IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received December 25, 2020, accepted January 23, 2021, date of publication January 26, 2021, date of current version February 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3054760

EM Algorithm for Estimating Reliability of Multi-Release Open Source Software Based on General Masked Data

JIANFENG YANG[1,3], JING CHEN[2], AND XIBIN WANG[1]
[1]School of Data Science, Guizhou Institute of Technology, Guiyang 550003, China
[2]College of Information Engineering, Guizhou University of Traditional Chinese Medicine, Guiyang 550025, China
[3]Special Key Laboratory of Artificial Intelligence and Intelligent Control of Guizhou Province, Guiyang 550003, China

Corresponding authors: Jianfeng Yang (yjf@git.edu.cn) and Jing Chen (gogochen06@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 71901078, in part by the Science and Technology Foundation of Guizhou under Grant QianKeHeJiChu[2020]1Y269, and in part by the Special Key Laboratory of Artificial Intelligence and Intelligent Control of Guizhou Province under Grant QianJiaoHeKY[2020]001.

ABSTRACT Multi-release is critical for modern open source software product in order to satisfy more customer requirements. Masked data, a kind of missing data, is the system failure data when the exact cause of the failures might be unknown. That is, the cause of the system failures may be any one of the objects. However, due to the influence of the test strategy in real project, the cause of the system failures may be a subset of the system objects, not any one of the objects. In this paper, the mathematical description of general masked data is presented based on the traditional masked data. Furthermore, a novel multi-release open source software (OSS) reliability model based on general masked data is proposed. Different from traditional multi-release OSS reliability model, the proposed approach is based on additive model with general masked data other than change point model. And then, the maximum likelihood estimation (MLE) process of the model parameters is derived in detail, and expectation maximization (EM) algorithm is used to solve the extremely complicated problem of the log-likelihood function. Finally, two data sets from real open source software project are applied to the proposed approach, and the results show that the proposed reliability model is useful and powerful.

INDEX TERMS General masked data, multi-release open source software, reliability model, maximum likelihood estimation, EM algorithm.

NOTATIONS

k	the number of objects (releases) in software system
i	the release number, $i = 1, 2, \cdots, k$
j	the observation number, $j = 1, 2, \cdots, m$
$N_i(t)$	counting number of failures for release i at time t
$N(t)$	counting number of failures for system at time t
N_j^i	the number of failures in interval $(t_{j-1}, t_j]$ due to release i
$m_i(t)$	mean value function of failure process for release i
$m(t)$	mean value function of failure process for system
$\lambda(t)$	failure intensity function for system
n_j^i	the observed number of failures in interval $(t_{j-1}, t_j]$ known due to release i
n_j^M	the observed number of failures that are masked in interval $(t_{j-1}, t_j]$
n_j	the observed number of failures for system in interval $(t_{j-1}, t_j]$
m_j	the observed cumulative number of failures for system until t_j
S_j	failure cause set (FCS), $S_j \subseteq 1, 2, \cdots, k$
θ_i	parameter vector of model for release i
τ_i	the release time for release i

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun Li.

I. INTRODUCTION

In order to develop high-quality, high-security, and satisfactory products for software users, software companies spend a lot of money to test the software, remove fault and improve the reliability of the software. In the process of software testing, it is usually assumed that the cumulative number of failures follows the non-homogeneous Poisson process (NHPP). This type of model is called NHPP-based software reliability growth model [1]–[5]. However, many factors affect the reliability growth of traditional closed source software, such as software complexity, error in requirements, test efficiency, test intensity, fault detection rate, fault exposure rate, fault remove and correction rate, fault introduction, change point, user behaviour, environmental factors, etc. Compared with the traditional closed source software development process, open source software has many characteristics. The development process of open source software is mainly mastered by community engineers. Many research results have been published on specific software reliability growth model(SRGM) for investigating the reliability of open source software (OSS) with a single release, which is a growing area of software development and applications. For example, Tamura and Yamada proposed a software reliability growth model based on stochastic differential equations [6]. Later, Tamura and Yamada proposed a method of software reliability assessment for the embedded OSS with flexible hazard rate modeling [7]. Luan and Huang proposed a modified Pareto-based distribution(PD) OSS reliability model, called the single-change-point 2-parameter generalized PD(SCP-2GPD) model [8], and a special form of the Generalized Pareto-based distribution model, named the Bounded Generalized Pareto distribution (BGPD) model, is further proposed to investigate the fault distributions of Open Source Software [9]. Ullah *et al.* proposed a method that selects the SRGM, which among several alternative models best predicts the reliability of the OSS, in terms of residual defects [10], [11]. Recently, a new proposed model considering the decreasing trend of fault detection rate is developed to effectively improve OSS reliability [12].

Staying competitive in the market and keep profitable for a software product unlikely happen in this increasing-innovational society if only has a single release especially when rival has a new release carrying more attractive features and satisfying more customer requirements [13]. Since multi-release is critical for modern software product, release planning is becoming a popular research topic in the past few years. Nevertheless, most of the proposed model only can be applied on a single release. It is thus necessary to investigate changes in reliability arising from ongoing releases, which is a rather complex problem as usually there are many reasons for a new release. Only a few researches studied multi-release software reliability. For example, Li et al proposed a modified non-homogeneous Poisson process model for open source software reliability modeling and analysis, optimal version-updating for open source software is investigated as well [14]. Hu *et al.* considered a scenario in which a software development team develops, tests, and releases software version by version, and proposed a number of practical assumptions [15]. Kapur *et al.* proposes a mathematical modeling framework for multiple releases of software products, and the model takes into consideration the combined effect of schedule pressure and resource limitations using a Cobb Douglas production function in modeling the failure process using a software reliability growth model [16]. Pachauri *et al.* proposed a modeling framework considering the inflection S-shaped fault reduction factor and extended this model into multi-release software [17]. Yang *et al.* investigated the failure processes in testing multi-release software by taking into consideration of the delays in fault repair time based on a proposed time delay model [18]. Ahmadi *et al.* proposed a multi up-gradation reliability model for open source software incorporating bugs removed from two different phases, namely a pre-commit test and parallel debugging test [19]. Singh *et al.* developed a Non-Homogeneous Poisson Process model for Open Source Software to understand the fixing of issues across releases, and optimal release-updating using entropy and maximizing the active user's satisfaction level subject to fixing of issues up to a desired level, is investigated as well [20]. Zhu *et al.* proposed a multi-release software reliability model with consideration of the remaining software faults from previous release and the new introduced-faults, and dependent fault detection process is taken into account in this model [21]. Last year, a method to evaluate reliability and maintainability of OSS by using both code-based and community-based aspects is proposed [22].

Large open source software is often composed of many components or subsystems. In order to make full use of the failure data of the components, the failure data of components can be used to build a software reliability model. It is well known that the additive NHPP-based model is an important reliability model for estimating system reliability using failure data of components. The hyper-exponential NHPP model proposed by Ohba [23] was one of this kind, in which the ordinary models were G-O model proposed by Goel and Okumoto [24]. A similar version of the hyper-exponential model is also studied by Yamada *et al.* [25]. Xie and Goh developed a system reliability growth analysis method using component failure data [26]. Furthermore, an additive Weibull model from Xie and Lai [27], Burr XII model from Wang [28] and power-law model from [29] by using the component failure data. Because there are many parameters in the additive model, how to effectively estimate the model parameters is the main problem of this type of model.

The above additive reliability model cannot consider the masked data. Masked data are the system failure data when the exact cause of the failures might be unknown. That is, the cause of the system failures may be any one of the components (modules, subsystem, object, etc.) [29]. Many research results have been obtained for hardware reliability analysis under masked data [30]–[33], but there are few research results for software reliability based on masked data. The observed failure data is incomplete, that is, there is masked in the failure data. At this time, the software reliability additive model cannot be decomposed into several simple NHPP models, so it is difficult to estimate the parameters. For the first time, Zhao established an additive software reliability model under masked data and used maximum likelihood estimation to estimate the parameters. He used EM algorithm to find the approximate value of the parameter estimates [29]. It is well known, software reliability assessment methods have been changed from closed to open source software, and maximum likelihood estimation is an effective estimation method in engineering application [34], [35]. Therefore, a modified additive reliability model for multi-release open source software using general masked data (GMA) is proposed in this paper. Moreover, the general masked data is generalization of traditional masked data in Zhao's research in order to estimate reliability with multi-release versions.

The remainder of this paper is organized as follows. Section II reviews the additive NHPP-based reliability models, and discusses the general masked data. In addition, a novel multi-release OSS reliability model based on general masked data is proposed in this section. Section III gives the MLE process of the model parameters with general masked data, and EM algorithm is used to solve the extremely complicated problem of the log-likelihood function. Section IV gives two numerical examples with real open source software using grouped general masked data, employing the proposed models. Finally, Section V concludes this paper.

