



A Fine-grained parametric bootstrap approach for NHPP-based software reliability modeling

Jingchi Wu , Tadashi Dohi , Junjun Zheng , Hiroyuki Okamura

Hiroshima University, Japan

ARTICLE INFO

Prof W. Eric Wong

Keywords:

Software reliability model
NHPP
Parametric bootstrapping
Interval estimation
Thinning-like resampling
Monte carlo simulation
Empirical study

ABSTRACT

In software reliability, practitioners demand to estimate software reliability measures accurately from software fault-count data, for making the release decision and project management. To achieve these objectives, software fault-count processes are often described using software reliability models (SRMs) based on stochastic counting processes like non-homogeneous Poisson processes (NHPPs), and statistical point estimation of model parameters is carried out. Substituting the point estimates of model parameters into several software reliability measures, one gets the point estimates of desired reliability measures. However, since such point estimators tend to have high variances, the resulting release decision and project management plans are not reliable under uncertainty. Then, interval estimation of software reliability measures is expected to realize more robust decision making, but is quite difficult to obtain the analytical confidence regions. Bootstrap is a statistical method that generates realizations of statistical estimators by resampling fault-count data. It allows us to evaluate the statistical properties of software reliability measures under uncertainty. In this paper, we propose a fine-grained parametric bootstrap method for NHPP-based SRMs, where a thinning-like resampling algorithm is employed instead of intuitive resampling algorithms which generate the bootstrap data with ties problem. We compare our thinning-like resampling algorithm with the existing ones in both Monte Carlo simulation and empirical study. It can be shown that the model parameters and their associated software reliability measures estimated by our fine-grained parametric bootstrap method are more accurate and robust than the other bootstrap algorithms.

1. Introduction

Software reliability is one of the key measures in software testing progress, software quality assurance, and software release engineering. It is important to develop the quantitative methods to evaluate software reliability and its associated measures, where the quantitative software reliability is defined as a probability that software system does not fail during a specified time period. From the reasons above, during the last five decades, a large number of software reliability models (SRMs) have been proposed from various perspectives. Among them, non-homogeneous Poisson process (NHPP)-based SRMs are widely used in software reliability modeling due to their analytical tractability. It is known that the NHPP can be characterized by the mean value function, which is an absolutely continuous and non-decreasing function. Goel and Okumoto [Goel and Okumoto \(1979\)](#) proposed the seminal exponential NHPP-based SRM under the assumption that software fault-detection times are given by mutually independent and identically ex-

ponential distributed random variables. Latter, Yamada et al. [Yamada et al. \(1983\)](#), Ohba [Ohba \(1984\)](#), Musa and Okumoto [Musa and Okumoto \(1984\)](#), Goel [Goel \(1985\)](#), Abdel-Ghaly et al. [Abdel-Ghaly et al. \(1986\)](#), Zhao and Xie [Zhao and Xie \(1996\)](#), Achcar et al. [Achcar et al. \(1998\)](#), Gokhale and Trivedi [Gokhale and Trivedi \(1998\)](#), Ohishi et al. [Ohishi et al. \(2009\)](#), among others, developed representative NHPP-based SRMs under different software fault-detection time distribution scenarios.

Once the mean value function in an NHPP-based SRM is given, the commonly used technique to estimate the model parameters is the maximum likelihood (ML) estimation. Hossain and Dahiya [Hossain and Dahiya \(1993\)](#) and Knafl and Morgan [Knafl and Morgan \(1996\)](#) investigated analytically the maximum likelihood equations in the NHPP-based SRM with the exponential mean value function [Goel and Okumoto \(1979\)](#). In the subsequent paper, the same authors [Hossain and Dahiya \(1992\)](#) obtained similar results for the delayed S-shaped NHPP-based SRM [Yamada et al. \(1983\)](#). Although the ML estimation is a

E-mail addresses: kure@hiroshima-u.ac.jp (J. Wu), dohi@hiroshima-u.ac.jp (T. Dohi), jzheng@hiroshima-u.ac.jp (J. Zheng), okamu@hiroshima-u.ac.jp (H. Okamura).

<https://doi.org/10.1016/j.jss.2026.112789>

Received 15 November 2025; Received in revised form 13 January 2026; Accepted 16 January 2026

Available online 18 January 2026

0164-1212/© 2026 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY-NC license (<http://creativecommons.org/licenses/by-nc/4.0/>).

representative point estimation, it is worth pointing out that the ML point estimates of model parameters are hard to be obtained analytically, due to their strong non-linearity nature. Especially, the general-purpose optimization algorithms to derive the ML estimates strongly depend on the initial guess, and do not possess global convergency. To get the numerical stability in the ML point estimation, Okamura et al. (2003), Okamura and Dohi (2013b,a, 2021) developed the EM (Expectation-Maximization) algorithms for NHPP-based SRMs. Recently, Yano et al. Yano et al. (2024) compared several ML estimation algorithms for NHPP-based SRMs in terms of global convergency.

Apart from the numerical computation algorithms for the point estimation, it is known that the ML point estimates of NHPP-based SRMs are statistically inconsistent, and do not converge to the *true* parameters even as the number of fault-count data increases Jeske and Pham (2001), Nayak et al. (2008), Zhao and Xie (1996). This fact motivates us to consider the interval estimation to give more robust software reliability assessment. Yamada and Osaki Yamada and Osaki (1985b) proposed a frequentist method for the interval estimation in exponential NHPP-based SRM and delayed S-shaped SRM. Joe Joe (1989) corrected the theoretical issues in Yamada and Osaki (1985b), and developed asymptotic interval estimates of NHPP-based SRMs. van Pul Pul (1992), Zhao and Xie Zhao and Xie (1996) also examined the interval estimates of model parameters in some specific NHPP-based SRMs, which were also based on asymptotic approximations. On one hand, several authors developed the interval estimation methods for NHPP-based SRMs from the Bayesian statistical points of view. Kuo and Yang Kuo and Yang (1996), Yin and Trivedi Yin and Trivedi (1999), Okamura et al. Okamura et al. (2007) applied the Markov chain Monte Carlo, numerical integration and variational Bayes approaches, respectively, to the NHPP-based SRMs under a few prior distribution assumptions.

Even in both ML and Bayesian estimations, it is still a challenging issue to obtain the confidence intervals of arbitrary reliability measures, such as mean value function, intensity function, the residual number of software faults, software reliability function. For instance, even though two-sided confidence regions of model parameters are given with a certain significance level, one cannot obtain the two-sided confidence region of software reliability measure with the same significance level. For this essentially technical issue, it has been recognized that bootstrap was a powerful statistical approach to generate approximately realizations of any estimator through resampling from the underlying data, and enabled us to investigate the statistical properties even with a small sample Efron (1979, 1981), Efron and Tibshirani (1993). Nowadays, it has been a well-known computer-intensive and general-purpose technique, which belonged to a broader class of resampling methods, for obtaining the interval estimates of any statistical measure.

van Pul Pul (1992) was the first work to apply bootstrap methods to a specific SRM, and employed a parametric bootstrap method in the well-known Jelinski-Morana SRM (see Lyu (1996), Musa et al. (1987), Pham (2006)). Kimura and Fujiwara Kimura and Fujiwara (2011) applied a bootstrap approach to a moving-average type SRM in an estimation framework of mean squares error method. In a different context from the software reliability estimation, Lei and Smith Lei and Smith (2002, 2003) used a non-parametric bootstrap technique to estimate the confidence intervals of software metrics. Cowling et al. Cowling et al. (1996) developed the bootstrap methods for constructing confidence regions for the intensity function of a non-homogeneous Poisson process. Following a significant work by Cowling et al. Cowling et al. (1996), the bootstrap techniques for NHPPs can be classified into two types; simulation-based bootstrapping and resampling-based bootstrapping. Kaneishi and Dohi Kaneishi and Dohi (2011) developed a Monte-Carlo simulation-based parametric bootstrapping and two resampling-based parametric bootstrapping methods for an exponential NHPP-based SRM Goel and Okumoto (1979), and evaluated the accuracy of their methods in both simulation experiment and actual project data analysis. The same authors Kaneishi and Dohi (2013) considered kernel-based non-parametric bootstrap techniques to estimate the software reliability measures, where

the three bootstrap methods employed also followed Cowling et al.'s methods Cowling et al. (1996). Xiao and Dohi Xiao and Dohi (2014) used the similar idea to Kaneishi and Dohi (2013) for development of a wavelet-based non-parametric bootstrap technique for NHPP-based SRM. Recently, Xiao and Dohi Xiao and Dohi (2021) compared two simulation-based bootstrap technique in software reliability assessment; the common time-scale transformation and the thinning algorithm by Lewis and Shedler Lewis and Shedler (1979).

Unfortunately, it should be noted that the resampling-based bootstrap methods employed in Cowling et al. (1996), Kaneishi and Dohi (2011, 2013) result in severe *ties* problem such that the exactly same data may be resampled with positive probability. More specifically, in Kaneishi and Dohi (2011, 2013), the authors applied an elementary random sampling scheme for an NHPP-based SRM, where each data point is picked with a constant probability with replacement. However, this approach could result in resampled data with ties and eventually the ML estimation could not work for the tie data. In this paper, we re-focus on the resampling-based bootstrapping and propose a fine-grained parametric bootstrap method for NHPP-based SRMs. It is worth mentioning that the accuracy of resampling-based bootstrap method depends on the quality of resampling scheme. We evaluate the accuracy of interval estimation of our bootstrap methods in both simulation experiments and actual software development project data analysis, and compare our thinning-like resampling-based algorithm with the existing three bootstrap method through a comprehensive study with eleven NHPP-based SRMs.

The paper is organized as follows. In Section 2 we summarize NHPP-based SRMs and the their maximum likelihood estimation. Section 3 overviews the existing three parametric bootstrap methods for NHPP-based SRMs, and propose our thinning-like bootstrap method. The proposed resampling scheme is investigated through Monte Carlo simulation in Section 4. We also conduct actual software development project data analysis, and compare with three existing bootstrap methods in Section 5. Section 6 concerns a software release problem to determine the optimal release time from the view point of interval estimation. Finally, we conclude the paper with some remarks in Section 7.

2. Software reliability modeling

2.1. NHPP-Based SRMs

Suppose that software fault-detection process is described by a non-homogeneous Poisson process (NHPP). Let $\{N(t), t \geq 0\}$ be an NHPP, where $N(t)$ denotes the cumulative number of software faults detected up to testing time t . Then the probability mass function of $N(t)$ is given by

$$\Pr\{N(t) = n\} = \frac{\{\Lambda(t; \theta)\}^n e^{-\Lambda(t; \theta)}}{n!}, \quad (1)$$

where $\Lambda(t; \theta) = E[N(t)]$ is the mean value function, $\lambda(t; \theta) = d\Lambda(t; \theta)/dt$ is the intensity function, and θ is the model parameter vector. It is clear that an NHPP is uniquely determined by the mean value function. In finite NHPP-based SRMs with $\Lambda(t; \theta) < \infty$ for any time $t (> 0)$, the mean value function and intensity function can be written in the following forms;

$$\Lambda(t; \theta) = aG(t; \theta'), \quad (2)$$

$$\lambda(t; \theta) = ag(t; \theta'), \quad (3)$$

where $0 \leq G(t; \theta') \leq 1$ and $g(t; \theta') = dG(t; \theta')/dt$ are the cumulative distribution function and the probability density function of each fault-detection time in a program, respectively, and $\theta = (a, \theta')$. It is easily seen that $\lim_{t \rightarrow \infty} \Lambda(t; \theta) = a$, so the parameter $a (> 0)$ is interpreted as the expected initial number of fault contents before the testing. Table 1 presents the mean value functions of the eleven representative NHPP-based SRMs implemented in SRATS Okamura and Dohi (2013b).

Suppose that the software test starts at time $t = 0$ and terminates at time $t = t_u$. After releasing the software at time t_u to the users or

Table 1
Representative eleven NHPP-based SRMs.

