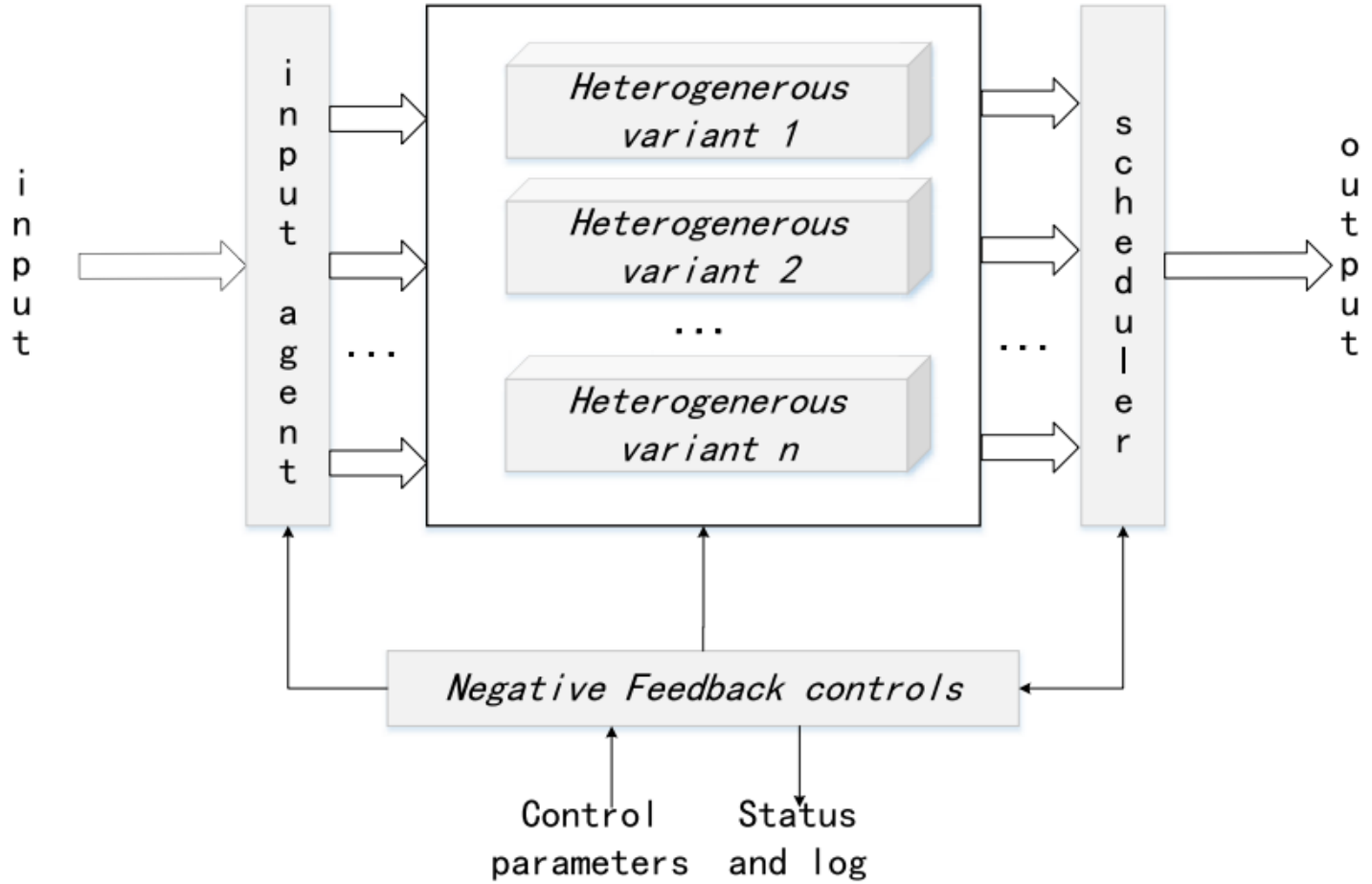


Figure: Classic model of mimic defense architecture

# Assumptions

1. Input agent, Scheduler, and Feedback controller are safe from network attacks.
2. Each vulnerability/backdoor of the system have the same probability being attacked.
3. Different implementations of each module have the same attacking probability.
4. Only one vulnerability / back door can be attacked at a time, and the attacked variant will be cleaned in a short time.