II. MULTI-RELEASE OPEN SOURCE SOFTWARE RELIABILITY MODEL

A. REVIEW OF ADDITIVE NHPP RELIABILITY MODEL

In general, an additive NHPP software reliability model is set up based on the following assumptions [29]:

1) The software contains k components.
2) The counting number of detected faults in component i at time t, denoted by $\{N_i(t), t \geq 0\}$, is characterized by NHPP with mean value function $m_i(t)$, $i = 1, 2, \cdots, k$.
3) $\{N_i(t), t \geq 0\}, i = 1, 2, \cdots, k$ are statistical independent.
4) The cumulative number of system failures, say $N(t)$, is given by:

$$N(t) = \sum_{i=1}^{k} N_i(t) \tag{1}$$

The mean value function (MVF) for NHPP $\{N_i(t), t \geq 0\}$ is then given by:

$$m(t) = \sum_{i=1}^{k} m_i(t) \tag{2}$$

The failure intensity function of software system is given by:

$$\lambda(t) = m'(t) = \sum_{i=1}^{k} m_i'(t) \tag{3}$$

Note that a component may be a subsystem, a module, or a failure mode in this model. According to the above assumptions and characters of NHPP, the reliability function of software system under the additive NHPP model is therefore given by

$$\begin{aligned} R(t) &= P\{N(t) - N(0) = 0\} \\ &= \exp\{-[m(t) - m(0)]\} \\ &= \exp\left\{-\sum_{i=1}^{k} m_i(t)\right\} \end{aligned} \tag{4}$$

Additionally, the probability of no failure happens during time interval $(t, t + \Delta t)$ can be calculated by:

$$R(\Delta t \,|t) = \exp\{-[m(t + \Delta t) - m(t)]\} \tag{5}$$

Below we briefly show that some classical SRGM based on NHPP, such as the Goel-Okumoto model [24], the Yamada delayed S-shaped model [36] and the generalized Goel NHPP model [37]. Table 1 gives the SRGM corresponding to mean value function (MVF) and failure intensity function (FIF) [1].

TABLE 1. Some classical SRGM corresponding to MVF and FIF.

No.	SRGM	MVF	FIF
#1	Goel-Okumoto Model (GO Model)	$a\left(1-e^{-bt}\right)$	abe^{-bt}
#2	Yamada Delayed S-shaped Model (DSS Model)	$a\left[1-\left(1+bt\right)e^{-bt}\right]$	ab^2te^{-bt}
#3	Generalized Goel Model (GGO Model)	$a\left[1-\exp\left(-bt^c\right)\right]$	$abct^{c-1}e^{-bt^c}$

B. GENERAL MASKED DATA

The masked data are the system failure data when the exact causes of the failures, i.e., the components that have caused the system failure, may be unknown. Note that a component may be a subsystem, a module, an object, or a failure mode in this model. Occasionally, the failure report provided by the testing team may not give us complete information on the types of failures. For example, the component that causes a system failure during system-level testing may not be identified or omitted in the failure report. Additionally, the failures due to errors in the interfaces between modules cannot be said to belong to a specific module. Another example is that the field data do not contain complete component failure information for economic reasons or human error. The analysts often collect a lot of field data and hope to make use of such extra information. Unfortunately, the failure data do not contain complete information on failure modes. It is a common phenomenon that the failure reporting from field does not provides the details of interest. Therefore, the masking phenomenon is often appeared in collecting field component failure data, especially for large software product. In such cases the components that may cause a system failure are said to be masked [29].

Zhao and Xie assumed that the cause of the system failures may be any one of the components to build the additive reliability model [29]. However, due to the influence of the test strategy in real project, the cause of the system failures may be a subset of the system components, not any one of the components. Therefore, the general masked data is defined in this paper based on existed theory, and is given as follows.

Definition 1: Suppose that the software contains k objects (subsystem, component, module, or failure mode), and $S = \{1, 2, \cdots, k\}$ is the object set in software system. Let $S_j \subseteq \{1, 2, \cdots, k\} (j = 1, 2, \cdots, m)$, and S_j contains the cause of the system failures at time t_j, named Failure Cause Set (FCS). Then the general masked data is defined by $\left(k, t_j, S_j\right)$.

According to Definition 1, when one failure arrives, the subset S_j and failure arrival time can be observed. Note that an object may be a subsystem, a component, a module, or a failure mode. For example, the software testing strategy is carried out using function testing, and function module is related to object $\{1, 2, 3\}$. In this case, if a masked failure arrives at time t_j, then S_j $\{1, 2, 3\}$. It is easy to know, if $S_j = \{s\}$ $(s = 1, 2, \ldots, k)$, then we know that the cause of failure is not masked. If $S_j = \{1, 2\}$, we have that the exact cause of failure is masked, and the cause of the system failures may be object 1 or object 2.

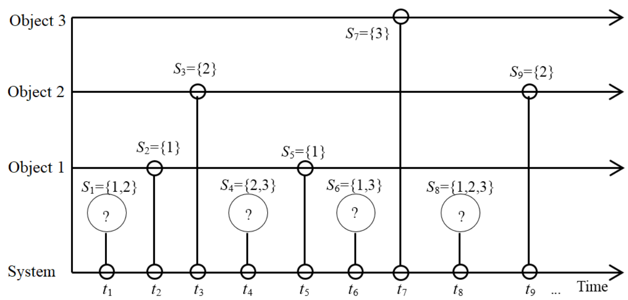

FIGURE 1. An example of failure process with general masked data for a software system of three objects.

As shown in Figure.1, an example of failure process with general masked data is described for a software product with three objects. It is shown that system failed at time t_1 and $S_1 = \{1, 2\}$, both objects 1 and 2 may cause the system failure. This is a masked data since it is impossible to determine which object is the cause. Furthermore, system failed at time t_2 and the cause of system failure is object 1, that is, there is not masked data. It is easy to know that the cause of the system failures are masked at time t_4, t_6, t_8 and not masked at time t_3, t_5, t_7, t_9.

Suppose that the software system has the grouped failure data at sequential observation times $t_1 < t_2 < \cdots < t_m$. The general masked data have the form as shown in the following table. In Table 2, n_{Mj} and n_{ij} are the numbers of the masked and ith object tested at observation time t_j, respectively, $i = 1, 2, \ldots, k; j = 1, 2, \ldots, m$. If there exist the general masked data as illustrated in Table 2, the failure process for the system cannot be decomposed into the simple object processes. Using maximum likelihood estimation or least squares estimation, the parameters of objects have to be estimated based on the overall objective function instead

TABLE 2. Grouped general masked data of software system with k objects.

Causes of System Failures	Observation Times					
	t_1	t_2	$\cdots$	t_j	$\cdots$	t_m
Unknown or Masked	(n_1^M, S_1)	(n_2^M, S_2)	$\cdots$	(n_j^M, S_j)	$\cdots$	(n_m^M, S_m)
Object 1	n_1^1	n_2^1	$\cdots$	n_j^1	$\cdots$	n_m^1
Object 2	n_1^2	n_2^2	$\cdots$	n_j^2	$\cdots$	n_m^2
$\vdots$	$\vdots$			$\vdots$		$\vdots$
Object i	n_1^i	n_2^i	$\cdots$	n_j^i	$\cdots$	n_m^i
$\vdots$	$\vdots$			$\vdots$		$\vdots$
Object k	n_1^k	n_2^k	$\cdots$	n_j^k	$\cdots$	n_m^k

of the objective functions for objects. Therefore, common techniques for maximizing or minimizing a multivariate nonlinear function are not easily used because there may exist so many unknown parameters.

C. MULTI-RELEASE OPEN SOURCE SOFTWARE RELIABILITY MODEL WITH GENERAL MASKED DATA

Modern software systems are with increasing complexity and these large systems are generally object-based. Moreover, since the new release has more attractive features and satisfies more customer requirements, is critical for modern software product. Additive model is not only one of important approach in object-based software reliability analysis, but also an important model for multi-release software reliability analysis. In general, additive NHPP-based software reliability model is set up based on the following assumptions. Due to the complexity of the testing environments, some other assumptions are also required in order to be able to model and analyze multi-release OSS reliability. The formulation of proposed model is based on the following assumptions:

1) The open source software contains k releases. It means that there are totally k releases. Failures data of each release are observed, and some may be masked. Denote τ_i $(\tau_1 \leq \cdots \leq \tau_k)$ is version-update time of release i.

2) The counting number of detected faults for release $i\,(i = 1, 2, \cdots, k)$ at time t, denoted by $\{N_i(t), t \geq \tau_i\}$, is characterized by NHPP with mean value function $m_i\,(t)$.

3) $\{N_i(t), t \geq \tau_i\}$ are statistical independent during the testing phase.

4) If any of the release version fail, the software system fails.

5) The cumulative number of system failures $N(t) = \sum_{i=1}^{k} N_i(t)$.

Based on the above assumptions, the mean value function $m(t)$ (Expecting cumulative number of failures for software system) for NHPP $\{N\,(t)\,, t \geq 0\}$ is given by

$$m(t) = \sum_{i=1}^{k} m_i(t) \tag{6}$$

where

$$m_i(t) = \begin{cases} 0, & t < \tau_i \\ m_i(t - \tau_i), & t \geq \tau_i \end{cases} \tag{7}$$

TABLE 3. Three selected additive model corresponding to MVF.

Proposed Model	Description	MVF
GOGM Model	Additive GO model with general masked data using EM algorithm	$m(t)=\sum_{i=1}^{k}\begin{cases}0 & ,t<\tau_i\\ a_i\left(1-e^{-b_i(t-\tau_i)}\right) & ,t\geq\tau_i\end{cases}$
DSSGM Model	Additive DSS model with general masked data using EM algorithm	$m(t)=\sum_{i=1}^{k}\begin{cases}0 & ,t<\tau_i\\ a_i\left[1-\left(1+b_i\left(t-\tau_i\right)\right)e^{-b_i(t-\tau_i)}\right] & ,t\geq\tau_i\end{cases}$
GGOGM Model	Additive GGO model with general masked data using EM algorithm	$m(t)=\sum_{i=1}^{k}\begin{cases}0 & ,t<\tau_i\\ a_i\left[1-\exp\left(-b_i\left(t-\tau_i\right)^{c_i}\right)\right] & ,t\geq\tau_i\end{cases}$

The failure intensity function of system is given by

$$\lambda(t)=\sum_{i=1}^{k}\lambda_i(t)=\sum_{i=1}^{k}m_i'(t) \tag{8}$$

Note that a software release version can be recorded as an object, therefore the object mentioned below in this paper means a software release version. Moreover, an object may be a subsystem, a module, or a failure mode in the future generalized reliability model.