Mean Value Function	$\Lambda(t; \theta)$ ($\theta = a, b, c > 0$)
Exp Goel and Okumoto (1979)	$\Lambda(t; \theta) = a(1 - e^{-bt})$
Gamma Yamada et al. (1983), Zhao and Xie (1996)	$\Lambda(t; \theta) = a \int_0^t \frac{c^b s^{b-1} e^{-cs}}{\Gamma(b)} ds$
Pareto Abdel-Ghaly et al. (1986)	$\Lambda(t; \theta) = a \left(1 - \left(\frac{c}{t+c} \right)^b \right)$
TruncNormal Okamura et al. (2013)	$\Lambda(t; \theta) = a \frac{F(t) - F(0)}{1 - F(0)}$, $F(t) = \frac{1}{\sqrt{2\pi b}} \int_{-\infty}^t e^{-\frac{(s-c)^2}{2b^2}} ds$
LogNormal Achcar et al. (1998), Okamura et al. (2013)	$\Lambda(t; \theta) = a \frac{1}{\sqrt{2\pi b}} \int_{-\infty}^{\log(t)} e^{-\frac{(s-c)^2}{2b^2}} ds$
TruncLogist Ohba (1984)	$\Lambda(t; \theta) = a \frac{\frac{1}{1+e^{-\frac{t-c}{b}}}}{1 - F(0)}$, $F(t) = \frac{1}{1+e^{-\frac{t-c}{b}}}$
LogLogist Gokhale and Trivedi (1998)	$\Lambda(t; \theta) = a \frac{1}{1+e^{-\frac{\log(t)-c}{b}}}$
TruncEVMMax Ohishi et al. (2009)	$\Lambda(t; \theta) = a \frac{1 + e^{-\frac{F(t)-F(0)}{1-F(0)}}}{1 - F(0)}$, $F(t) = e^{-e^{-\frac{t-c}{b}}}$
LogEVMMax Ohishi et al. (2009)	$\Lambda(t; \theta) = ae^{-e^{-\frac{\log(t)-c}{b}}}$
TruncEVMMin Ohishi et al. (2009)	$\Lambda(t; \theta) = a \frac{F(t) - F(-t)}{F(0)}$, $F(t) = e^{-e^{-\frac{t-c}{b}}}$
LogEVMMin Goel (1985), Ohishi et al. (2009)	$\Lambda(t; \theta) = a \left(1 - e^{-e^{-\frac{\log(t)-c}{b}}} \right)$

market, one predicts the quantitative software reliability $R(t_u, t_s; \theta)$ at time t_s ($> t_u$). The $R(t_u, t_s; \theta)$ is a probability that software is fault-free during the operational phase $(t_u, t_s]$. From Eq. (1), we obtain the software reliability $R(t_u, t_s; \theta)$ as

$$R(t_u, t_s; \theta) = e^{-\{\Lambda(t_s; \theta) - \Lambda(t_u; \theta)\}}. \quad (4)$$

2.2. Parameter estimation

Once the NHPP-based SRM is given, it is necessary to estimate the model parameter θ from the observed fault-count data. In general, θ are estimated by means of the ML estimation. Suppose that the software time-domain fault-count data, $t = \{t_1, t_2, \dots, t_n \leq t_u\}$, are available, where t_i ($i = 1, 2, \dots, n$) is the i -th software fault-detection time. In the failure truncation with $t_n = t_u$, the likelihood function of model parameters in the NHPP-based SRM is given by

$$\mathcal{L}(\theta; t) = e^{\lambda(t_n; \theta)} \prod_{i=1}^n \lambda(t_i; \theta). \quad (5)$$

By taking the logarithm of Eq. (5), we have the log-likelihood function as follows:

$$\ln \mathcal{L}(\theta; t) = \sum_{i=1}^n \ln \lambda(t_i; \theta) - \Lambda(t_n; \theta). \quad (6)$$

Then, we obtain the ML point estimate of model parameters by

$$\hat{\theta} = \arg \max_{\theta} \ln \mathcal{L}(\theta; t). \quad (7)$$

2.3. Interval estimation

The ML point estimates of model parameters, $\hat{\theta}$, can be regarded as functions of the observed software fault-detection time data, $t = \{t_1, t_2, \dots, t_n\}$. These observations, t_i ($i = 1, 2, \dots, n$), are realizations of software fault-detection times T_i which are random variables. Therefore, the ML point estimators are also random variables and functions of T_i denoting the i -th fault-detection time. The interval estimation is to derive the one-sided or two-sided confidence regions of the ML point estimators with a given significance level. In the exponential NHPP-based SRM Goel and Okumoto (1979), since estimators of two model parameters are written as $\hat{a}(T_1, T_2, \dots, T_n)$ and $\hat{b}(T_1, T_2, \dots, T_n)$, the problems are to derive the probability distributions of these two random variables. The earlier papers Joe (1989), Yamada and Osaki (1985b), Zhao and Xie (1996) derived approximately the two-sided confidence regions of the two parameters under the assumption of asymptotic normality. Kuo

Table 2

Comparison on median of model parameters estimated by four bootstrap methods.

Parameter $a = 150$				
Method Name	Median	Average	Variance	95% Interval
BS1	151.00	152.32	135.64	[134.6, 172.2]
BS2	151.85	152.35	289.23	[125.4, 181.4]
BS3	148.70	150.25	130.44	[132.5, 170.5]
BS4	152.40	153.02	284.40	[126.4, 181.7]

Parameter $b = 0.00005$				
Method Name	Median	Average	Variance	95% Interval
BS1	0.00004854	0.00004877	5.04176E-11	[3.761e-05, 6.089e-05]
BS2	0.00004854	0.00004875	5.20673E-11	[3.743e-05, 6.098e-05]
BS3	0.00004962	0.00004937	2.75582E-11	[4.043e-05, 5.792e-05]
BS4	0.00004795	0.00004815	4.98046E-11	[3.692e-05, 6.015e-05]

Table 3

Comparison on KL divergence and Hellinger distance for four bootstrap methods.

KL divergence				
Method	Median	Average	Variance	95% Confidence
Name	Interval			
BS1	0.794	1.162	1.357	[0.051, 3.438]
BS2	1.179	1.671	2.644	[0.089, 4.869]
BS3	0.578	0.809	0.583	[0.026, 2.237]
BS4	1.154	1.649	2.599	[0.086, 4.877]

Hellinger distance				
Method	Median	Average	Variance	95% Confidence
Name	Interval			
BS1	0.181	0.225	0.033	[0.013, 0.579]
BS2	0.257	0.297	0.046	[0.022, 0.704]
BS3	0.135	0.169	0.020	[0.006, 0.426]
BS4	0.251	0.294	0.046	[0.021, 0.706]

and Yang Kuo and Yang (1996), Okamura et al. Okamura et al. (2007), and Yin and Trivedi Yin and Trivedi (1999) applied Markov chain Monte Carlo, variational Bayes, and direct numerical integration methods, respectively. These approaches were used for Bayesian interval estimation under given prior distributions of the model parameters. In the following section, we concern non-Bayesian (frequentist) approaches based on the parametric bootstrapping and improve the underlying resampling algorithm.

3. Parametric bootstrapping

3.1. Existing methods

Bootstrap is a statistical method which can generate realizations from an unknown distribution. Kaneishi and Dohi Kaneishi and Dohi (2011) applied three bootstrap methods by Cowling et al. Cowling et al. (1996), referred to as Method 1, Method 2, Method 3 in their paper. Here, we briefly summarize these three methods as follows. For single time series data, $t = \{t_1, t_2, \dots, t_n\}$, we generate ϕ time series data, $\hat{t}_k = \{\hat{t}_1, \hat{t}_2, \dots, \hat{t}_{n_k}\}$ ($k = 1, 2, \dots, \phi$), by one of the following three methods, where $\hat{t}_{n_k} \leq t_n$ is assumed. Method 1 (Simulation-based bootstrap) Cowling et al. (1996): Given the ML estimates $\hat{\theta}$ from the observed data t , we use the plug-in ML estimate of the intensity function, $\lambda^*(t; \hat{\theta})$. We then generate ϕ time-series datasets as pseudo-random numbers, with dataset sizes n_k ($k = 1, 2, \dots, \phi$), by applying a pseudo-random number generator such as the thinning algorithm Lewis and Shedler (1979). The pseudo-code for Method 1 is given by Algorithm 11.

Method 2 and Method 3 (Resampling-based bootstrap) Cowling et al. (1996): Cowling et al. Cowling et al. (1996) proposed two

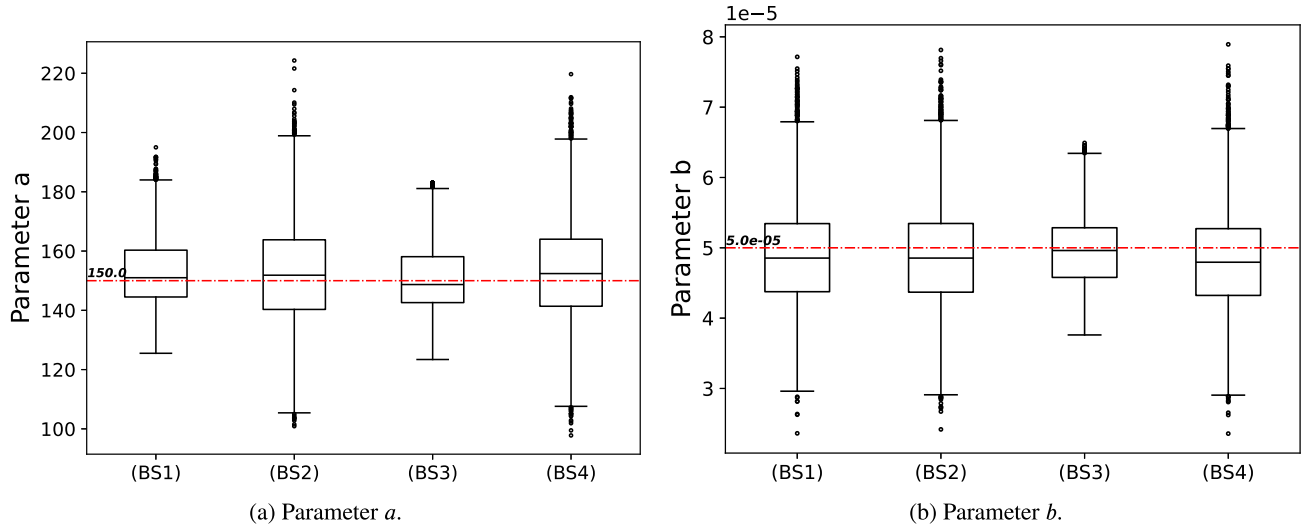


Fig. 1. Box plots of model parameters in exponential NHPP-based SRM with three bootstrap methods in simulation experiment.

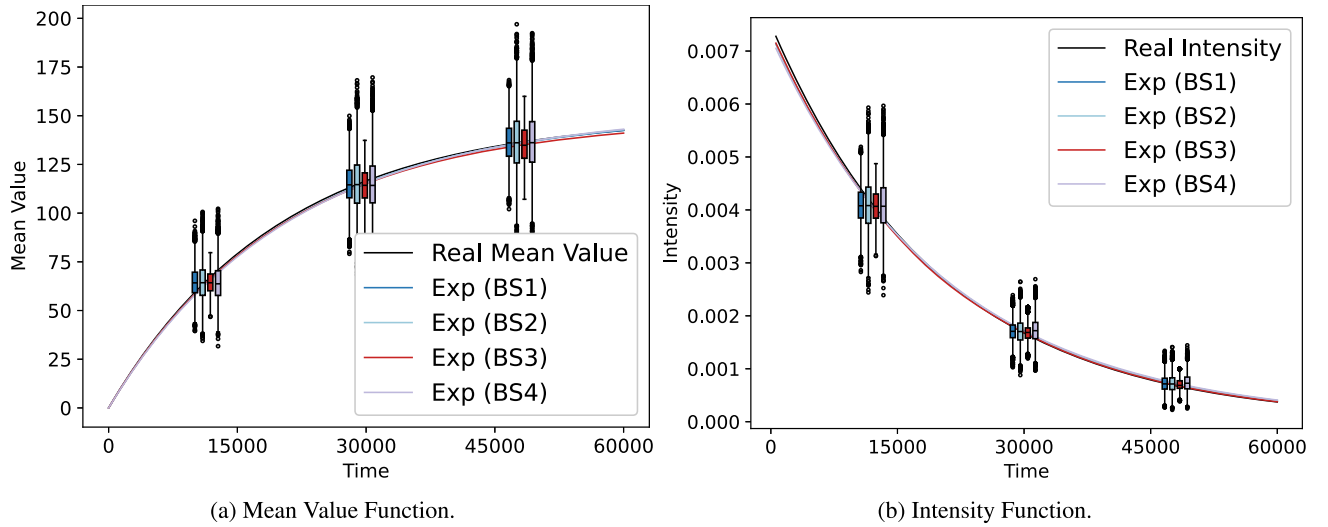
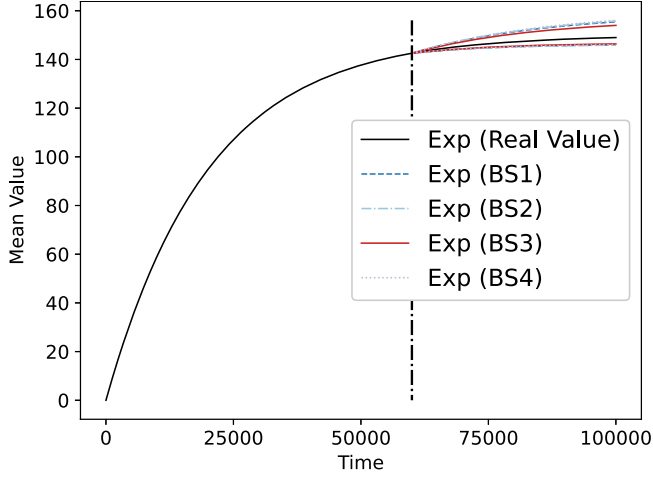


Fig. 2. Confidence intervals of mean value function and intensity function in simulation experiment..