The probability of successfully attacking each module is $\beta_i$ , which should satisfy $\sum_i \sum_{\psi_i} \beta_i = 1$, $\psi_i$ is the number of implementations for module i.

# Binary Division Vectors

## Module diversity $\psi_i$

The implementation number of the module $g_i$, which can be calculated by formula $|\bigcup_i {g_k}^i|$. Implementation set is $(a_1 a_1 a_1 a_2 a_3)^T$, then union the contents and get the set $\{a_1 a_2 a_3\}$, which contain 3 elements, so the diversity of module a is 3.

## Binary division vector ${\eta_i}^k$

1. In a module, divide the same implementations into one group and other implementations into another group, and use a vector to represent. Assign corresponding values in the vector of the same implementations to 1 and others to 0.
2. For example, the implementation of $g_1$ is $(a_1 a_1 a_1 a_2 a_3)^T$, then there are 3 binary division vectors for module $g_1$, the binary division vector of $a_2$ is $(00010)^T$, $a_3$ is $(00001)^T$, $a_1$ is $(11100)^T$.

# Binary Division Vectors

## Complement binary division vector ${\sim\eta_i}^k$

which is reversing every element in the binary division vector. Based on the binary division vector $(11100)^T$ of module implementation $a_1$, reverse all the elements in it, and its complement vector will be $(00011)^T$.

## Isomorphic number of binary division vector ${\lambda_i}^k$

The number of elements whose value is equal to 1. There are three 1 in the binary division vector of module implementation $a_1$, which is $(11100)^T$, then there are three $a_1$, and Isomorphic number of $(11100)^T$ is 3.

# Important Property

1. Assuming that there are T executions in the mimic defense system, there will be at most 1 implementation whose isomorphic number is not less than $\left\lfloor \frac{T+1}{2} \right\rfloor$.
2. If the diversity of a module is 2, there must be one implementation whose isomorphic number is not less than $\left\lfloor \frac{T+1}{2} \right\rfloor$.

The following conditions are generally required in order to reduce the failure probability of mimic defense system :

1. Do not add the same implementation of a module so that its Isomorphic number exceeds $\left\lfloor \frac{T+1}{2} \right\rfloor$.
2. Add different implementation of a module or balance the same implementation of a module so that its maximum implementation is less than $\left\lfloor \frac{T+1}{2} \right\rfloor$.

# Majority voting algorithm

1. Divide the variants by their results, put variants with the same result into a group $G_k$ . According to the hypothesis only one vulnerability/backdoor is attacked at a time, so there are usually 2 groups, suppose they are $G_1$ and $G_2$.
2. If $|G_1| > |G_2|$, then select the result of G1 as the final output; otherwise, select the result of G2 as the final output.
3. Clean the variants which have been arbitrated to be abnormal.

# Majority voting algorithm

1. $V_g = \text{NULL}$
2. for i $= 1:$ N
3. for k $= 1:$ $\psi_i$
4. If($\lambda_i{}^k \geq \lfloor \frac{T+1}{2} \rfloor$)
5. add i in $V_g$
6. endfor
7. endfor
8. msum $= 0$
9. for each index in $U_g$
10. msum $=$ msum $+ \beta_k$
11. endfor

# Conditional probability voting algorithm

1. The variants which generated the same results are divided into one group $G_k$ , generally there are only two groups, assumed as $G_1$ and $G_2$.
2. Calculation $\beta_{G_1}$ and $\beta_{G_2}$, $\beta_{G_1} = \sum_k \beta_k$, if k $\in \bigcap_{i \subset G_1} (z^i - \bigcap_{j \subset G_2} z^j)$, $\beta_{G_2} = \sum_k \beta_k$, if k $\in \bigcap_{i \subset G_2} (z^i - \bigcap_{j \subset G_1} z^j)$.
3. If $\beta_{G_1} > \beta_{G_2}$, the result of $G_2$ shall be used, otherwise, the result of $G_1$ shall be used.
4. Clean the abnormal variants which have generated wrong result.