According to the above assumptions and characters of NHPP, the reliability function of software system under the additive NHPP model is therefore given by

$$\begin{aligned}R(t)&=P\{N(t)-N(0)=0\}\\&=\exp\{-[m(t)-m(0)]\}\\&=\exp\left\{-\sum_{i=1}^{k}m_i(t)\right\}\end{aligned} \tag{9}$$

Additionally, the probability of no failure happens during time interval $(t, t+\Delta t)$ can be calculated:

$$R(\Delta t\,|t)=\exp\{-[m(t+\Delta t)-m(t)]\} \tag{10}$$

When release times $\tau_i=0$, the proposed model becomes the existed model proposed by Zhao and Xie [29]. Especially, GO model with general masked data (GOGM Model), DSS model with general masked data(DSSGM Model) and GGO model with general masked data(GGOGM Model) are proposed using EM algorithm, as shown in Table 3. Table 3 also shows the mean value functions (MVF) of selected models. Furthermore, EM algorithm will be described in the following Section. Without a doubt, the proposed model in this paper can be extended to other NHPP-based SRGMs.

III. MAXIMUM LIKELIHOOD ANALYSIS OF SOFTWARE RELIABILITY WITH GROUPED GENERAL MASKED DATA

Two commonly used methods in parameter estimation are the Maximum Likelihood Estimation (MLE) and Least Squares Estimation (LSE) methods. However, when the masked data are present, the failure process for the system cannot be decomposed into the simple object processes. Moreover, the objective function in MLE and LSE becomes a complex multivariable function with a very high dimension. For example, GOGM model from Table 3 for software system contained k releases has totally $2k$ parameters to be estimated simultaneously from failure data. Therefore, common techniques for maximizing or minimizing a multivariate nonlinear function are not easily used because there may exist so many unknown parameters. Fortunately, Zhao and Xie applied Expectation Maximization(EM) algorithm to solve the problem of maximum likelihood estimation with masked data, and it is shown that the EM algorithm is powerful to deal with the masked data [29]. But, maximum likelihood estimation with general masked data is more complicated than traditional maximum likelihood estimation with masked data in Zhao's research. In the following Sections, the maximum likelihood estimation process of the model parameters is derived in detail, and EM algorithm is used to solve the extremely complicated problem of the log-likelihood function.

A. MAXIMUM LIKELIHOOD ESTIMATION WITH GENERAL MASKED DATA

The grouped failure data is the cumulative number of failures in time interval. In order to describe the observed data clearly, the following notation is given

$$S_j^*=\begin{cases}\emptyset, & \text{no masked}\\ S_j, & \text{masked}\end{cases} \tag{11}$$

Assume that the failure process is observed at time points $0=t_0<t_1<t_2<\cdots<t_m$, the observed data with general masked data based on Table 2 are

$$\begin{aligned}&\left(n_1^1, n_1^2, \cdots, n_1^k, n_1^M, S_1^*\right),\\&\left(n_2^1, n_2^2, \cdots, n_2^k, n_2^M, S_2^*\right), \cdots,\\&\left(n_m^1, n_m^2, \cdots, n_m^k, n_m^M, S_m^*\right)\end{aligned} \tag{12}$$

where n_j^i is the number of failures in interval $\left(t_{j-1}, t_j\right]$ known due to release i, n_j^M is the number of failures that are not identified corresponding to $S_j^*, i = 1, 2, \ldots, k; j = 1, 2, \ldots, m$, i denotes the release number and j denotes the observation number. It means that there are totally k releases, and there are m number of observations. It is easy to know $S_j^* = \emptyset$ if and only if $n_j^M = 0$.

Denote N_j^i is the random variable of the number of failures in $\left(t_{j-1}, t_j\right]$ due to release i. It is well known that N_j^i are independent Poisson distributed with mean value function $m_i(t_j) - m_i(t_{j-1})$. Let

$$\lambda_{ij} = m_i(t_j) - m_i(t_{j-1}) \tag{13}$$

We only observed the occurrences of successive events:

$$\begin{aligned} A_j &= \left\{\cap_{i \notin S_j^*}\left\langle N_j^i = n_j^i\right\rangle, \cap_{i \in S_j^*}\left\langle N_j^i \geq n_j^i\right\rangle, \sum_{i=1}^{k} N_j^i = n_j\right\} \\ &= \left\{\cap_{i \notin S_j^*}\left\langle N_j^i = n_j^i\right\rangle, \cap_{i \in S_j^*}\left\langle N_j^i \geq n_j^i\right\rangle, \sum_{i \in S_j^*} N_j^i = n_j^*\right\} \end{aligned} \tag{14}$$

where

$$n_j = \sum_{i=1}^{k} n_j^i + n_j^M, \; n_j^* = \sum_{i \in S_j^*}\left(n_j^i\right) + n_j^M = n_j - \sum_{i \notin S_j^*} n_j^i, \quad j = 1, 2, \tag{15}$$

The probability of events A_j are calculated by:

$$\begin{aligned} P\left(A_j\right) &= P\left\{\cap_{i \notin S_j^*}\left\langle N_j^i = n_j^i\right\rangle\right\} \cdot P\left\{\left(\cap_{i \in S_j^*}\left\langle N_j^i \geq n_j^i\right\rangle\right) \cdot \left(\sum_{i \in S_j^*} N_j^i = n_j^*\right)\right\} \\ &= \prod_{i \notin S_j^*} P\left\langle N_j^i = n_j^i\right\rangle \cdot P\left\{\left(\cap_{i \in S_j^*}\left\langle N_j^i \geq n_j^i\right\rangle\right) \cdot \left(\sum_{i \in S_j^*} N_j^i = n_j^*\right)\right\} \end{aligned} \tag{16}$$

where

$$P\left\langle N_j^i = n_j^i\right\rangle = \frac{\lambda_{ij}^{n_j^i}}{n_j^i!} \cdot \exp\left(-\lambda_{ij}\right) \tag{17}$$

and

$$P\left\{\left(\cap_{i \in S_j^*}\left\langle N_j^i \geq n_j^i\right\rangle\right) \cdot \left(\sum_{i \in S_j^*} N_j^i = n_j^*\right)\right\} = \exp\left(-\sum_{i \in S_j^*} \lambda_{ij}\right) \cdot \sum_{\cap_{r \in S_j^*}\left\langle i_r \geq n_j^r\right\rangle, \sum_{r \in S_j^*} i_r = n_j^*} \left(\prod_{r \in S_j^*} \frac{\lambda_{rj}^{i_r}}{i_r!}\right) \tag{18}$$

According to the formulas (16), (17) and (18), $P\left(A_j\right)$ has the following form:

$$P(A_j) = \prod_{i \notin S_j^*}\left(\frac{\lambda_{ij}^{n_j^i}}{n_j^i!} \cdot \exp\left(-\lambda_{ij}\right)\right) \cdot \exp\left(-\sum_{i \in S_j^*} \lambda_{ij}\right) \cdot \sum_{\cap_{r \in S_j^*}\left\langle i_r \geq n_j^r\right\rangle, \sum_{r \in S_j^*} i_r = n_j^*} \left(\prod_{r \in S_j^*} \frac{\lambda_{rj}^{i_r}}{i_r!}\right) \tag{19}$$

Therefore, the overall likelihood function given observation $\left(n_j^i, n_j^M, S_j^*, i = 1, 2, \ldots, k; j = 1, 2, \ldots, m\right)$ is

$$\begin{aligned} L(\cdot \left| n_j^i, n_j^M, S_j^*\right.) &= \prod_{j=1}^{m} P(A_j) \\ &= \prod_{j=1}^{m}\left\{\prod_{i \notin S_j^*}\left(\frac{\lambda_{ij}^{n_j^i}}{n_j^i!} \cdot \exp\left(-\lambda_{ij}\right)\right) \cdot \exp\left(-\sum_{i \in S_j^*} \lambda_{ij}\right) \cdot \sum_{\cap_{r \in S_j^*}\left\langle i_r \geq n_j^r\right\rangle, \sum_{r \in S_j^*} i_r = n_j^*} \left(\prod_{r \in S_j^*} \frac{\lambda_{rj}^{i_r}}{i_r!}\right)\right\} \\ &= \prod_{j=1}^{m}\left\{\prod_{i \notin S_j^*}\left(\frac{\lambda_{ij}^{n_j^i}}{n_j^i!}\right) \cdot \exp\left(-\sum_{i=1}^{k} \lambda_{ij}\right) \cdot \sum_{\cap_{r \in S_j^*}\left\langle i_r \geq n_j^r\right\rangle, \sum_{r \in S_j^*} i_r = n_j^*} \left(\prod_{r \in S_j^*} \frac{\lambda_{rj}^{i_r}}{i_r!}\right)\right\} \end{aligned} \tag{20}$$

Then the log-likelihood function has the following form:

$$\begin{aligned} &\log L(\cdot \left| n_j^i, n_j^M, S_j^*\right.) \\ &= \log \prod_{j=1}^{m} P(A_j) \\ &= \sum_{j=1}^{m}\left\{\sum_{i \notin S_j^*} \log\left(\frac{\lambda_{ij}^{n_j^i}}{n_j^i!}\right) - \sum_{i=1}^{k} \lambda_{ij} + \log\left(\sum_{\cap_{r \in S_j^*}\left\langle i_r \geq n_j^r\right\rangle, \sum_{r \in S_j^*} i_r = n_j^*} \left(\prod_{r \in S_j^*} \frac{\lambda_{rj}^{i_r}}{i_r!}\right)\right)\right\} \end{aligned} \tag{21}$$

The MLE can be obtained by maximizing the log-likelihood function as shown in formula (21). However, it can be seen that the likelihood function is very complicated with general masked data, and it is also difficult to obtain the MLE by a numerical algorithm. This paper will use the EM algorithm to solve the problem of maximum likelihood estimation.