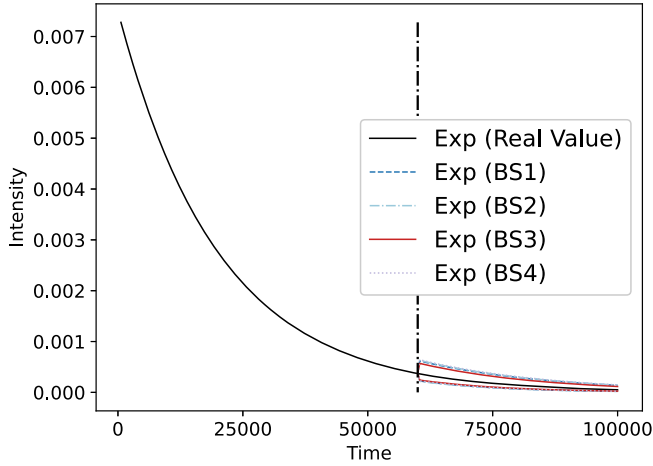
Table 4

Summary of software development project data.

Data Set	Total Number of Faults	Testing Length (CPU sec)	Project Name	System Description
DS1	256	158871485	tabler Aaa (2018a)	A HTML Dashboard UI Kit built on bootstrap
DS2	285	57501263	astro Aaa (2021)	The web framework for content-driven websites
DS3	314	132377444	notable Aaa (2018b)	The Markdown-based note-taking application
DS4	212	136309329	caffe Aaa (2013)	A fast open framework for deep learning
DS5	293	166129498	Monitor Control Aaa (2018c)	A controller for your display on your Mac.
DS6	229	295419627	bootstrap Aaa (2011)	The most popular HTML, CSS, and JavaScript framework
DS7	335	325373644	hexo Aaa (2012a)	A fast and powerful blog framework, powered by Node.js
DS8	152	346633131	HTTPIeCi Aaa (2012b)	A modern command-line HTTP client for the API era



(a) Mean Value Function.



(b) Intensity Function.

Fig. 3. Predictive two-sided confidence intervals of mean value function and intensity function in simulation experiment.

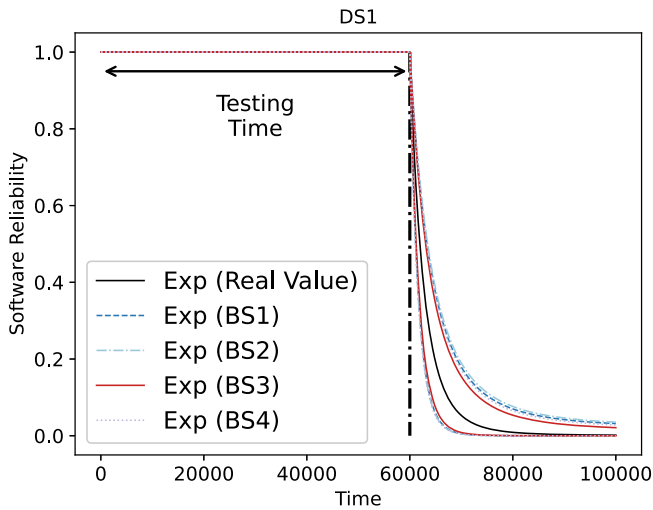


Fig. 4. Predictive two-sided confidence intervals of software reliability function in simulation experiment.

Table 5

Optimal software release times with varying c_2 .

c_1	c_2	c_3	Theoretical Results			
10	1	5	150070			
10	2	5	144316			
10	3	5	136207			
10	4	5	122344			
c_1	c_2	c_3	BS1			
			Median	Average	Variance	95% Confidence Interval
10	1	5	154,277	156,317	422,780,885	[126188,192980]
10	2	5	148,343	150,289	386,657,967	[121455,185306]
10	3	5	139,986	141,794	338,501,730	[114776,174551]
10	4	5	125,712	127,271	263,645,942	[103385,156187]
c_1	c_2	c_3	BS2			
			Median	Average	Variance	95% Confidence Interval
10	1	5	154,222	156,402	443,931,460	[125532,194026]
10	2	5	148,276	150,367	406,138,730	[120817,186336]
10	3	5	139,884	141,862	355,766,049	[114156,175565]
10	4	5	125,584	127,322	277,492,522	[102789,157071]
c_1	c_2	c_3	BS3			
			Median	Average	Variance	95% Confidence Interval
10	1	5	150942	153122	213,858,024	[131150,182271]
10	2	5	145161	147228	195,745,267	[126235,175185]
10	3	5	137012	138920	171,593,661	[119172,165198]
10	4	5	122978	124719	134,037,001	[107005,148126]
c_1	c_2	c_3	BS4			
			Median	Average	Variance	95% Confidence Interval
10	1	5	155,908	158,133	445,242,977	[127532,196331]
10	2	5	149,911	152,026	407,337,398	[122721,188516]
10	3	5	141,432	143,418	356,813,570	[115944,177577]
10	4	5	126,964	128,703	278,303,092	[104371,158980]

Algorithm 1: Method 1 (Simulation-based bootstrap) Algorithms.

Data: Observed software time-domain fault-count data

$t = (t_1, t_2, \dots, t_n)$, number of resampling ϕ , a given intensity function $\lambda(t; \theta)$.

Result: The intensity functions $\{\tilde{\lambda}(t; \tilde{\theta}_1), \tilde{\lambda}(t; \tilde{\theta}_2), \dots, \tilde{\lambda}(t; \tilde{\theta}_\phi)\}$

```

1 # Obtain the ML estimates of model parameters  $\tilde{\theta}$  with
  observed data  $t$ ;
2  $\tilde{\theta} = \arg \max_{\theta} \ln \mathcal{L}(\theta; t)$ ;
3  $k = 1$ ;
4 while  $k \leq \phi$  do
5   Generate the simulation data  $\hat{t}_k$  from  $\lambda(t; \tilde{\theta})$  by thinning
    algorithm Lewis and Shedler (1979);
6   # Obtain the ML estimates of model parameters  $\tilde{\theta}_k$  with
    simulation data  $\hat{t}_k$ ;
7    $\tilde{\theta}_k = \arg \max_{\theta} \ln \mathcal{L}(\theta; \hat{t}_k)$ ;
8   # Obtain the intensity function for bootstrap
     $\tilde{\lambda}(t; \tilde{\theta}_k) = \lambda(t; \tilde{\theta}_k)$ ;
9   # Counter increment;
10   $k = k + 1$ ;
11 end
```

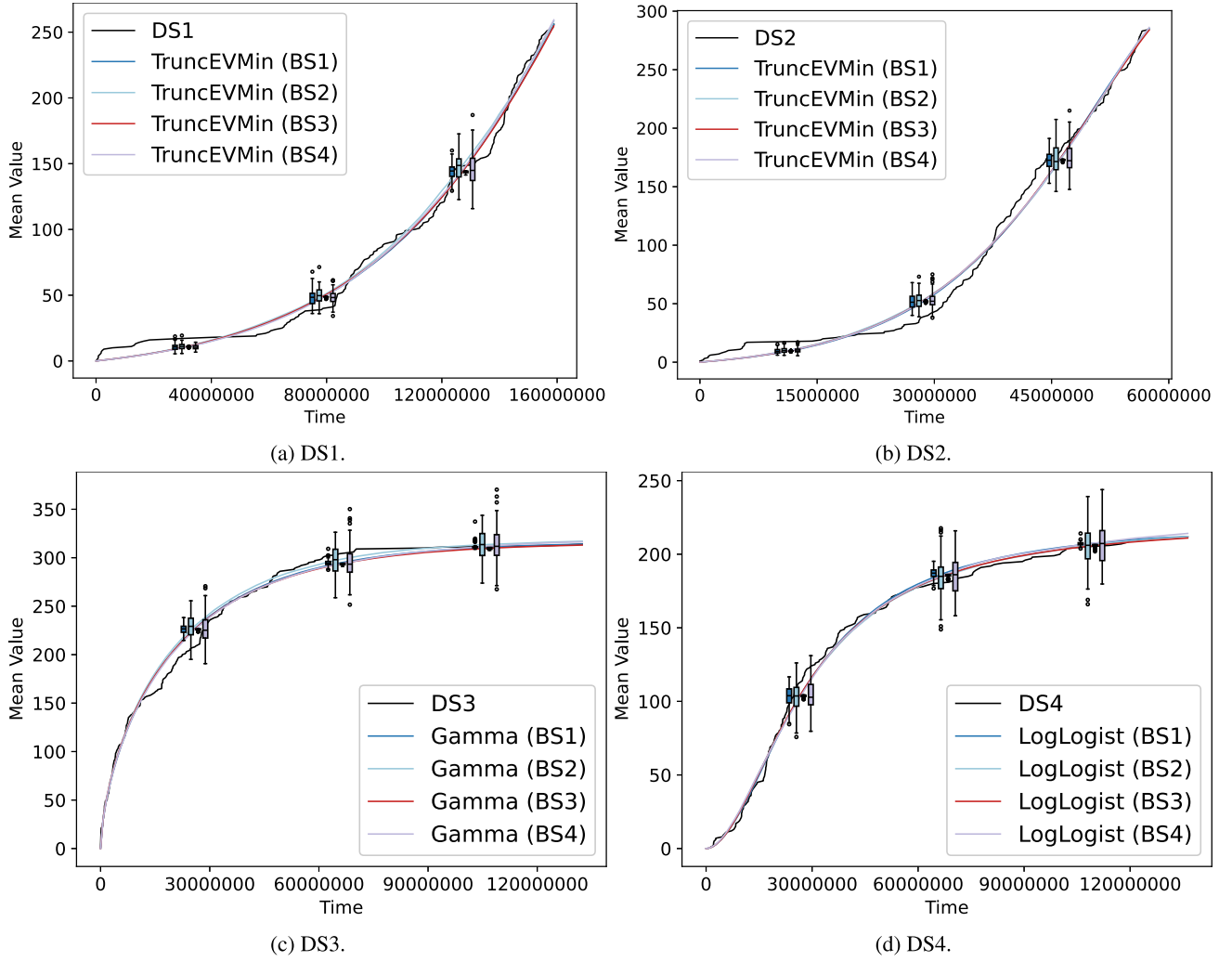


Fig. 5. Confidence intervals of mean value function with DS1 ~ DS4 (100%). .

resampling-based bootstrap algorithms for NHPP, using the following fact. Conditioned on the data length n , $t = \{t_1, t_2, \dots, t_n\}$ can be regarded as the order statistics of a realization of size n obtained from the uniform distribution with density $f(t) = \lambda(t)/\Lambda(t_n)$, where $t \geq t_n$. In Method 2, we pick the sample from t randomly with replacement n times, and form the resampled data sequence, $\hat{t}_k = \{\hat{t}_1, \hat{t}_2, \dots, \hat{t}_{n_k}\}$, where $n_k = n$. More specifically, in one resampling procedure, each sample data is picked with same probability $1/n$. In Method 3, by contrast, the resampled data $\hat{t}_k = \{\hat{t}_1, \hat{t}_2, \dots, \hat{t}_{n_k}\}$ are generated by randomly sampling from t with replacement. The sampling is repeated n_k times, where $n_k \sim P(n)$ and $P(n)$ is the Poisson distribution with parameter n . It is easily seen that Method 2 and Method 3 in Cowling et al. (1996) may produce resampled data that include ties, whereas the simulation-based bootstrap, Method 1, does not. The pseudo-code for Method 2 and Method 3 are given by Algorithm 9 and Algorithm 10.