# Conditional probability voting algorithm

1. $G_u$ = NULL
2. for i = 1: N
3. for k = 1: $\psi_i$
4. if($\lambda_i{}^k \geq \lfloor \frac{T+1}{2} \rfloor$)
5. $G_u$ = union($G_u$,$\eta_i{}^k$)
6. endif
7. endfor
8. endfor
9. csum = 0
10. $G_L = G_S$ = NULL
11. for each vctorl in $G_u$
12. for i = 1: N
13. for k = 1: $\psi_i$
14. if(vctorl == $\eta_i{}^k$)
15. add i in $G_L$
16. else if($\sim vctorl == \eta_i{}^k$)
17. add i in $G_S$
18. endif
19. endfor
20. endfor
21. lsum = ssum = 0
22. for each k in $G_L$
23. lsum = lsum + $\beta_k$
24. endfor
25. for each k in $G_S$
26. ssum = ssum + $\beta_k$
27. endfor
28. if *lsum* < *ssum*
29. csum=csum+lsum
30. else
31. csum=csum+ssum
32. endif
33. endfor

# Results for 3-variants experiment

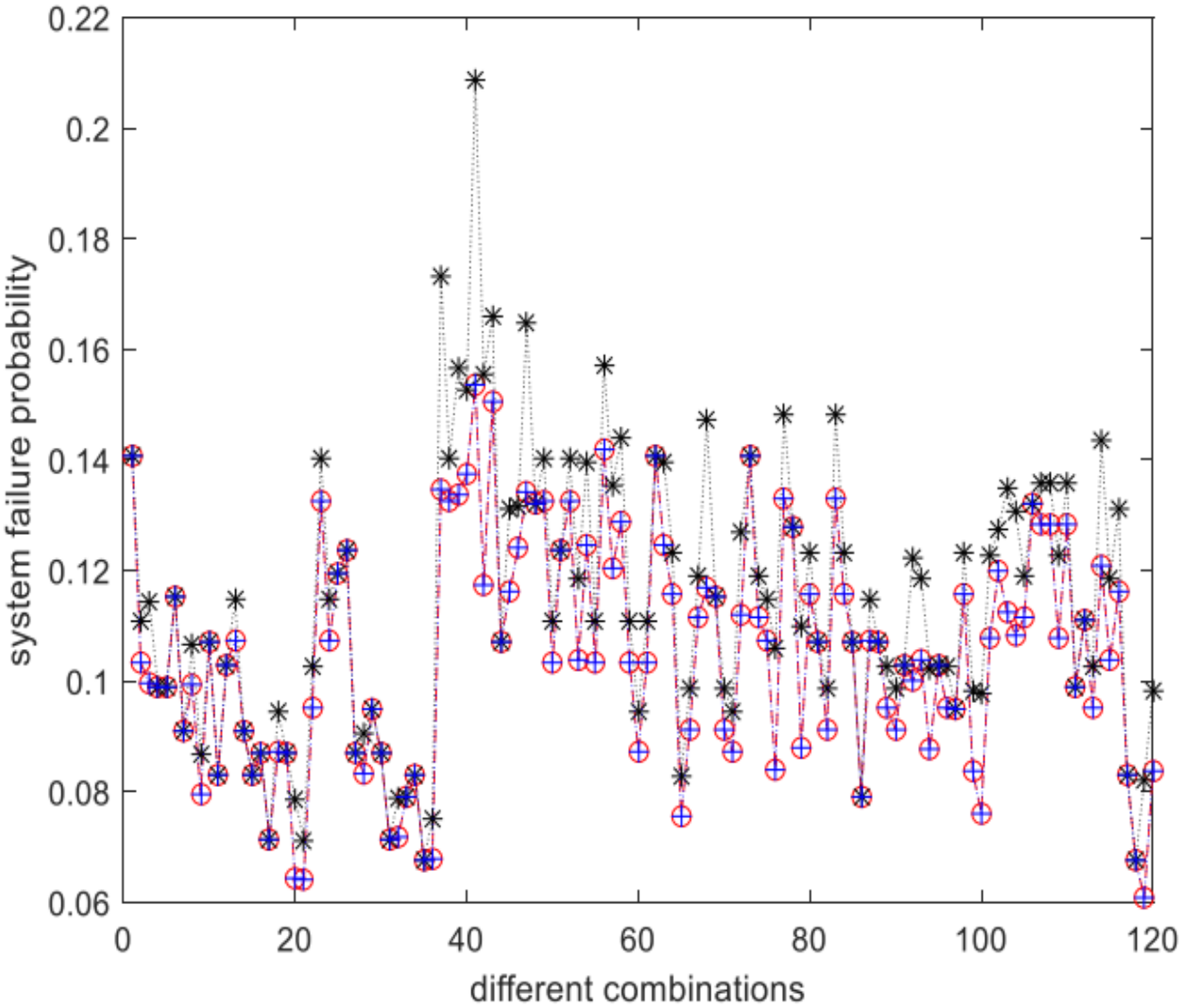