When the observed data are non-masked, i.e. $n_j^M = 0$ or $S_j^* = \emptyset$ for $j = 1, 2, \ldots, m$. The additive model can be decomposed into k NHPP models. The likelihood function becomes simple and is given by:

$$L(\cdot \left| n_j^i \right.) = \prod_{i=1}^{k} e^{-m_i(t_m)} \prod_{j=1}^{m} \left\{ \frac{(\lambda_{ij})^{n_j^i}}{n_j^i!} \right\} \tag{22}$$

Then the log-likelihood function has the following form:

$$\log L(\cdot \left| n_j^i \right.) = \sum_{i=1}^{k} \left[\sum_{j=1}^{m} n_j^i \log\left(\lambda_{ij}\right) - m_i(t_m) - \sum_{j=1}^{m} \log\left(n_j^i!\right) \right] \tag{23}$$

The likelihood function has the same forms as given in existed researches, and the computations of MLE are not complicate, see some references for details [24], [36], [37].

If the failure data are traditional masked described in Zhao and Xie [29], i.e. $S_j^* = \{1, 2, \ldots, k\}$. The likelihood function is reduced to

$$L(\cdot \left| n_j^i, n_j^M \right.) = e^{-m(t_m)} \cdot \prod_{j=1}^{m} \left\{ \frac{[m(t_j)-m(t_{j-1})]^{n_j}}{n_j!} \cdot \sum_{\cap\left\langle r_i \geq n_j^i \right\rangle, \sum_{i=1}^{k} r_i = n_j} \left(n_j! \cdot \prod_{i=1}^{k} \frac{p_{ij}^{r_i}}{r_i!} \right) \right\} \tag{24}$$

B. EM ALGORITHM MAXIMIZING LIKELIHOOD FUNCTION WITH GENERAL MASKED DATA

The fact that the estimation from non-masked data is reduced to simple cases stimulates the application of the EM algorithm. The EM algorithm has become increasingly popular today and has been used in various areas. It can be expected to make the maximization of likelihood functions very easy in some cases.

More generally, if the data consist of two parts: the observation x_{obs} and the missing data x_{miss}, we can state the EM algorithm in two steps to maximize the log-likelihood:

Expectation step: For current estimate $\theta^{(l)}$ of parameter θ, calculate the conditional expectation of the full log-likelihood:

$$Q\left(\theta^{(l)}, \theta\right) = E_{\theta^{(l)}} \left\{ \log\left(\theta | x_{obs}, x_{miss}\right) | x_{obs} \right\} \tag{25}$$

Maximization step: Find a new estimate $\theta^{(l+1)}$ as the value of θ by maximizing function $Q\left(\theta^{(l)}, \theta\right)$.

Under fairly general conditions, the sequence $\{\theta^{(l)}, l = 1, 2, \cdots\}$ will converge to the MLE obtained by maximizing the overall likelihood function. More discussions on this algorithm are referred to reference [38].

Suppose that the mean value function $m_i(t)$ for each release i contains unknown parameter θ_i. In current problem, the missing data are occurred when $n_j^M > 0$ since the observation of random variable N_j^i is not complete. By using the formula (23) and (25), the function Q has the form of

$$Q\left(\theta^{(l)}, \theta\right) = \sum_{i=1}^{k} \left\{ \sum_{j=1}^{m} \begin{matrix} E\left(N_j^i | n_j^i, n_j^M, \theta^{(l)}\right) \cdot \\ \log\left[m_i\left(t_j, \theta_i\right) - m_i\left(t_{j-1}, \theta_i\right)\right] \end{matrix} - m_i\left(t_m, \theta_i\right) \right\} \tag{26}$$

Note that $E\left(N_j^i | n_j^i, n_j^M, \theta^{(l)}\right)$ is independent on the dummy variable θ and can be regarded as constant in maximizing function $Q\left(\theta^{(l)}, \theta\right)$, so that the maximizing step can be completed by maximizing the following functions separately:

$$Q\left(\theta^{(l)}, \theta\right) = \sum_{j=1}^{m} \begin{matrix} E\left(N_j^i | n_j^i, n_j^M, \theta^{(l)}\right) \\ \log\left[m_i\left(t_j, \theta_i\right) - m_i\left(t_{j-1}, \theta_i\right)\right] \end{matrix} - m_i\left(t_m, \theta_i\right) \tag{27}$$

To realize the EM algorithm, one needs to find out what is the expected number of failures for each release i at each observation time interval when the system has n_j^M masked failures. Next, we focus on the case of $n_j^M \neq 0$, that is $S_j^* \neq \emptyset$. Since, when $S_j^* = \emptyset$, there are no masked. Let random vector $N_j^* = (N_j^{r_1}, \ldots, N_j^{r_{Lj}}), r_l \in S_j^*, j = 1, 2, \ldots, m$, where Lj is the number of elements in set S_j^* and $l = 1, 2, \ldots, Lj$. It is well known that random vector N_j^* obeys multinomial distribution, that is, $N_j^* \sim M(n_j^*, p_{r_1}, p_{r_2}, \cdots, p_{r_{Lj}})$. Where n_j^* is shown in formula (15), and

$$\begin{aligned} p_{rj} &= \frac{m_r(t_j) - m_r(t_{j-1})}{m(t_j) - m(t_{j-1})} \\ &= \frac{m_r(t_j) - m_r(t_{j-1})}{\sum_{r \in S_j^*} \left[m_r(t_j) - m_r(t_{j-1})\right]}, \\ & r \in S_j^* \neq \emptyset, \quad j = 1, 2, \cdots, m \end{aligned} \tag{28}$$

It is easy to know $\sum_{r \in S_j^*} p_{rj} = \sum_{l=1}^{Lj} p_{r_l} = 1$. Furthermore, we can obtain the following conditional probability.

$$P\left\{ \left(\bigcap_{r \in S_j^*} \left\langle N_j^r \geq n_j^r \right\rangle \right) \left| \sum_{r \in S_j^*}^{N_j^r} = n_j^* \right. \right\} = \sum_{\cap_{r \in S_j^*}\left\langle \alpha_r \geq n_j^r \right\rangle, \sum_{r \in S_j^*} \alpha_r = n_j^*} \left(n_j^*! \cdot \prod_{r \in S_j^*} \frac{p_{rj}^{\alpha_r}}{\alpha_r!} \right) \tag{29}$$

The problem is now focused on how to calculate, for each release i, the conditional expectation of the number of failures. For a specific release i, the conditional expectation of N_j^i, denoted by $\hat{N}_j^i = E\left(N_j^i | n_j^i, n_j^M, \theta^{(l)}\right)$, can be shown to be that:

$$\hat{N}_j^i = \begin{cases} n_j^i, & i \notin S_j^* \neq \emptyset \text{ or } S_j^* = \emptyset \\ \dfrac{\displaystyle\sum_{\substack{\cap_{r\in S_j^*}\langle \alpha_r \geq n_j^r \rangle, \\ \sum_{r\in S_j^*} \alpha_r = n_j^*}} \left(\alpha_r \cdot \prod_{r\in S_j^*} \frac{p_{rj}^{\alpha_r}}{\alpha_r!} \right)}{\displaystyle\sum_{\substack{\cap_{r\in S_j^*}\langle \alpha_r \geq n_j^r \rangle, \\ \sum_{r\in S_j^*} \alpha_r = n_j^*}} \left(\prod_{r\in S_j^*} \frac{p_{rj}^{\alpha_r}}{\alpha_r!} \right)}, & i \in S_j^* \neq \emptyset \end{cases}$$

$$i = 1, 2, \cdots, k; j = 1, 2, \cdots, m \tag{30}$$

Now, we summarize the EM procedure for the estimates of parameters $\theta = (\theta_1, \theta_2, \cdots \theta_k)$ with general masked data as follows:

Step 1: Give initial values of parameters $(\theta_1, \cdots, \theta_k)^{(0)}$.

Step 2: Calculate conditional expectations $\hat{N}_j^i$ by formula (30).

Step 3: Obtain the new estimates $(\theta_1, \cdots, \theta_k)^{(1)}$ by maximizing the log-likelihood function with respect to θ_i using formula (27).

Step 4: Replace $(\theta_1, \cdots, \theta_k)^{(0)}$ by $(\theta_1, \cdots, \theta_k)^{(1)}$ and go to step 2.

Step 5: Repeat step 2-step 4 until stable values are obtained.

IV. NUMERICAL EXAMPLES

A. DATA DESCRIPTION

The failure data required in this paper comes from the user bug tracking system, which is a bug reporting system. Mozilla Bugzilla (http://www.bugzilla.org/) is the most popular bug tracking system. It is a web application for software bug tracking management, developed by the Mozilla Foundation program.