3.2. Refinement

Three bootstrap methods in Cowling et al. (1996), Kaneishi and Dohi (2011); Method 1, Method 2 and Method 3, have some theoretical issues. First, Method 1 is based on the Monte Carlo approximation for the underlying NHPP with the ML estimates. Since the ML estimate from the single fault-count time series data does not possess the statistical consistency Jeske and Pham (2001), Nayak et al. (2008), Zhao and Xie

Algorithm 2: Method 2 (Resampling-based bootstrap) Algorithms.

Data: Observed software time-domain fault-count data

$t = (t_1, t_2, \dots, t_n)$, number of resampling ϕ , a given intensity function $\lambda(t; \theta)$.

Result: The intensity functions $\{\tilde{\lambda}(t; \tilde{\theta}_1), \tilde{\lambda}(t; \tilde{\theta}_2), \dots, \tilde{\lambda}(t; \tilde{\theta}_\phi)\}$

```

1  $k = 1$ ;
2 while  $k \leq \phi$  do
3   Generate the resampling data  $\hat{t}_k$  by pick the sample from  $t$ 
   randomly with replacement  $n$  times;
4   # Obtain the ML estimates of model parameters  $\tilde{\theta}_k$  with
   simulation data  $\hat{t}_k$ ;
5    $\tilde{\theta}_k = \arg \max_{\theta} \ln \mathcal{L}(\theta; \hat{t}_k)$ ;
6   # Obtain the intensity function for bootstrap
    $\tilde{\lambda}(t; \tilde{\theta}_k) = \lambda(t; \tilde{\theta}_k)$ ;
7   # Counter increment;
8    $k = k + 1$ ;
9 end

```

(1996), the resulting estimate of an arbitrary reliability measure must have any bias even in a large sample case. Hereafter, we concern only the resampling-based parametric bootstrap methods, referred to as Method

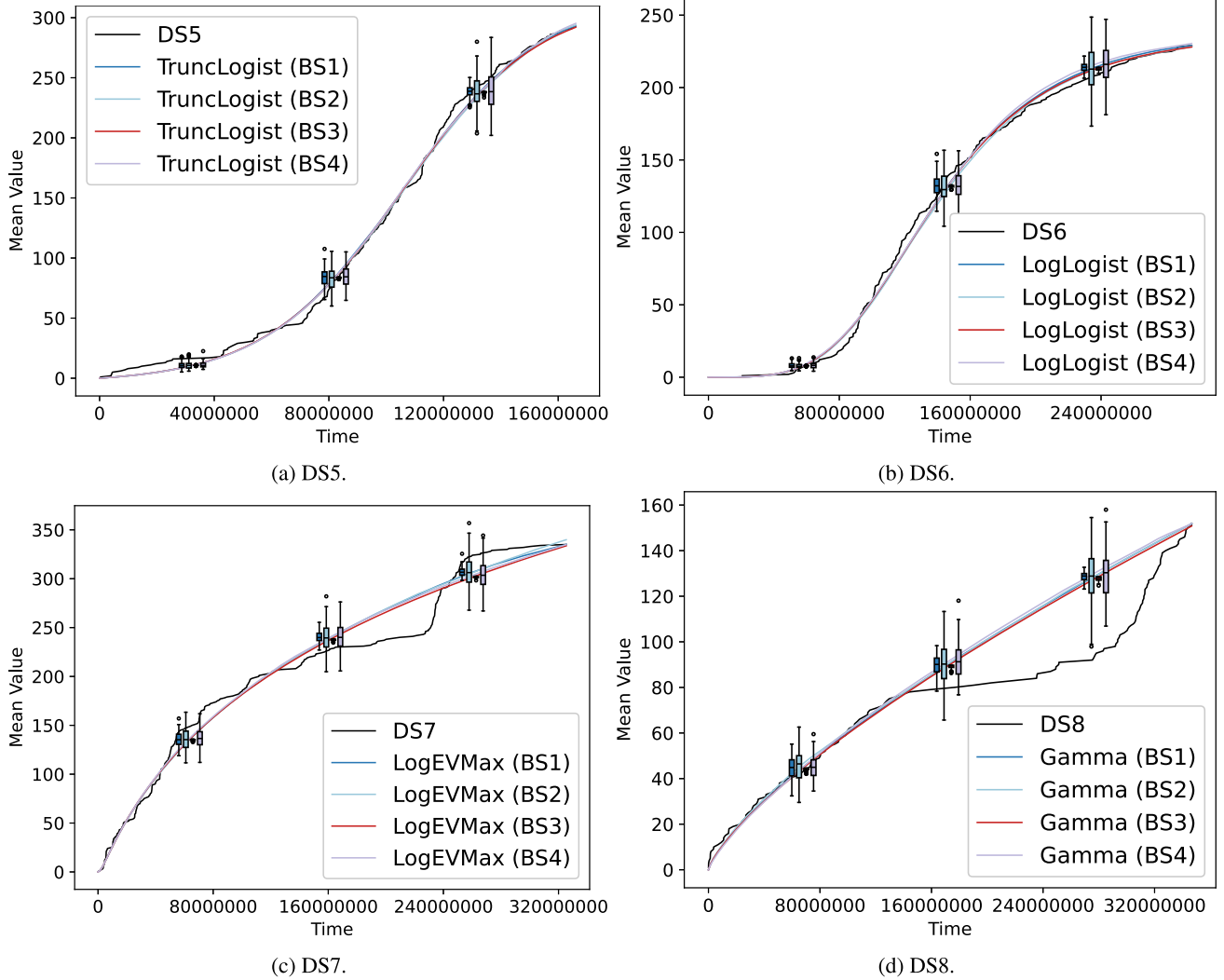


Fig. 6. Confidence intervals of mean value function with DS5 ~ DS8 (100%). .

2 and Method 3 in Kaneishi and Dohi (2011). As noted in Cowling et al. (1996), our concern is to overcome the tie problem in Method 2 and Method 3.

First, we begin from the statement by Lewis and Shedler Lewis and Shedler (1979):

Lemma 1: Consider an NHPP $\{N(t), t \geq 0\}$ with intensity function $\lambda(t; \theta)$ and mean value function $\Lambda(t; \theta)$, so that the number of events, $N(t_n)$, in a fixed interval $(0, t_n]$ has a Poisson distribution with parameter $\Lambda(t_n; \theta) - \Lambda(0; \theta)$. Let $T_1, T_2, \dots, T_{N(t_n)}$ be the event-occurrence points of the process in the interval $(0, t_n]$. Assume that $\lambda(t; \theta) \leq \lambda(t; \theta)$ for $0 \leq t \leq t_n$. That is to say, there exists the infimum of $\lambda(t; \theta)$ such as $\lambda(t; \theta)$. Then, for $i = 1, 2, \dots, N(t_n)$, delete the point T_i with probability $\mu_i = 1 - (\lambda(T_i; \theta) / \lambda(T_i; \theta))$. The remaining points are drawn from the NHPP $\{N(t), t \geq 0\}$ with intensity function $\lambda(t; \theta)$ in the interval $(0, t_n]$.

Suppose again that the observed data, $t = \{t_1, t_2, \dots, t_n\}$, are realizations of $\{T_1, T_2, \dots, T_n\}$ of an unknown NHPP, $\{N(t), t \geq 0\}$, with unknown intensity function $\lambda(t; \theta)$. If we remove each time point t_i with same probability $\mu_i = 1/n$, then the remaining data points can be regarded as realizations, $\hat{t}_k = \{\hat{t}_1, \hat{t}_2, \dots, \hat{t}_{n_k}\}$ ($n_k \leq n$), from another NHPP, $\{N(t), t \geq 0\}$ with unknown intensity function $\lambda(t; \theta)$. From Lemma 1, $\lambda(t; \theta)$ is a lower bound of $\lambda(t; \theta)$, and has the following relationship;

$$\lambda(t_i; \theta) = \frac{1}{1 - 1/n} \cdot \lambda(t_i; \theta). \quad (8)$$

From Eq.(8), one can obtain estimates of $\lambda(t; \theta)$ from the resampled data $\hat{t}$ with the correction coefficient $\alpha = 1/(1 - 1/n)$. When the above procedure is repeated ϕ times, then the interval estimation can be obtained by using the multiple samples, $\hat{t}_k$ ($k = 1, 2, \dots, \phi$).

Specifically, let $\lambda^*(t; \theta)$ be a given intensity function with unknown parameters θ . Then, we can obtain the ML estimates, $\hat{\theta}$, from the resampled data $\hat{t}_k$. Hence, we give an estimate of $\lambda(t; \theta)$ as

$$\hat{\lambda}(t; \hat{\theta}) = \lambda^*(t; \hat{\theta}) \quad (9)$$

and the estimate of our target intensity function $\lambda(t; \theta)$ by

$$\hat{\lambda}(t; \hat{\theta}) = \frac{1}{1 - \frac{1}{n}} \cdot \lambda^*(t; \hat{\theta}). \quad (10)$$

To our best knowledge, the above resampling bias correction has not been considered yet in the existing resampling-based bootstrap Cowling et al. (1996), Kaneishi and Dohi (2011). In the following sections, we investigate the above thinning-like parametric bootstrapping with bias correction through Monte Carlo simulation experiment and actual software development project data analysis. The pseudo-code for fine-grained parametric bootstrap is given by Algorithm 9.

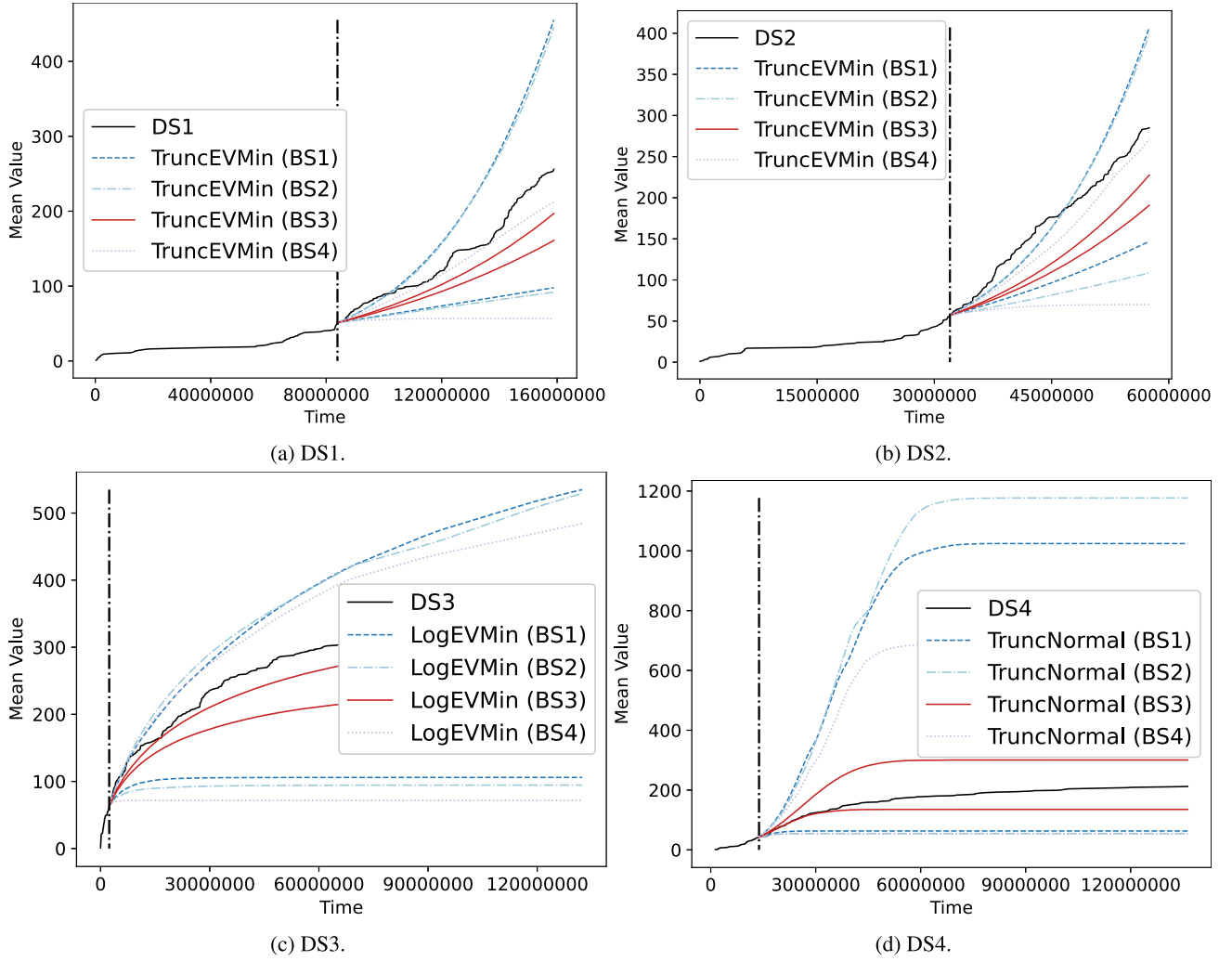


Fig. 7. Confidence intervals of mean value function with DS1 ~ DS4 (20%). .