Figure: System failure probabilities of MHA, MVA and CPVA when N=100 M=10

# Results for 5-variants experiment

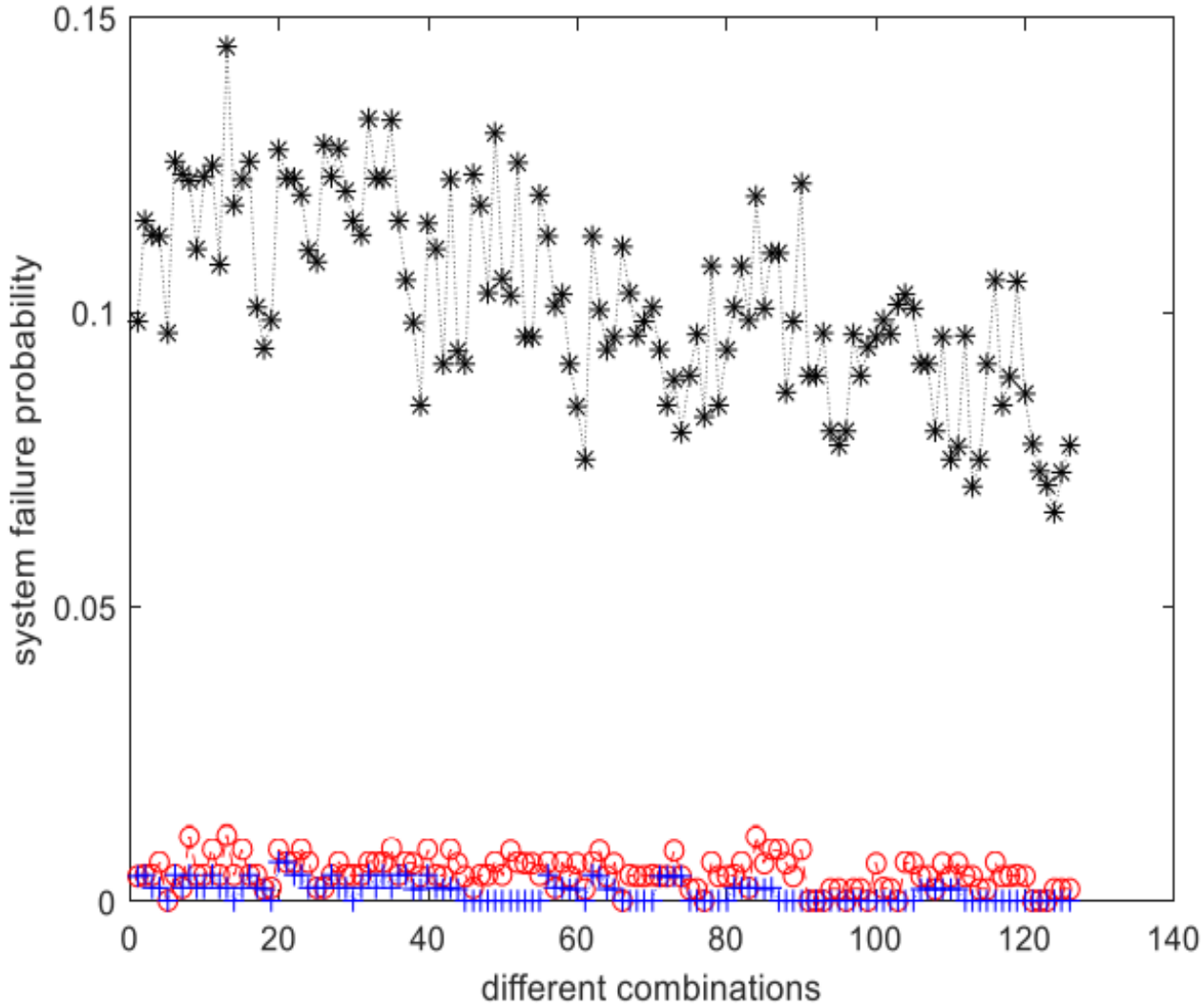


Figure: System failure probabilities of MHA, MVA and CPVA when N=100 M=10

# Results for scalability experiment

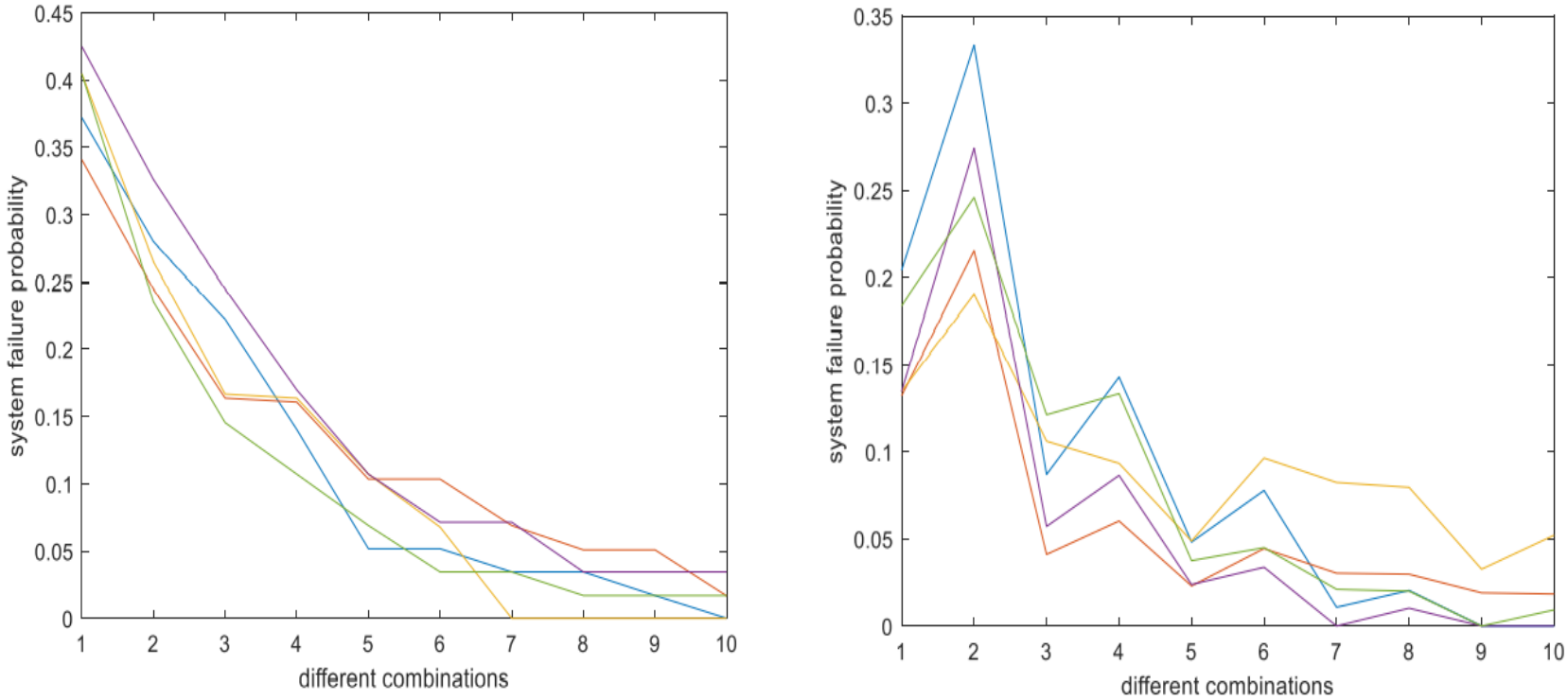


Figure: System failure probabilities of CPVA and MVA with variants increase