To validate our model, the first data set (DS-1) we employed was from a real open source project, is called Apache Tomcat. Tomcat is a core project in the Jakarta project of the Apache Software Foundation. It was jointly developed by Apache, Sun, and other companies and individuals. The Tomcat server is a free open source web application server. The failure data come from bug tracking system of Tomcat (https://bz.apache.org/bugzilla/). The failure data set has 162 corresponding data entries from October 2006 to March 2020, as shown in Table 4. The data set contains the failure data of three software release versions of Apache Tomcat, namely release version 6.x, 7.x and 8.x. The release times of version 6.0.0, 7.0.0 and 8.0.0 are October 2006, June 2010, and August 2013 respectively (See the official website for details: https://tomcat.apache.org/oldnews.html). In Table 4, R6, R7, R8 represents the number of failures at each observation time interval for release version 6, version 7 and version 8 respectively. M stands for the number of failures for Masked or Unknown. S_j^* is the failure cause set shown in formula (11).

The second data set (DS-2) was obtained from a real open source project, is called Apache POI (Poor Obfuscation Implementation). The Apache POI project is the master project for developing pure Java ports of file formats based on Microsoft's OLE 2 Compound Document Format. Apache POI is also the master project for developing pure Java ports of file formats based on Office Open XML. The failure data come from bug tracking system of Tomcat (https://bz.apache.org/bugzilla/). The failure data set has 211 corresponding data entries from March 2002 to September 2019, as shown in Table 4. The data set contains the failure data of three software release versions of Apache POI, namely release version 1.x, 2.x and 3.x. The release times of version 1.1.0, 2.0-pre2 and 3.0-final are January 2002, July 2003, and May 2007 respectively (See the official website: https://poi.apache.org/devel/history/index.html). In Table 5, R1, R2, R3 represents the number of failures at each observation time interval for release version 1.x, 2.x and 3.x respectively. M stands for the number of failures for Masked or Unknown. S_j^* is the failure cause set shown in formula (11).

B. MODEL PERFORMANCE EVALUATION CRITERIA

To validate the proposed model, it is necessary to apply some measurement on how well the model can fit the observed data. The Mean Squared Error (MSE), Akaike information Criterion (AIC) and Bayesian Information Criterion(BIC) are used to compare the goodness of the model fit. The MSE can be calculated as:

$$\begin{aligned} MSE &= \frac{1}{m}\sum_{j=1}^{m}\left(m(t_j) - m_j\right)^2 \\ &= \frac{1}{m}\sum_{j=1}^{m}\left(\sum_{i=1}^{k} m_i(t_j) - m_j\right)^2 \end{aligned} \tag{31}$$

where, $m(t_j)$ and m_j is the estimated and observed cumulative number of failures for system until t_j respectively. It is obvious to see that the smaller of MSE, the better the model gives the fit to the observed data.

AIC is a standard for measuring the goodness of fit of a statistical model. It can weigh the complexity of the estimated model and the goodness of the fit data of the model. In general, AIC can be expressed as:

$$\begin{aligned} AIC &= 2K - \log(L) \\ &= 2K - \frac{1}{k}\sum_{i=1}^{k}\log(L_i) \end{aligned} \tag{32}$$

where K is the number of parameters in model and L is the maximum value of the likelihood function. Increasing the number of parameters improves the goodness of fitting. AIC encourages the goodness of data fitting but tries to avoid

TABLE 4. The number of failures with general masked data for apache tomcat (DS-1).

No.	Date	R6	R7	R8	M	S_j^*	No.	Date	R6	R7	R8	M	S_j^*	No.	Date	R6	R7	R8	M	S_j^*
1	200610	1	0	0	0	∅	55	201104	2	5	0	0	S1	109	201510	1	2	2	0	S2
2	200611	3	0	0	0	∅	56	201105	6	6	0	3	S1	110	201511	1	0	6	1	S2
3	200612	5	0	0	0	∅	57	201106	7	11	0	0	S1	111	201512	0	1	6	0	S2
4	200701	3	0	0	0	∅	58	201107	5	9	0	3	S1	112	201601	0	0	3	1	S3
5	200702	3	0	0	0	∅	59	201108	0	13	0	4	S1	113	201602	0	2	5	0	S3
6	200703	4	0	0	0	∅	60	201109	2	8	0	1	S1	114	201603	0	4	10	1	S3
7	200704	4	0	0	0	∅	61	201110	1	7	0	3	S1	115	201604	0	2	4	0	S3
8	200705	9	0	0	0	∅	62	201111	2	5	0	1	S1	116	201605	0	1	5	0	S3
9	200706	3	0	0	0	∅	63	201112	2	3	0	1	S1	117	201606	0	0	2	1	S3
10	200707	4	0	0	0	∅	64	201201	2	12	0	3	S1	118	201607	0	2	11	0	S3
11	200708	12	0	0	0	∅	65	201202	3	11	0	4	S1	119	201608	0	1	8	0	S3
12	200709	13	0	0	0	∅	66	201203	6	13	0	1	S1	120	201609	0	1	8	0	S3
13	200710	20	0	0	0	∅	67	201204	4	7	0	6	S1	121	201610	0	0	7	0	S3
14	200711	8	0	0	0	∅	68	201205	0	14	0	2	S1	122	201611	0	1	6	1	S3
15	200712	2	0	0	0	∅	69	201206	1	16	0	6	S1	123	201612	0	0	8	0	S3
16	200801	6	0	0	0	∅	70	201207	2	8	0	6	S1	124	201701	0	1	5	0	S3
17	200802	9	0	0	0	∅	71	201208	2	10	0	2	S1	125	201702	0	0	9	0	S3
18	200803	7	0	0	0	∅	72	201209	1	4	0	1	S1	126	201703	0	1	11	0	S3
19	200804	5	0	0	0	∅	73	201210	3	7	0	1	S1	127	201704	0	0	3	0	S3
20	200805	8	0	0	0	∅	74	201211	3	9	0	1	S1	128	201705	0	0	6	1	S3
21	200806	10	0	0	0	∅	75	201212	1	6	0	2	S1	129	201706	0	0	4	0	S3
22	200807	14	0	0	0	∅	76	201301	0	18	0	0	S1	130	201707	0	1	4	0	S3
23	200808	5	0	0	0	∅	77	201302	2	7	0	1	S1	131	201708	0	1	4	0	S3
24	200809	9	0	0	0	∅	78	201303	1	8	0	1	S1	132	201709	0	0	6	0	S3
25	200810	15	0	0	0	∅	79	201304	0	6	0	0	S1	133	201710	0	1	4	0	S3
26	200811	4	0	0	0	∅	80	201305	1	10	0	1	S1	134	201711	0	0	13	0	S3
27	200812	3	0	0	0	∅	81	201306	1	11	0	1	S1	135	201712	0	0	3	0	S3
28	200901	7	0	0	0	∅	82	201307	4	10	0	2	S1	136	201801	0	2	7	0	S3
29	200902	1	0	0	0	∅	83	201308	2	5	2	3	S2	137	201802	0	0	2	0	S3
30	200903	9	0	0	0	∅	84	201309	0	5	2	0	S2	138	201803	0	0	0	0	S3
31	200904	6	0	0	0	∅	85	201310	4	4	3	1	S2	139	201804	0	1	3	0	S3
32	200905	2	0	0	0	∅	86	201311	2	6	3	4	S2	140	201805	0	1	2	0	S3
33	200906	13	0	0	0	∅	87	201312	0	6	3	0	S2	141	201806	0	0	0	0	S3
34	200907	5	0	0	0	∅	88	201401	0	8	4	8	S2	142	201807	0	0	2	0	S3
35	200908	3	0	0	0	∅	89	201402	0	2	6	0	S2	143	201808	0	0	2	0	S3
36	200909	9	0	0	0	∅	90	201403	1	10	3	3	S2	144	201809	0	1	4	0	S3
37	200910	6	0	0	0	∅	91	201404	0	12	5	2	S2	145	201810	0	1	1	0	S3
38	200911	7	0	0	0	∅	92	201405	2	8	3	1	S2	146	201811	0	0	1	0	S3
39	200912	8	0	0	0	∅	93	201406	6	4	7	0	S2	147	201812	0	1	5	0	S3
40	201001	14	0	0	0	∅	94	201407	1	8	4	0	S2	148	201901	0	0	1	0	S3
41	201002	21	0	0	0	∅	95	201408	2	3	3	0	S2	149	201902	0	1	4	0	S3
42	201003	18	0	0	0	∅	96	201409	0	4	8	0	S2	150	201903	0	1	2	1	S3
43	201004	14	0	0	0	∅	97	201410	2	3	11	1	S2	151	201904	0	0	1	0	S3
44	201005	3	0	0	0	∅	98	201411	0	9	8	1	S2	152	201905	0	0	0	0	S3
45	201006	9	6	0	4	S1	99	201412	1	7	1	2	S2	153	201906	0	1	1	0	S3
46	201007	8	4	0	2	S1	100	201501	0	2	11	0	S2	154	201907	0	0	2	0	S3
47	201008	2	7	0	2	S1	101	201502	1	4	9	3	S2	155	201908	0	2	1	0	S3
48	201009	13	4	0	3	S1	102	201503	3	9	9	1	S2	156	201909	0	2	1	0	S3
49	201010	7	4	0	6	S1	103	201504	1	5	6	1	S2	157	201910	0	2	1	0	S3
50	201011	6	11	0	1	S1	104	201505	2	4	5	0	S2	158	201911	0	1	3	0	S3
51	201012	6	5	0	2	S1	105	201506	0	1	2	2	S2	159	201912	0	0	5	0	S3
52	201101	10	9	0	5	S1	106	201507	0	4	5	1	S2	160	202001	0	0	1	0	S3
53	201102	5	7	0	1	S1	107	201508	0	1	4	0	S2	161	202002	0	0	2	0	S3
54	201103	5	7	0	1	S1	108	201509	0	1	6	2	S2	162	202003	0	2	3	0	S3

Note that: $S1 = \{1, 2\}$, $S2 = \{1, 2, 3\}$, $S3 = \{2, 3\}$

TABLE 5. The number of failures with general masked data for apache POI (DS-2).