4. Simulation experiment

4.1. Setup

We perform a Monte Carlo simulation to thoroughly investigate our fine-grained bootstrap method. We compare our method with existing methods; two resampling methods and a simulation-based method. Suppose that the *true model* is given by an exponential NHPP-based SRM [Goel and Okumoto \(1979\)](#) with given parameters $(a, b) = (150, 0.00005)$, where the failure truncation time is $t_e = 60000$. This setup implies that the total number of software faults in the program before testing follows a Poisson distribution with mean 150. It also assumes that system testing is conducted for 60,000 time units, with a fault-detection rate of 0.00005 per fault. It is assumed through the simulation experiment that the exponential NHPP-based SRM is given but the model parameters are unknown. We apply four bootstrap methods for eleven NHPP-based SRMs in [Table 1](#), where BS1 (Method 2 in [Kaneishi and Dohi \(2011\)](#)), BS2 (Method 3 in [Kaneishi and Dohi \(2011\)](#)), and BS3 (our method in this paper) are resampling-based algorithms, and BS4 (Method 1 in [Kaneishi and Dohi \(2011\)](#)) is simulation-based algorithm.

In our simulation experiment, generate the 100 baseline pseudo time series data up to truncation time $t_e = 60000$ by the exponential NHPP-based SRM as the true model. In BS1, BS2 and BS3, we further generate 100 resampled time series data from each baseline pseudo time series

data, and yield a total of $100 \times 100 = 10,000$ time series sequences for each bootstrap method. For the resulting 10,000 time series data, we apply the exponential NHPP-based SRM, and get 10,000 estimates of model parameters and their associated mean value function, intensity function and software reliability. Because these estimates are regarded as realizations of model parameters and the reliability measures for the exponential NHPP-based SRM, we construct the empirical distribution for each target estimator and derive the two-sided confidence intervals. In BS4, we first perform ML estimation of the model parameters from the data and then generate 100 time-series sequences from a single baseline dataset. As a result, we obtain $100 \times 100 = 10000$ time series sequences and compute the resulting two-sided confidence intervals for each estimator.

Note that the above bootstrap methods give the confidence intervals during the time interval $(0, t_e]$. If one is interested in prediction over the future time period $(t_e, t_u]$, we plug one of the 10,000 parameter estimates into the mean value function, intensity function, and software reliability function. We then obtain the *predictive* software reliability measures and their associated predictive two-sided confidence intervals.

4.2. Measures

For a given NHPP-based SRM, we investigate whether four bootstrap methods could provide accurate estimates of the true model. Recently,

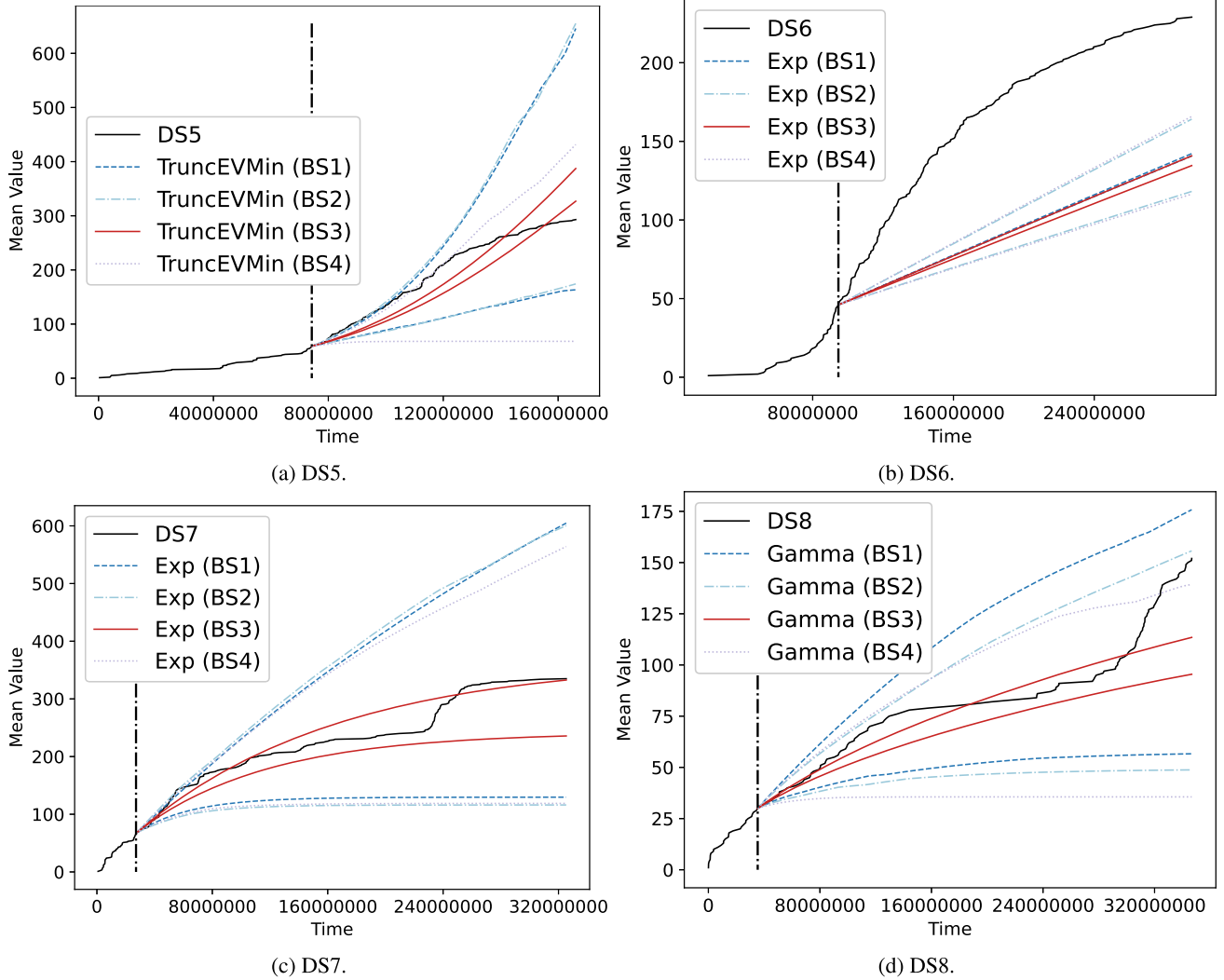


Fig. 8. Confidence intervals of mean value function with DS5 ~ DS8 (20%). .

Leskela [Leskelä \(2024\)](#) proposed the Kullback-Leibler (KL) divergence and Hellinger distance for two NHPPs which can measure the distance between two stochastic point processes. Let $\lambda_1(t; \theta_1)$ and $\lambda_2(t; \theta_2)$ be the intensity functions of two NHPPs. Then the KL divergence and Hellinger distance of NHPP are given by

$$KL(\lambda_1, \lambda_2) = \int_0^{t_e} \lambda_1(x; \theta_1) \ln \left(\frac{\lambda_1(x; \theta_1)}{\lambda_2(x; \theta_2)} \right) + \lambda_2(x; \theta_2) - \lambda_1(x; \theta_1) dx, \quad (11)$$

$$H(\lambda_1, \lambda_2) = 1 - e^{-\frac{1}{2} \int_0^{t_e} [\lambda_1(x; \theta_1) - \lambda_2(x; \theta_2)]^2 dx}, \quad (12)$$

respectively. The smaller KL divergence and Hellinger distance indicate the superior goodness-of-fit performance between the two NHPPs. In this paper, we utilize these two distances to assess the goodness-of-fit performance of bootstrap methods in our simulation experiment, where $\lambda_1(x; \theta_1)$ is the intensity function of the true model with $\theta_1 = (150, 0.00005)$, and $\lambda_2(x; \theta_2)$ denotes the intensity function of our estimated exponential NHPP-based SRM with the ML estimates for each re-sampled/simulated data. Since these distances are calculated for 10,000 time series sequences, we also get a total of 10,000 estimates of KL divergence and Hellinger distance.

4.3. Discussions

[Fig. 1](#) gives the box plot of the model parameters estimated with by three resampling-based bootstrap methods (BS1, BS2, BS3) and simulation-based bootstrap method (BS4), where the red horizontal dashed line in [Fig. 1](#) denotes the true values of model parameters in the simulation experiment. It can be seen that BS3 gave somewhat closer median values to respective true parameters with smaller variances than the other methods. More concretely, [Table 2](#) presents the median, average variance and 95% confidence interval of the ML estimates of model parameter with different bootstrap methods, where the bolded values indicate the method whose median and average of interval estimation is most closely aligned with the true value. It is evident that BS1 and BS3 could yield the most accurate median estimates for the parameters of a and b , respectively. The BS3 gives the most accurate average estimates for both parameters of a and b . We also find that BS3 demonstrated the smallest variance in the interval estimation of both model parameters a and b , and that both the interval estimates with BS1 and BS3 were more accurate, compared to the other two bootstrap methods.

Apart from the parameter estimation, we compare the distances between the true model and our estimated NHPP-based SRM in terms of the KL divergence and Hellinger distance. [Table 3](#) presents the comparison of KL divergence and Hellinger distance for each bootstrap method. It can be observed that BS3 could yield the smallest

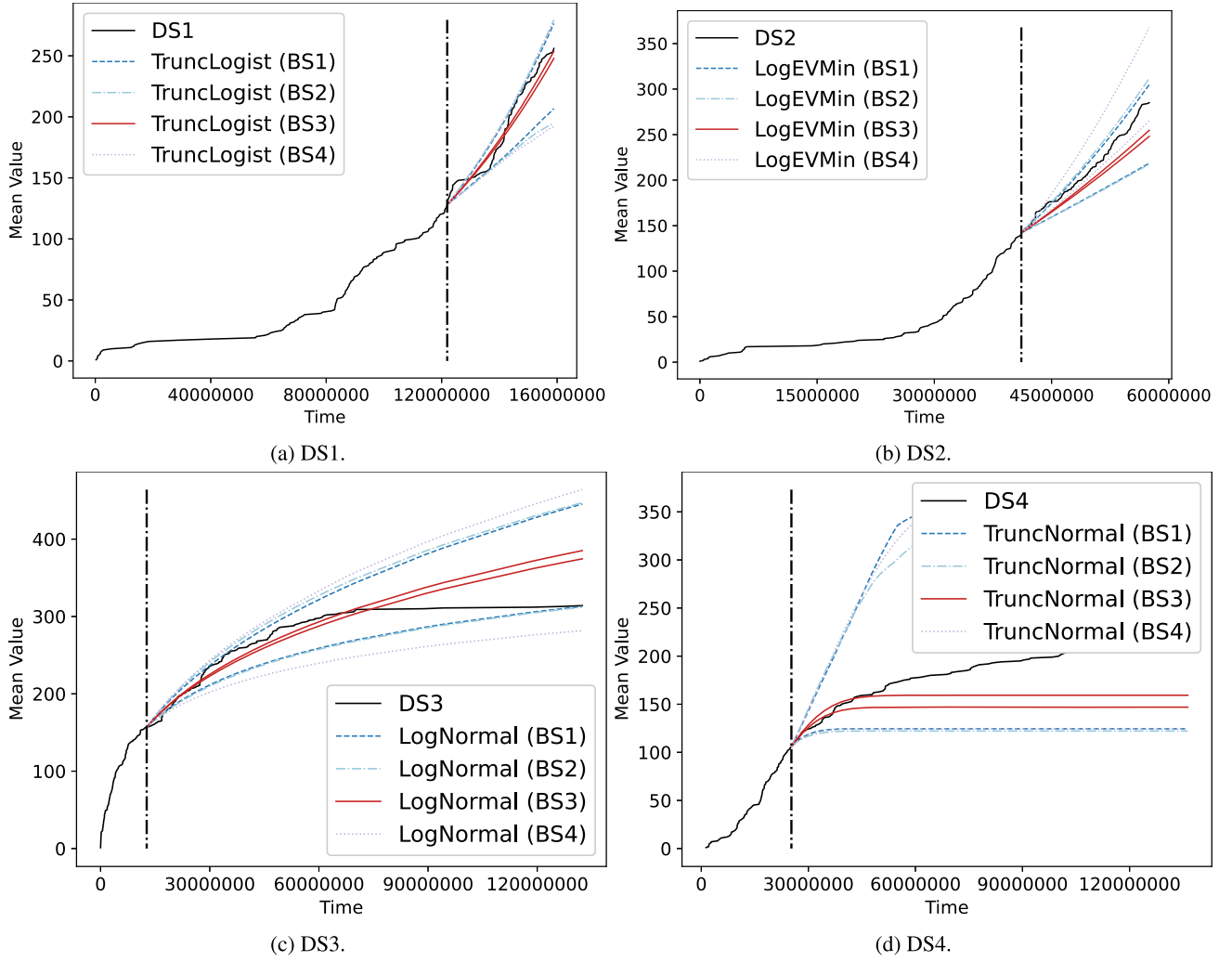


Fig. 9. Confidence intervals of mean value function with DS1 ~ DS4 (50%). .