No.	Date	R1	R2	R3	M	S_j^*	No.	Date	R1	R2	R3	M	S_j^*	No.	Date	R1	R2	R3	M	S_j^*
1	200203	14	0	0	0	Ø	72	200802	0	0	6	4	S2	143	201401	0	0	6	0	Ø
2	200204	8	0	0	0	Ø	73	200803	0	0	9	8	S2	144	201402	0	0	9	0	Ø
3	200205	4	0	0	0	Ø	74	200804	0	0	7	3	S2	145	201403	0	0	5	0	Ø
4	200206	4	0	0	0	Ø	75	200805	0	0	5	6	S2	146	201404	0	0	6	0	Ø
5	200207	6	0	0	0	Ø	76	200806	0	0	3	6	S2	147	201405	0	0	13	0	Ø
6	200208	3	0	0	0	Ø	77	200807	0	0	4	20	S2	148	201406	0	0	3	0	Ø
7	200209	3	0	0	0	Ø	78	200808	0	1	0	25	S2	149	201407	0	0	4	0	Ø
8	200210	2	0	0	0	Ø	79	200809	0	0	2	0	Ø	150	201408	0	0	5	0	Ø
9	200211	1	0	0	0	Ø	80	200810	0	0	0	0	Ø	151	201409	0	0	2	0	Ø
10	200212	3	0	0	0	Ø	81	200811	0	0	12	0	Ø	152	201410	0	0	4	0	Ø
11	200301	1	0	0	0	Ø	82	200812	0	0	3	0	Ø	153	201411	0	0	2	0	Ø
12	200302	0	0	0	0	Ø	83	200901	0	0	6	0	Ø	154	201412	0	0	3	0	Ø
13	200303	0	0	0	0	Ø	84	200902	0	0	7	0	Ø	155	201501	0	0	5	0	Ø
14	200304	2	0	0	0	Ø	85	200903	0	0	1	0	Ø	156	201502	0	0	8	0	Ø
15	200305	2	0	0	0	Ø	86	200904	0	0	3	0	Ø	157	201503	0	0	6	0	Ø
16	200306	0	0	0	0	Ø	87	200905	0	0	5	0	Ø	158	201504	0	0	4	0	Ø
17	200307	1	5	0	1	S1	88	200906	0	0	3	0	Ø	159	201505	0	0	10	0	Ø
18	200308	2	2	0	0	S1	89	200907	0	0	0	0	Ø	160	201506	0	0	4	0	Ø
19	200309	1	6	0	0	S1	90	200908	0	0	3	0	Ø	161	201507	0	0	7	0	Ø
20	200310	1	8	0	1	S1	91	200909	0	0	3	0	Ø	162	201508	0	0	3	0	Ø
21	200311	2	2	0	2	S1	92	200910	0	0	17	0	Ø	163	201509	0	0	3	0	Ø
22	200312	1	2	0	1	S1	93	200911	0	0	11	0	Ø	164	201510	0	0	2	0	Ø
23	200401	0	1	0	1	S1	94	200912	0	0	7	0	Ø	165	201511	0	0	7	0	Ø
24	200402	1	4	0	6	S1	95	201001	0	0	9	0	Ø	166	201512	0	0	4	0	Ø
25	200403	0	11	0	4	S1	96	201002	0	0	5	0	Ø	167	201601	0	0	4	0	Ø
26	200404	0	4	0	4	S1	97	201003	0	0	13	0	Ø	168	201602	0	0	5	0	Ø
27	200405	0	1	0	3	S1	98	201004	0	0	4	0	Ø	169	201603	0	0	7	0	Ø
28	200406	1	6	0	2	S1	99	201005	0	0	5	0	Ø	170	201604	0	0	2	0	Ø
29	200407	0	8	0	3	Ø	100	201006	0	0	7	0	Ø	171	201605	0	0	10	0	Ø
30	200408	0	8	0	2	Ø	101	201007	0	0	6	0	Ø	172	201606	0	0	6	0	Ø
31	200409	0	4	0	0	Ø	102	201008	0	0	3	0	Ø	173	201607	0	0	6	0	Ø
32	200410	0	6	0	0	Ø	103	201009	0	0	4	0	Ø	174	201608	0	0	1	0	Ø
33	200411	0	2	0	0	Ø	104	201010	0	0	2	0	Ø	175	201609	0	0	3	0	Ø
34	200412	0	2	0	1	Ø	105	201011	0	0	9	0	Ø	176	201610	0	0	13	0	Ø
35	200501	0	3	0	1	Ø	106	201012	0	0	7	0	Ø	177	201611	0	0	8	0	Ø
36	200502	0	1	0	1	Ø	107	201101	0	0	2	0	Ø	178	201612	0	0	5	0	Ø
37	200503	0	5	0	0	Ø	108	201102	0	0	10	0	Ø	179	201701	0	0	1	0	Ø
38	200504	0	0	0	0	Ø	109	201103	0	0	4	0	Ø	180	201702	0	0	2	0	Ø
39	200505	0	3	0	3	Ø	110	201104	0	0	2	0	Ø	181	201703	0	0	3	0	Ø
40	200506	0	5	0	3	Ø	111	201105	0	0	3	0	Ø	182	201704	0	0	6	0	Ø
41	200507	0	2	0	1	Ø	112	201106	0	0	3	0	Ø	183	201705	0	0	4	0	Ø
42	200508	0	2	0	1	Ø	113	201107	0	0	4	0	Ø	184	201706	0	0	2	0	Ø
43	200509	0	0	0	0	Ø	114	201108	0	0	4	0	Ø	185	201707	0	0	5	0	Ø
44	200510	0	2	0	0	Ø	115	201109	0	0	3	0	Ø	186	201708	0	0	3	0	Ø
45	200511	0	1	0	2	Ø	116	201110	0	0	1	0	Ø	187	201709	0	0	7	0	Ø
46	200512	0	1	0	0	Ø	117	201111	0	0	0	0	Ø	188	201710	0	0	7	0	Ø
47	200601	0	2	0	2	Ø	118	201112	0	0	1	0	Ø	189	201711	0	0	6	0	Ø
48	200602	0	3	0	0	Ø	119	201201	0	0	2	0	Ø	190	201712	0	0	8	0	Ø
49	200603	0	1	0	5	Ø	120	201202	0	0	4	0	Ø	191	201801	0	0	2	0	Ø
50	200604	0	2	0	6	Ø	121	201203	0	0	2	0	Ø	192	201802	0	0	3	0	Ø
51	200605	0	0	0	1	Ø	122	201204	0	0	8	0	Ø	193	201803	0	0	6	0	Ø
52	200606	0	0	0	1	Ø	123	201205	0	0	12	0	Ø	194	201804	0	0	4	0	Ø
53	200607	0	2	0	6	Ø	124	201206	0	0	10	0	Ø	195	201805	0	0	0	0	Ø

TABLE 5. ***(Continued.)*** **The number of failures with general masked data for apache POI (DS-2).**

54	200608	0	1	0	3	∅	125	201207	0	0	6	0	∅	196	201806	0	0	4	0	∅
55	200609	0	1	0	2	∅	126	201208	0	0	6	0	∅	197	201807	0	0	1	0	∅
56	200610	0	2	0	2	∅	127	201209	0	0	6	0	∅	198	201808	0	0	2	0	∅
57	200611	0	1	0	4	∅	128	201210	0	0	9	0	∅	199	201809	0	0	0	0	∅
58	200612	0	1	0	2	∅	129	201211	0	0	5	0	∅	200	201810	0	0	0	0	∅
59	200701	0	0	0	1	∅	130	201212	0	0	0	0	∅	201	201811	0	0	0	0	∅
60	200702	0	1	0	1	∅	131	201301	0	0	6	0	∅	202	201812	0	0	0	0	∅
61	200703	0	0	0	0	∅	132	201302	0	0	7	0	∅	203	201901	0	0	0	0	∅
62	200704	0	0	0	0	∅	133	201303	0	0	5	0	∅	204	201902	0	0	1	0	∅
63	200705	0	1	0	1	S2	134	201304	0	0	4	0	∅	205	201903	0	0	0	0	∅
64	200706	0	0	3	3	S2	135	201305	0	0	6	0	∅	206	201904	0	0	0	0	∅
65	200707	0	0	2	1	S2	136	201306	0	0	2	0	∅	207	201905	0	0	0	0	∅
66	200708	0	0	10	2	S2	137	201307	0	0	5	0	∅	208	201906	0	0	0	0	∅
67	200709	0	0	0	3	S2	138	201308	0	0	3	0	∅	209	201907	0	0	0	0	∅
68	200710	0	0	2	0	S2	139	201309	0	0	5	0	∅	210	201908	0	0	0	0	∅
69	200711	0	0	6	2	S2	140	201310	0	0	6	0	∅	211	201909	0	0	1	0	∅
70	200712	0	0	3	4	S2	141	201311	0	0	5	0	∅							
71	200801	0	0	7	2	S2	142	201312	0	0	8	0	∅							

Note that: $S1 = \{1, 2\}, S2 = \{2, 3\}$

over fitting. So the priority model should be the one with the smallest AIC value.