Algorithm 3: Method 3 (Resampling-based bootstrap) Algorithms.

Data: Observed software time-domain fault-count data $t = (t_1, t_2, \dots, t_n)$, number of resampling ϕ , a given intensity function $\lambda(t; \theta)$.

Result: The intensity functions $\{\tilde{\lambda}(t; \tilde{\theta}_1), \tilde{\lambda}(t; \tilde{\theta}_2), \dots, \tilde{\lambda}(t; \tilde{\theta}_\phi)\}$

```

1  $k = 1$ ;
2 while  $k \leq \phi$  do
3   Generate a random value  $n_k \sim P(n)$ , where  $P(n)$  is the
   Poisson distribution with parameter  $n$ ;
4   Generate the resampling data  $\hat{t}_k$  by pick the sample from  $t$ 
   randomly with replacement  $n_k$  times;
5   # Obtain the ML estimates of model parameters  $\tilde{\theta}_k$  with
   simulation data  $\hat{t}_k$ ;
6    $\tilde{\theta}_k = \arg \max_{\theta} \ln \mathcal{L}(\theta; \hat{t}_k)$ ;
7   # Obtain the intensity function for bootstrap
    $\tilde{\lambda}(t; \tilde{\theta}_k) = \lambda(t; \tilde{\theta}_k)$ ;
8   # Counter increment;
9    $k = k + 1$ ;
10 end
```

Algorithm 4: Method 4 (Fine-grained parametric bootstrap) Algorithms.

Data: Observed software time-domain fault-count data $t = (t_1, t_2, \dots, t_n)$, number of resampling ϕ , a given intensity function $\lambda(t; \theta)$.

Result: The intensity functions $\{\tilde{\lambda}(t; \tilde{\theta}_1), \tilde{\lambda}(t; \tilde{\theta}_2), \dots, \tilde{\lambda}(t; \tilde{\theta}_\phi)\}$

```

1  $k = 1$ ;
2 while  $k \leq \phi$  do
3   Remove each time point  $t_i$  in  $t$  with same probability  $1/n$ ,
   then form the resampling data  $\hat{t}_k$  by the remaining data
   points;
4   # Obtain the ML estimates of model parameters  $\tilde{\theta}_k$  with
   simulation data  $\hat{t}_k$ ;
5    $\tilde{\theta}_k = \arg \max_{\theta} \ln \mathcal{L}(\theta; \hat{t}_k)$ ;
6   # Obtain the intensity function for bootstrap
    $\tilde{\lambda}(t; \tilde{\theta}_k) = \frac{1}{1 - \frac{1}{n}} \lambda(t; \tilde{\theta}_k)$ ;
7   # Counter increment;
8    $k = k + 1$ ;
9 end
```

median, average and variance for both KL divergence and Hellinger distance. Fig. 2 illustrates the box plots of the mean value function

and intensity function at three different testing time points. It can be found that the medians for each bootstrap method are closely

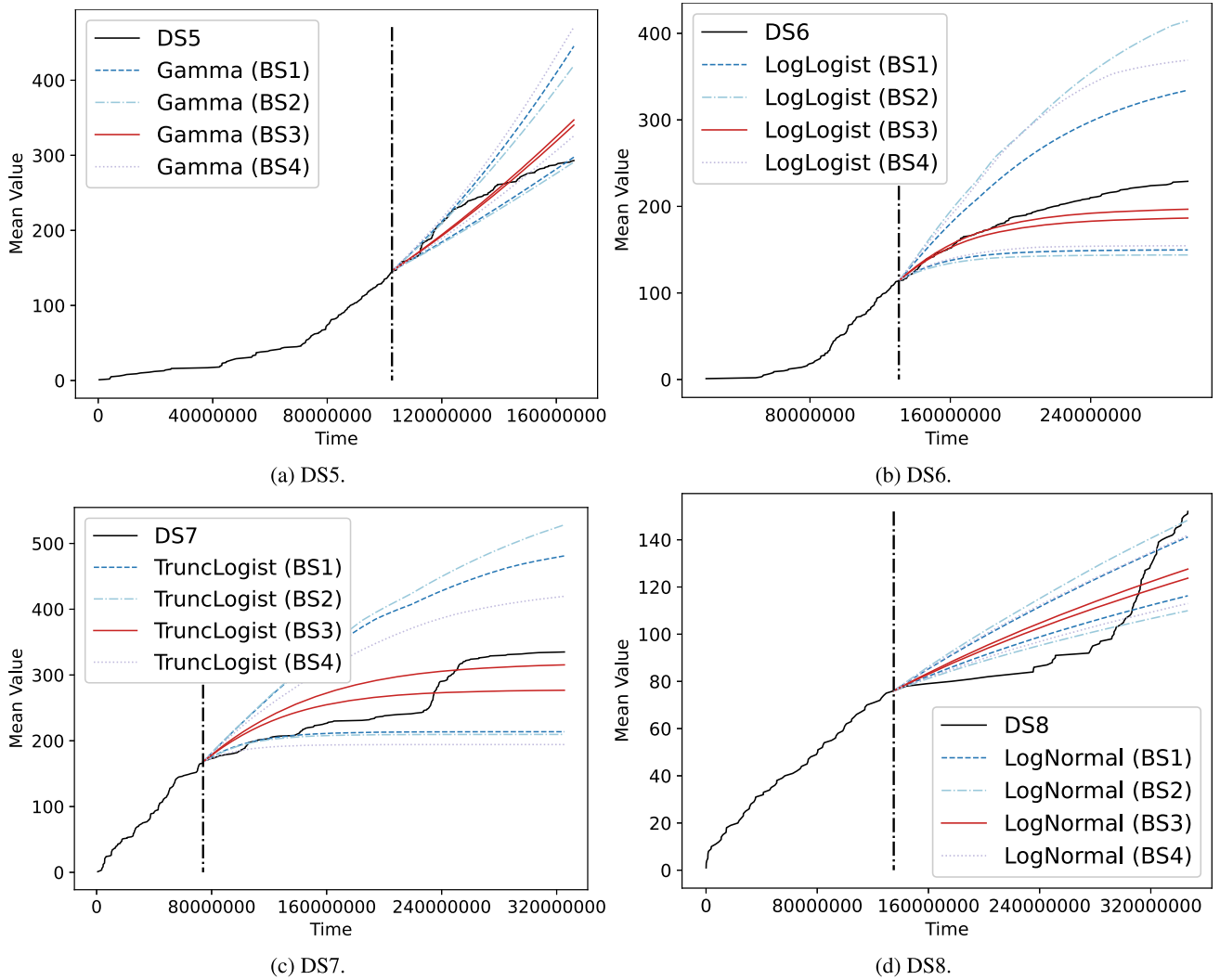


Fig. 10. Confidence intervals of mean value function with DS5 ~ DS8 (50%).

aligned, but the variance of BS3 is significantly smaller than that of other bootstrap methods. Hence, our thinning-like bootstrap method could offer more accurate and robust interval estimates of the mean value function and intensity function for an exponential NHPP-based SRM.

Next, we concern the predictive performances of four bootstrap methods. Fig. 3a shows the predictive two-sided confidence intervals with 95% significance level of the mean value function in time interval $[60000, 100000]$ in simulation experiment. It is shown that all of the bootstrap methods (BS1 ~ BS4) could provide the predictive intervals which contain the true mean value function. Especially, it is noted that BS3 could always make narrower predictive intervals which also contain the true mean value function, and that the upper limit of predictive intervals with 95% significance level provided by BS3 was closer to the true mean value function than the existing three bootstrap methods in $[60000, 100000]$. This fact enables us to conclude that our fine-grained bootstrap method can generate the more appropriate resampled data which lead to the ML estimation of model parameters more exactly. In addition, the predictive intervals by the other bootstrap methods were very close from each other. Fig. 3b plots the predictive confidence intervals of the software intensity function in the exponential NHPP-based SRM. Similar to the results in Fig. 3a, we remark that BS3 provided the more accurate predictive confidence intervals than BS1 and BS2.

Consider the predictive intervals of software reliability function. Fig. 4 illustrates the predictive intervals obtained by different bootstrap methods, where all bootstrap methods gave the predictive confidence regions which contain the true software reliability function. It is worth to note that regardless of the kind of bootstrap methods, the lower limit of software reliability function tended to converge to a certain value close to the true software reliability value. In contrast with the existing methods, our approach, BS3, gave the predictive confidence intervals of software reliability function with less variance. Through the Monte Carlo simulation experiment, we can conclude that our fine-grained bootstrap method could outperform the existing bootstrap methods in almost all cases, and could provide the stable interval predictions with more accurate median and less variance.

4.4. Threats to the validity of the simulation experiments

The degree of the validity of the experimental design is as important as the experimental results. In this paper, we are concerned to the internal, external, and construct validity of the experiment, respectively.

Internal validity: Internal validity describes whether there are uncontrolled factors in the experiment that we have not considered. In simulation experiments, we select the exponential NHPP-based software reliability model as the true model where the exponential model

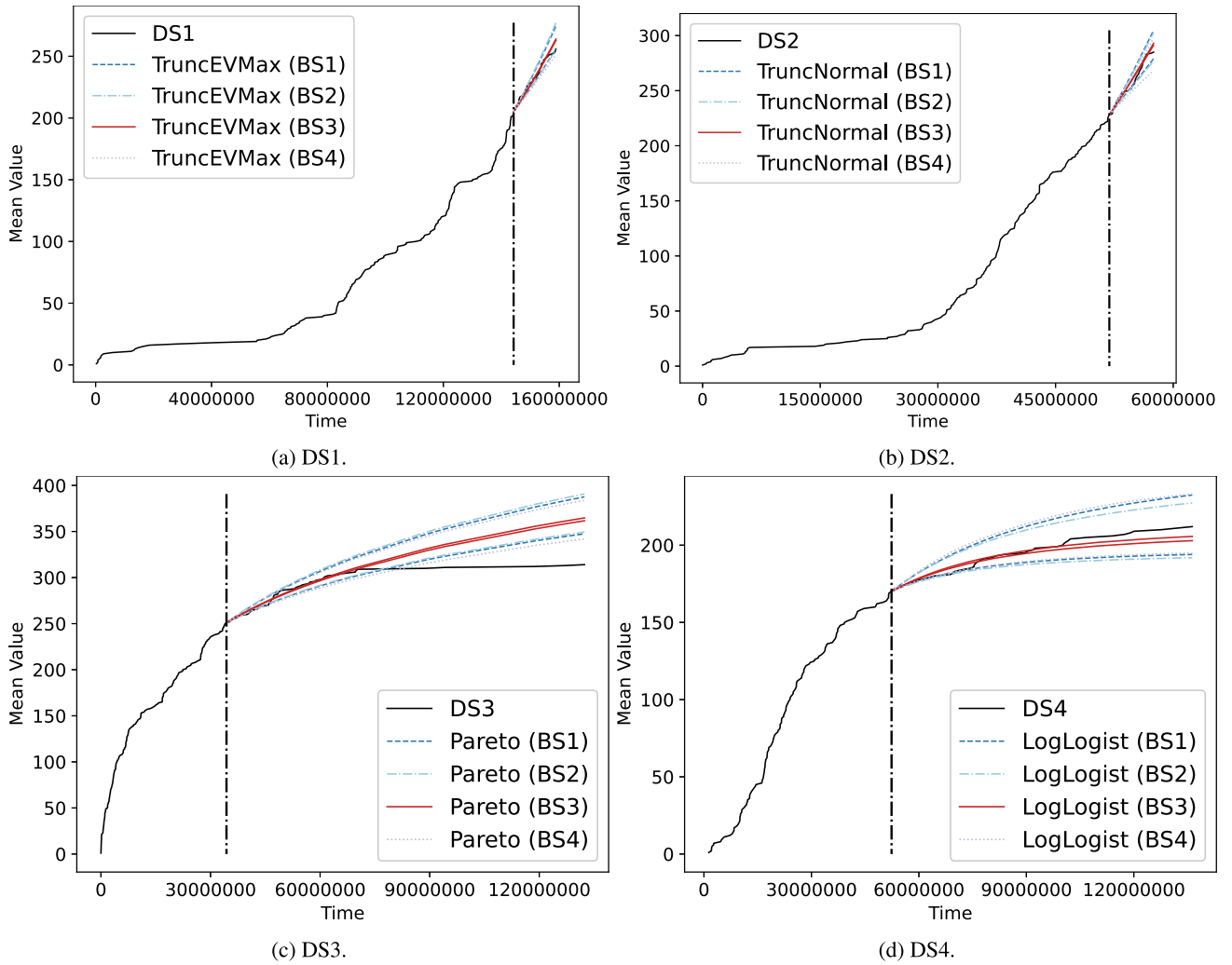


Fig. 11. Confidence intervals of mean value function with DS1 ~ DS4 (80%). .

is the most famous model in software reliability. The model parameters are estimated by maximum likelihood estimation. Obviously, the choice of maximum algorithm is also an important threat to internal validity. In our experiments, we used the EM algorithm which has been implemented in the SRATS tool and is widely used in the field of software reliability.