BIC is also a standard for measuring the goodness of fit of a statistical model. When the sample size is large, BIC penalizes the model parameters more than AIC, which causes BIC to prefer simple models with fewer parameters. BIC can be expressed as:

$$\begin{aligned} BIC &= K \log(m) - \log(L) \\ &= K \log(m) - \frac{1}{k}\sum_{i=1}^{k} \log(L_i) \end{aligned} \quad (33)$$

where K is the number of parameters in model, m is the sample size, and L is the maximum value of the likelihood function.

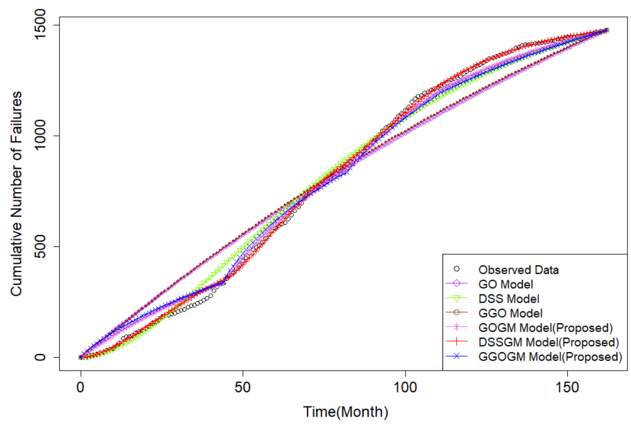

FIGURE 2. **Fitted versus Observed for all Selected Models (DS-1).**

C. PERFORMANCE ANALYSIS

In this section, the datasets collected from bug tracking system of Apache Tomcat and POI are used to conduct a comparative analysis of model performance. Those two datasets are grouped general masked data with equal time interval of one month.

1) THE FIRST DATA Set (DS-1)

Using the EM algorithm described in the last section, the estimated parameters and comparison results of all selected models for dataset 1 are shown in Table 6. Figure 2 displays the observed cumulative number of failures and the fitted mean value functions in all selected models. From Table 6, we can see that the MSE, AIC and BIC of the GOGM model(proposed) are less than the traditional GO model. Moreover, the MSE, AIC and BIC of the DSSGM model (proposed) are also less than the traditional DSS model. Finally, the GGOGM model(proposed) are also less than the traditional GGO model. On the other hand, we can see that the MSE, AIC and BIC of all proposed models (GOGM model, DSSGM model and GGOGM model), are still small than any traditional models (GO model, DSS model and GGO model). On the whole, it is reasonable to conclude that the proposed models have the better goodness-of-fit than traditional models. Furthermore, we insist that the DSSGM model (proposed) has the best goodness-of-fit of all selected model.

2) THE SECOND DATA Set (DS-2)

Using the EM algorithm described in the last section, the estimated parameters and comparison results of all selected models for dataset 2 are also shown in Table 7. Figure 3 displays the observed cumulative number of failures and the fitted mean value functions in all selected models. From Table 7, we can see that the MSE, AIC

TABLE 6. Parameter estimation and comparison results of all selected models for DS-1.

Model	MLE			MSE	AIC	BIC
GO Model	a = 3327.3877	b = 0.0036	-	9384.5285	1199.7300	1205.9052
DSS Model	a = 1716.1629	b = 0.0214	-	1773.4567	999.9490	1006.1241
GGO Model	a = 2875.9338	b = 0.0038	c = 1.0301	8891.1937	1185.7152	1194.9780
GOGM Model (Proposed Model)	a_1 = 696.1411 a_2 = 565.4845 a_3 = 601.9866	b_1 = 0.0151 b_2 = 0.0256 b_3 = 0.0120	- - -	911.6854	901.4500	919.9756
DSSGM Model (Proposed Model)	a_1 = 596.3407 a_2 = 540.0104 a_3 = 402.4157	b_1 = 0.0447 b_2 = 0.0591 b_3 = 0.0498	- - -	185.6288	854.2109	872.7365
GGOGM Model (Proposed Model)	a_1 = 792.8599 a_2 = 609.2556 a_3 = 1195.2901	b_1 = 0.0205 b_2 = 0.0416 b_3 = 0.0101	c_1 =0.8710 c_2 =0.8281 c_3 =0.8246	1570.0215	949.4400	977.2284

TABLE 7. Parameter estimation and comparison results of all selected models for DS-2.

Model	MLE			MSE	AIC	BIC
GO Model	a = 3060.5731	b = 0.0019	-	1353.1231	1296.3597	1303.0634
DSS Model	a = 1231.0410	b = 0.0153	-	851.1757	1395.7537	1402.4574
GGO Model	a = 2201.8160	b = 0.0022	c = 1.0518	1256.1634	1294.6146	1304.6702
GOGM Model (Proposed Model)	a_1 = 66.9863 a_2 = 237.0553 a_3 = 1164.4941	b_1 = 0.1233 b_2 = 0.0350 b_3 = 0.0069	- - -	280.1563	447.0671	467.1783
DSSGM Model (Proposed Model)	a_1 = 62.8289 a_2 = 207.7537 a_3 = 822.7653	b_1 = 0.2979 b_2 = 0.1003 b_3 = 0.0278	- - -	1198.7559	473.2420	493.3531
GGOGM Model (Proposed Model)	a_1 = 230.7889 a_2 = 219.5563 a_3 = 2039.6187	b_1 = 0.0627 b_2 = 0.0271 b_3 = 0.0075	c_1 =0.5183 c_2 =1.1158 c_3 =0.8229	386.4840	458.1382	488.3049

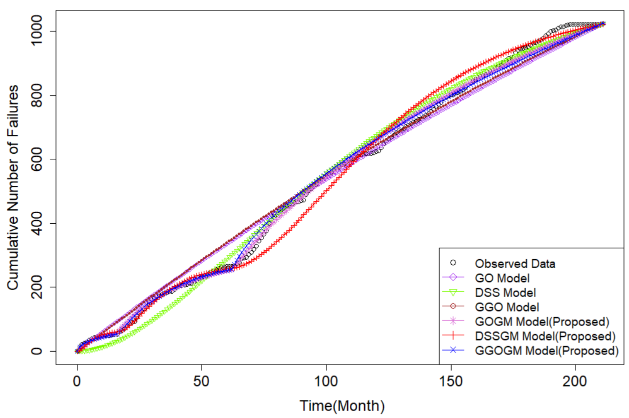

FIGURE 3. Fitted versus Observed for all Selected Models (DS-2).

and BIC of the GOGM model(proposed) are less than the traditional GO model. Moreover, the AIC and BIC of the DSSGM model (proposed) are also less than the traditional DSS model. And the DSSGM model doesn't provide the smaller MSE compared to the traditional DSS model, but the differences are not big. Finally, the GGOGM model(proposed) are also less than the traditional GGO model. On the other hand, we can see that the MSE, AIC and BIC of all proposed models (GOGM model, DSSGM model and GGOGM model), are still small than any traditional models (GO model, DSS model and GGO model) expect for the MSE of DSSGM model. On the whole, it is reasonable to conclude that the proposed models have the better goodness-of-fit than traditional models. Furthermore, we insist that the GOGM model (proposed) has the best goodness-of-fit of all selected model.

V. CONCLUSION

Masked data are the system failure data when the exact cause of the failures might be unknown. That is, the cause of the system failures may be any one of the components. However, due to the influence of the test strategy in real project, the cause of the system failures may be a subset of the system objects, not any one of the objects. Additionally, multi-release is critical for modern open source software product in order to satisfy more customer requirements. If there exist the masked data, the objective function in MLE and LSE becomes a complex multivariable function with a very high dimension. Therefore, common techniques for maximizing or minimizing a multivariate nonlinear function are not easily used because there may exist so many unknown parameters.

In this paper, we first discuss the mathematical description of general masked data based on the traditional masked data and review the additive NHPP-based reliability model. Furthermore, a novel multi-release OSS reliability model based on general masked data is proposed and EM algorithm is used to solve the extremely complicated problem of the log-likelihood function. In general, the proposed mode can be extended to other NHPP-based model for estimating multi-release OSS reliability. Using open source software Apache Tomcat and POI grouped masked data to conduct a comparative analysis of model performance, the results show that the proposed models are useful and powerful. Finally, the reliability modeling of hardware/software system considering masked failure data will be studied in the future works.

REFERENCES

[1] C.-Y. Huang, M. R. Lyu, and S.-Y. Kuo, "A unified scheme of some nonhomogenous Poisson process models for software reliability estimation," *IEEE Trans. Softw. Eng.*, vol. 29, no. 3, pp. 261–269, Mar. 2003, doi: 10.1109/TSE.2003.1183936.

[2] X. Xiao and T. Dohi, "Wavelet shrinkage estimation for non-homogeneous Poisson process based software reliability models," *IEEE Trans. Rel.*, vol. 62, no. 1, pp. 211–225, Mar. 2013, doi: 10.1109/tr.2013.2240897.

[3] J. Yang, M. Zhao, and W. Hu, "Web software reliability modeling with random impulsive shocks," *J. Syst. Eng. Electron.*, vol. 25, no. 2, pp. 349–356, Apr. 2014.

[4] S. E. Rigdon, "Non-homogeneous Poisson process models for software reliability," in *Analytic Methods in Systems and Software Testing*. Hoboken, NJ, USA: Wiley, 2018, pp. 195–211, doi: 10.1002/9781119357056.ch7.