External validity: External validity discusses whether the same experimental results can be generalized to other situations or settings. The exponential NHPP-based software reliability model as the most famous and widely used model in software reliability can improve the external validity of our experiments.

Construct validity: Construct validity is the degree to which a test measures what it claims, or purports, to be measuring. To verify that our proposed method enhanced the interval estimation performance, we performed 10,000 simulation experiments for 4 bootstrap methods. It means that we verified the validity of our methods in total $40000 = 10,000 \times 4$ times in simulation experiments.

5. Real data analysis

5.1. Setup

As a more realistic scenario, consider the case where both the true model and its parameters are unknown. Here we analyze actual software fault-count time-domain data in eight open-source software (OSS)

development projects, collected from GitHub in Table 4. As the baseline NHPP-based SRMs, we assume eleven NHPP-based SRMs in Table 1. Similar to the simulation experiment, we generate 100 resampled/simulated bootstrap data at four observation points; the first 20%, 50%, 80% and 100% of the entire data, and apply four bootstrap methods for each data set. For instance, when the data set consists of 100 time-domain data, $t = \{t_1, t_2, \dots, t_{100}\}$, then the observation points are t_{20} , t_{50} , t_{80} and t_{100} (whole data). Applying eleven NHPP-based SRMs in Table 1, we carry out the ML estimation for each model and select the best goodness-of-fit model based on the Akaike information criterion (AIC);

$$AIC = -2\ln\mathcal{L}(\hat{\theta}; t) + 2\phi, \quad (13)$$

where $\hat{\theta}$ is the ML point estimate in Eq. (7) and ϕ is the number of free parameters in an NHPP. The smaller AIC denotes the better goodness-of-fit model among eleven NHPP-based SRMs. In the above example, we generate 100 bootstrap data with length $n = 20, 50, 80$ or 100 for each bootstrap method, and estimate 100 sets of model parameters by means of ML estimation. Once the ML point estimates of model parameters are given, we predict the reliability measures by substituting the ML estimates into mean value function, intensity function and software reliability function for each bootstrap data. After getting an empirical distribution of each software reliability measure, we seek the two-sided confidence intervals of each reliability measure.

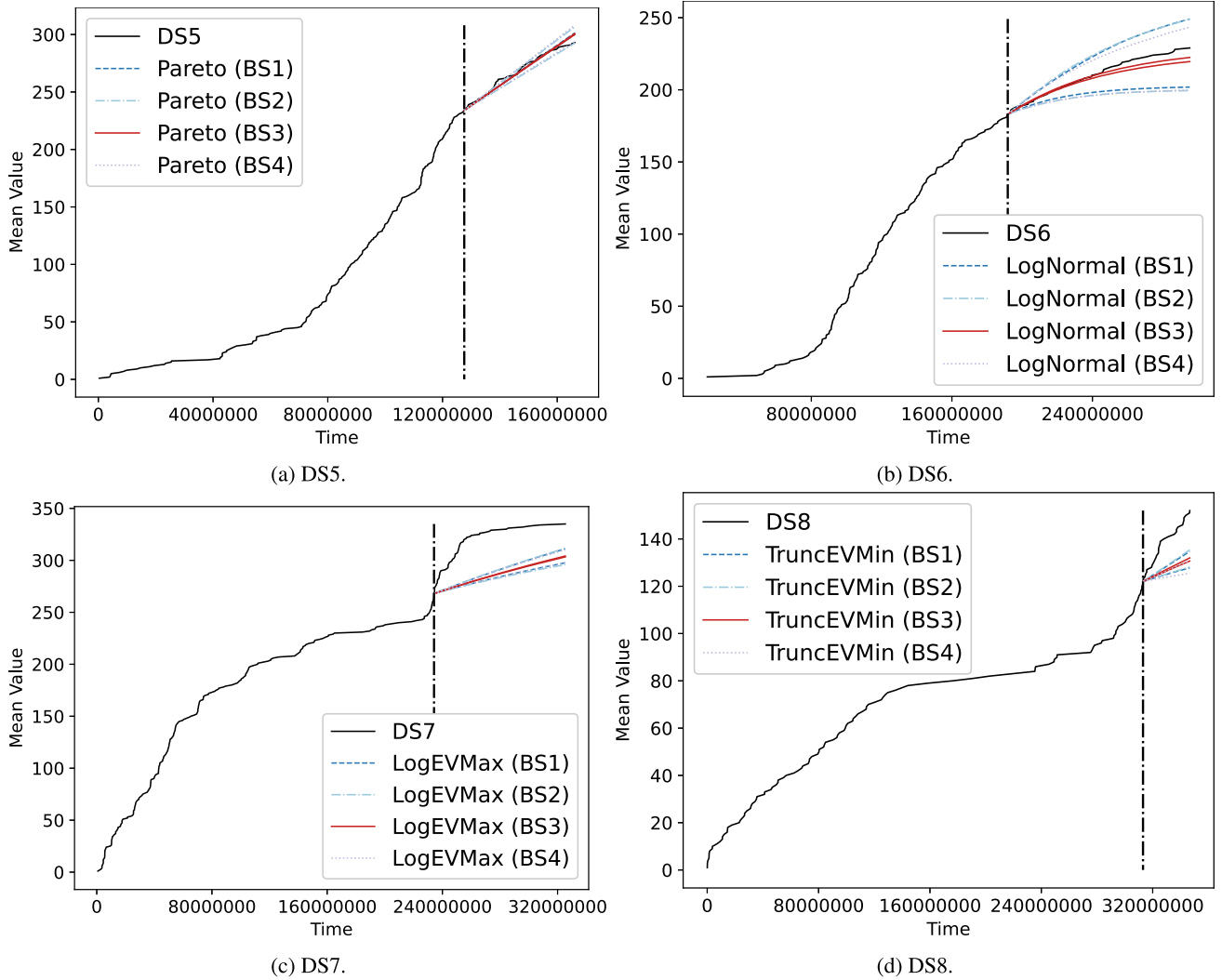


Fig. 12. Confidence intervals of mean value function with DS5 ~ DS8 (80%).

5.2. Interval estimation accuracy in each bootstrap approach

Here, we examine the interval estimation accuracy of the four bootstrap methods in actual software faults data. First, we consider a fitting problem of mean value function. We try to find the best fitting model in the sense of AIC, among 11 candidate NHPP-based SRMs in eight datasets with the 100% entire data. Then, we estimate the model parameters of the best fitting model with 100 bootstrap data for one of four bootstrap approaches, and calculate the mean value function in the time interval $[0, t_n]$, where t_n is the testing termination time. Figs. 5a ~ 6d illustrate the cumulative number of software faults and the mean value functions with different bootstrap methods.

In Figs. 5a ~ 6d, the solid lines are the median of interval estimation of the best fitting model for each bootstrap method, and the box plots present the empirical distributions with different bootstrap methods at different time points. It can be seen that the medians in different bootstrap methods are quite similar from each other even in different datasets, so that four bootstrap methods have similar asymptotic properties in nature. On one hand, we find that their variances are quite different. From the box plot in Figs. 5a ~ 6d, it can be observed that our bootstrap method (BS3) always yields a lower variance in all datasets at all observation points. This is because our refined bootstrap method provides more accurate interval estimates

when the median of the interval estimate fits the original data very well, while the existing three bootstrap methods can give the confidence intervals that cover the data even in the presence of high noise levels.

Next, we consider the predictive interval with different bootstrap methods in actual software faults data. We first select the best goodness-of-fit model with smaller AIC from 11 candidate SRMs at 20%, 50%, and 80% of entire data, where the remaining 80%, 50%, and 20% of the data are used for validation by calculating the PMAE. In deriving the predictive confidence intervals, we obtain the predictive empirical distribution of mean value function for the remaining time period at 20%, 50% and 80% of entire data by generating 100 bootstrap samples (pseudo random numbers) data with different bootstrap methods.

For better demonstration, we give the predictive confidence intervals of the mean value function at 20%, 50%, and 80% observation points for different data sets in Figs. 7a ~ 8d, Figs. 9a ~ 10d and Figs. 11a ~ 12d, where the dashed lines in the figures denote the 95% confidence intervals of prediction values of mean value function with different bootstrap methods. In most of cases, the existing resampling-based and simulation-based bootstrap methods provide a wider predictive interval because those methods have a larger variance. This property indeed leads to the fact that the existing bootstrap methods tend to outperform our refined bootstrap method (BS3) from the viewpoint of data coverage. However,

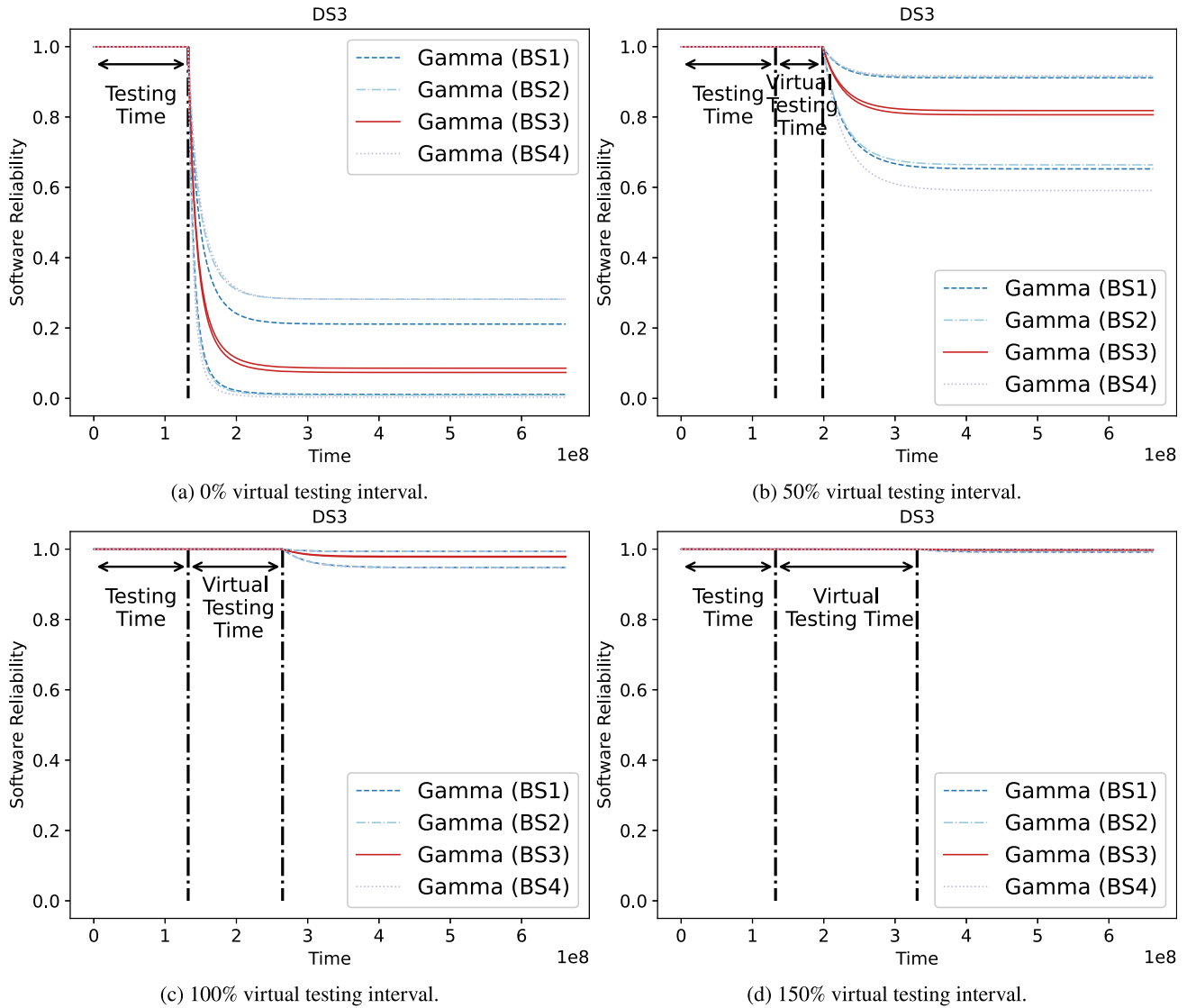


Fig. 13. Predictive two-sided confidence intervals of software reliability without virtual testing..

in some of cases such as Fig. 8c and Fig. 7d, we verify that our bootstrap method gives the predictive confidence interval which can cover the future data with much less variance.