[5] Q. Li and H. Pham, "A generalized software reliability growth model with consideration of the uncertainty of operating environments," *IEEE Access*, vol. 7, pp. 84253–84267, 2019, doi: 10.1109/access.2019.2924084.

[6] Y. Tamura and S. Yamada, "Optimisation analysis for reliability assessment based on stochastic differential equation modelling for open source software," *Int. J. Syst. Sci.*, vol. 40, no. 4, pp. 429–438, Apr. 2009, doi: 10.1080/00207720802556245.

[7] Y. Tamura and S. Yamada, "Reliability assessment based on hazard rate model for an embedded OSS porting-phase," *Softw. Test., Verification Rel.*, vol. 23, no. 1, pp. 77–88, Jan. 2013, doi: 10.1002/stvr.455.

[8] S.-P. Luan and C.-Y. Huang, "An improved Pareto distribution for modelling the fault data of open source software," *Softw. Test., Verification Rel.*, vol. 24, no. 6, pp. 416–437, Sep. 2014, doi: 10.1002/stvr.1504.

[9] C.-Y. Huang, C.-S. Kuo, and S.-P. Luan, "Evaluation and application of bounded generalized Pareto analysis to fault distributions in open source software," *IEEE Trans. Rel.*, vol. 63, no. 1, pp. 309–319, Mar. 2014, doi: 10.1109/tr.2013.2285056.

[10] N. Ullah, "A method for predicting open source software residual defects," *Softw. Qual. J.*, vol. 23, no. 1, pp. 55–76, Mar. 2015, doi: 10.1007/s11219-014-9229-3.

[11] N. Ullah, M. Morisio, and A. Vetro, "Selecting the best reliability model to predict residual defects in open source software," *Computer*, vol. 48, no. 6, pp. 50–58, Jun. 2015, doi: 10.1109/mc.2013.446.

[12] J. Wang and X. Mi, "Open source software reliability model with the decreasing trend of fault detection rate," *Comput. J.*, vol. 62, no. 9, pp. 1301–1312, Sep. 2019, doi: 10.1093/comjnl/bxy111.

[13] O. Saliu and G. Ruhe, "Software release planning for evolving systems," *Innov. Syst. Softw. Eng.*, vol. 1, no. 2, pp. 189–204, Sep. 2005, doi: 10.1007/s11334-005-0012-2.

[14] X. Li, Y. F. Li, M. Xie, and S. H. Ng, "Reliability analysis and optimal version-updating for open source software," *Inf. Softw. Technol.*, vol. 53, no. 9, pp. 929–936, Sep. 2011, doi: 10.1016/j.infsof.2011.04.005.

[15] Q. P. Hu, R. Peng, M. Xie, S. H. Ng, and G. Levitin, "Software reliability modelling and optimization for multi-release software development processes," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2011, pp. 1534–1538, doi: 10.1109/IEEM.2011.6118174.

[16] P. K. Kapur, H. Pham, A. G. Aggarwal, and G. Kaur, "Two dimensional multi-release software reliability modeling and optimal release planning," *IEEE Trans. Rel.*, vol. 61, no. 3, pp. 758–768, Sep. 2012.

[17] B. Pachauri, J. Dhar, and A. Kumar, "Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth," *Appl. Math. Model.*, vol. 39, nos. 5–6, pp. 1463–1469, Mar. 2015, doi: 10.1016/j.apm.2014.08.006.

[18] J. Yang, Y. Liu, M. Xie, and M. Zhao, "Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes," *J. Syst. Softw.*, vol. 115, pp. 102–110, May 2016, doi: 10.1016/j.jss.2016.01.025.

[19] M. Ahmadi, I. Mahdavi, and A. H. S. Garmabaki, "Multi up-gradation reliability model for open source software," in *Current Trends in Reliability, Availability, Maintainability and Safety*, U. Kumar, A. Ahmadi, A. K. Verma, and P. Varde, Eds. Cham, Switzerland: Springer, 2016, pp. 691–702.

[20] V. B. Singh, M. Sharma, and H. Pham, "Entropy based software reliability analysis of multi-version open source software," *IEEE Trans. Softw. Eng.*, vol. 44, no. 12, pp. 1207–1223, Dec. 2018, doi: 10.1109/tse.2017.2766070.

[21] M. Zhu and H. Pham, "A multi-release software reliability modeling for open source software incorporating dependent fault detection process," *Ann. Oper. Res.*, vol. 269, nos. 1–2, pp. 773–790, Oct. 2018, doi: 10.1007/s10479-017-2556-6.

[22] N. Yilmaz and A. Tarhan, "A two-dimensional method for evaluating maintainability and reliability of open source software," *J. Fac. Eng. Archit. Gazi Univ.*, vol. 34, no. 4, pp. 1807–1829, 2019, doi: 10.17341/gazimmfd.571563.

[23] M. Ohba, "Software reliability analysis models," *IBM J. Res. Develop.*, vol. 28, no. 4, pp. 428–443, Jul. 1984.

[24] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Trans. Rel.*, vol. R-28, no. 3, pp. 206–211, Aug. 1979.

[25] S. Yamada, S. Osaki, and H. Narihisa, "A software reliability growth model with two types of errors," *RAIRO Oper. Res.*, vol. 19, no. 1, pp. 87–104, 1985.

[26] M. Xie and T. N. Goh, "System reliability growth analysis using component failure data," *Int. J. Rel., Qual. Saf. Eng.*, vol. 1, no. 1, pp. 71–83, Mar. 1994.

[27] M. Xie and C. D. Lai, "Reliability analysis using an additive weibull model with bathtub-shaped failure rate function," *Rel. Eng. Syst. Saf.*, vol. 52, no. 1, pp. 87–93, Apr. 1996.

[28] F. K. Wang, "A new model with bathtub-shaped failure rate using an additive burr XII distribution," *Rel. Eng. Syst. Saf.*, vol. 70, no. 3, pp. 305–312, Dec. 2000.

[29] M. Zhao and M. Xie, "EM algorithms for estimating software reliability based on masked data," *Microelectron. Rel.*, vol. 34, no. 6, pp. 1027–1038, Jun. 1994, doi: 10.1016/0026-2714(94)90067-1.

[30] B. Zhao, J. Yang, M. Zhao, Q. Li, and Y. Liu, "Wireless sensor network reliability modelling based on masked data," *Int. J. Sensor Netw.*, vol. 17, no. 4, pp. 217–223, 2015.

[31] J. Cai, Y. Shi, and H. Yue, "Accelerated life tests for log-normal series system with dependent masked data under Type-I progressive hybrid censoring," *Commun. Statist. Simul. Comput.*, vol. 46, no. 2, pp. 1628–1646, Feb. 2017, doi: 10.1080/03610918.2015.1045078.

[32] A. S. Rodrigues, C. A. D. B. Pereira, and A. Polpo, "Estimation of component reliability in coherent systems with masked data," *IEEE Access*, vol. 7, pp. 57476–57487, 2019, doi: 10.1109/access.2019.2913675.

[33] B. Liu, Y. Shi, J. Cai, X. Bai, and C. Zhang, "Nonparametric Bayesian analysis for masked data from hybrid systems in accelerated lifetime tests," *IEEE Trans. Rel.*, vol. 66, no. 3, pp. 662–676, Sep. 2017, doi: 10.1109/tr.2017.2704582.

[34] Y. Zhang, M. Zhao, Y. Zhang, R. Pan, and J. Cai, ''Dynamic and steady-state performance analysis for multi-state repairable reconfigurable manufacturing systems with buffers,'' *Eur. J. Oper. Res.*, vol. 283, no. 2, pp. 491–510, Jun. 2020.
[35] Y. Zhang, M. Zhao, S. Zhang, J. Wang, and Y. Zhang, ''An integrated approach to estimate storage reliability with initial failures based on E-Bayesian estimates,'' *Rel. Eng. Syst. Saf.*, vol. 159, pp. 24–36, Mar. 2017.
[36] S. Yamada, M. Ohba, and S. Osaki, ''S-shaped software reliability growth models and their applications,'' *IEEE Trans. Rel.*, vol. R-33, no. 4, pp. 289–292, Oct. 1984.
[37] M. Xie, *Software Reliability Modelling*. Singapore: World Scientific, 1991.
[38] A. P. Dempster, N. M. Laird, and D. B. Rubin, ''Maximum likelihood from incomplete data via the EM algorithm,'' *J. Roy. Statist. Soc., B, Methodol.*, vol. 39, no. 1, pp. 1–38, 1977. [Online]. Available: www.jstor.org/stable/2984875

JIANFENG YANG received the B.S. and M.S. degrees in mathematics, in 2008 and 2011, respectively, and the Ph.D. degree in software engineering from Guizhou University, in 2014. He is currently an Associate Professor with the School of Data Science, Guizhou Institute of Technology. His research interests include reliability modeling and applied statistics.

JING CHEN received the B.S. and M.S. degrees in mathematics, in 2009 and 2012, respectively. She is currently a Teacher with the College of Information Engineering, Guizhou University of Traditional Chinese Medicine. Her research interests include intelligent optimization algorithm and applied statistics.

XIBIN WANG received the M.S. degree in computer science from Guizhou University, in 2012, and the Ph.D. degree from the College of Computer Science, Chongqing University, in 2015. Since 2017, he has been an Associate Professor with the School of Big Data, Guizhou Institute of Technology. His research interests include computational intelligence, data mining and business intelligence, and machine learning.