At the first look, the results in real-data analyses and simulation experiments are contradiction. However, in our simulation experiments we assumed that the true model form is known, but the model parameters are unknown in an ideal noise-free situation where the number of data is sufficiently large. In contrast, the actual data analyses assumed not only an unknown true model but also a limited number of data contaminated by substantial noise. Consequently, the existing methods which exhibit larger variances, are less sensitive to the noise and thus achieve higher coverage on actual software fault-count data. By comparison, our BS3 method with smaller variance can make it more vulnerable to the noise inference; so our refined bootstrap method results the interval estimators with lower variances that are closer to the true values, yielding the superior performances.

It is worth noting that while the existing methods can achieve higher coverage with larger variances in prediction, such tendencies are caused by the substantially greater uncertainty, compared to our refined method. In other words, the higher coverage of prediction value may lead to a very optimistic prediction of the cumulative number of software faults detected in future. As confirmed in simulation experiments,

our refined bootstrap method, BS3, simultaneously attains higher estimation accuracy in the sense of the proximity of median of predictive confidence interval to the true value and lower uncertainty. Hence, by adjusting the significant level the coverage in prediction can be easily improved in our method.

5.3. A Case Study for software reliability prediction

Finally, we give a case study for the predictive confidence intervals of software reliability in real-data analyses. Dissimilar to the simulation experiments, since the real software reliability function is unknown, we concern the behavior of software reliability with/without *virtual testing time*. Zhao et al. (2018) introduced the so-called virtual testing time to provide a plausible software reliability assessment under the hypothesis that no software fault will be detected during the virtual testing time period. More specifically, when no software fault are found for a fixed future time period, such statistical data affect the parameter estimation and the resulting software reliability can be also improved. This hypothesized time period without bug detection is called the virtual testing time. Assessing software reliability for the virtual testing time is meaningful because we can make a decision to continue software test for a given virtual testing time period. Suppose that t_u is the software release

Table 6Minimum expected costs for optimal software release times with varying c_2 .

c_1	c_2	c_3	Theoretical Results			
10	1	5	152			
10	2	5	302			
10	3	5	452			
10	4	5	602			
c_1	c_2	c_3	BS1			
			Median	Average	Variance	95% Confidence Interval
10	1	5	154	155	138	[137,175]
10	2	5	304	307	547	[271,347]
10	3	5	455	459	1227	[406,519]
10	4	5	606	611	2178	[540,691]
c_1	c_2	c_3	BS2			
			Median	Average	Variance	95% Confidence Interval
10	1	5	154	155	292	[128,184]
10	2	5	306	307	1163	[253,365]
10	3	5	458	459	2611	[378,546]
10	4	5	609	611	4638	[503,727]
c_1	c_2	c_3	BS3			
			Median	Average	Variance	95% Confidence Interval
10	1	5	151	153	132	[135,173]
10	2	5	299	303	524	[267,343]
10	3	5	448	453	1178	[400,514]
10	4	5	596	603	2092	[532,684]
c_1	c_2	c_3	BS4			
			Median	Average	Variance	95% Confidence Interval
10	1	5	155	155	287	[129,184]
10	2	5	307	308	1143	[255,366]
10	3	5	459	461	2568	[381,547]
10	4	5	612	614	4561	[507,729]

time, t_n is the testing termination time, and $t_s > t_u$ is the future time. Under the hypothesis that no software fault are found during $(t_u, t_s] = (t_n, t_{n+k}]$ in Eq.(4), the ML estimates are obtained by adding zero-count data at $t_{n+1}, t_{n+2}, \dots, t_{n+k}$, and the software reliability can be predicted for $t > t_s = t_{n+k}$. Even though software faults are found for $(t_u, t_s]$, we need to re-estimate the model parameters with the updated fault count data. The subsequent virtual testing time is reset at the fault detection point.

For brevity we discuss the software reliability prediction for only DS3 for aiming to show the predictive interval for software reliability. Fig. 13a illustrates the behaviors of software reliability in the time interval $(t_n, 5 \times t_n]$ longer, where t_n is the testing termination time in the original data set and "Gamma" denotes the best fitting NHPP-based SRM for the 100% data. Similar to the previous real-data analyses, the dashed lines in Fig. 13a implies the 95% confidence interval of software reliability with different bootstrap methods. It can be seen that the software reliability function drops down to the zero level immediately, even for a relatively short operational time interval. Such a reliability measurement is of course meaningless in the software release decision. However, this observation is statistically convinced because any software fault could be found even at the observation point, say t_n . In another word, the software reliability assessment without virtual testing time is almost meaningless.

Figs. 13b ~ 13c show the behaviors of 95% predictive confidence intervals of software reliability function with 50%, 100% and 150% vir-

tual testing time for different bootstrap methods, respectively. It can be observed that the lower bounds of the software reliability predictive confidence intervals with 50% 100% and 150% virtual testing times converge to $0.6 \sim 0.75$, $0.93 \sim 0.96$ and 0.99 , respectively. It is straightforward to understand that a sufficiently long virtual testing time without any detected software fault leads to much higher software reliability level. This fact shows that experiences on no fault detection by testers results in reliable software. From the above experiments, we suggest that the acceptance test by user/customer should be performed during longer time period than the system testing time in the development side, if the high level software reliability is required in a software development project.

5.4. Threats to the validity of the real data analysis

Next, we discuss about the threats to the validity of the real data analysis.

Internal validity: Collecting representative software failure data is an important threat to internal validity. Considering the recent development of open-source software, We used 8 sets of software fault testing data from well-known projects which cover different application domains and different programming languages on GitHub.

Furthermore, the representative software reliability models we selected in numerical experiments also constitute an aspect of the threat to internal validity. In our real data analysis, we applied 11 of the most representative SRGMs in the field of software reliability and fairly tested their interval estimation performance. The model parameters are estimated by maximum likelihood estimation. Obviously, the choice of maximum algorithm is also an important threat to internal validity. In our real data analysis, the EM algorithm was used to the 11 SRGM which has been implemented in the SRATS tool.

External validity: We used the software fault data from well-known projects on GitHub to improve the external validity of our experiments. The size of the dataset may also affect the experimental results. To address this issue, we applied 8 sets of software failure data of varying sizes.

Construct validity: To verify that our proposed method enhanced the interval estimation performance in real data analysis, we performed 100 bootstrap experiments for 8 data set and 4 observation points with 4 bootstrap methods. We verified the validity of our methods in total $12800 = 100 \times 8 \times 4 \times 4$ times real data analysis.

6. Optimal software release decision

In this section, we consider an optimal software release problem. Determination of software release timing is an important issue in software development. In the past literature, a so-called software cost model Dalal and McIntosh (1994), Koch and Kubat (1983), Okumoto and Goel (1979), Yamada and Osaki (1985a), Yang et al. (2008) has been considered for determining an optimal software release time by minimizing software testing cost. In this section, we obtain the interval estimation of the optimal release time and its associated optimal testing cost with four bootstrap methods, and compare with those theoretical values which are calculated by using the real model parameters. Let c_1 , c_2 , and c_3 be the software testing cost per unit of time, debugging cost to remove a software fault in the testing phase, and debugging cost to remove a software fault in the operational phase, respectively. The system test starts at time $t_0 = 0$ and terminates at time t_e , where t_{CL} implies the termination of software life cycle. Let $\Lambda(t) = E[N(t)]$ be the expected cumulative number of software faults detected by time t . Then, the expected software testing cost over the entire software life cycle is given by

$$C(t_e) = c_1 t_e + c_2 \Lambda(t_e) + c_3 [\Lambda(t_{LC}) - \Lambda(t_e)], \quad (14)$$

where without any loss of generality $c_2 < c_3$ and $t_e < t_{CL}$. The optimal software release time t_e^* is determined by minimizing the

expected software testing cost $C(t_e)$, i.e.,

$$t_e^* = \arg \min_{t_e} C(t_e). \quad (15)$$

It is easily understood that the optimization problem in Eq. (15) is well defined, because the release time t_e must take account of the balance between the testing cost and the debugging cost. If the release time t_e tends to be shorter, then a number of software faults will be detected after the system is released and a large debugging cost will occur because of $c_2 < c_3$. On the other hand, if the release time t_e becomes large enough, the testing cost $c_1 \times t_e$ in testing phase will be also increase very fast. In our simulation experiments, the cost parameters, c_1 , c_2 and c_3 , are cited in the seminal contribution by Okumoto and Goal [Okumoto and Goel \(1979\)](#); $c_1 = 10$ (USD per week) = $1.6534e - 05$ (USD per sec), $c_3 = 5$ (USD per fault). The software life cycle is given as $t_{CL} = 2.6$ (weeks) = 1581120 (sec). We seek the optimal release time t_e^* and its corresponding minimum expected software testing cost $C(t_e^*)$ with varying $c_2 = 1, 2, 3, 4$ (USD per fault).

[Table 5](#) and [Table 6](#) present the estimation intervals of the optimal software release time and its associated expected software testing cost with varying c_2 under different bootstrap methods, where "Real Results" denotes the optimal release time and the minimum expected software testing cost with the real model parameters in our simulation experiments. It is clear to see that the optimal release time t_e^* becomes earlier and the expected software testing cost $C(t_e^*)$ becomes larger as the cost parameter c_2 increases. In contrast with the real values, our bootstrap method (BS3) can always provide the closer median and average and more narrow estimation intervals for the optimal release time and the minimum expected software testing cost than the other bootstrap methods.

7. Conclusions

In this paper, we have proposed a novel bootstrap method for NHPP-based SRMs with a thinning-like approach. In contrast to the existing three bootstrap methods, our refined bootstrap method has shown several advantages. First, our bootstrap method did not contradict in theory the underlying NHPP assumption of software fault-detection process, while the existing two resampling-based methods did not consider carefully the non-stationarity of underlying NHPP. We examined the interval estimation accuracy of our refined bootstrap method in the numerical experiments with 11 parametric NHPP-based SRMs [Abdel-Ghaly et al. \(1986\)](#), [Achcar et al. \(1998\)](#), [Goel \(1985\)](#), [Gokhale and Trivedi \(1998\)](#), [Ohba \(1984\)](#), [Ohishi et al. \(2009\)](#), [Yamada et al. \(1983\)](#), [Zhao and Xie \(1996\)](#). In simulation experiments, we have confirmed that our refined bootstrap method could provide the more accurate interval estimation and lower variance than the existing three bootstrap methods in both goodness-of-fit and prediction. In real-data analyses, however, it has been observed that our refined method failed to cover the actual software fault data in the predictive confidence intervals, because it has been caused by randomness of the underlying data and the lack of statistical information in single time series data. Hence, in the case where the real model is not known, our bootstrap method proposed in this paper tends to give pessimistic prediction with narrow confidence regions. In other words, our results positively show that the realistic prediction can be made by adjusting the significance level when we apply the refined bootstrap method.

We have also applied four bootstrap methods to estimate software reliability measures including software intensity function and software reliability function by means of the interval estimation. We also consider the interval prediction for the cost optimal software release time. To our best knowledge, the bootstrap-based approach for the interval estimation/prediction in software reliability has argued in a very few papers. Our contribution in this paper is the latest research result by improving the bootstrap method if software reliability prediction.

In the future article, we will consider non-parametric NHPP-based SRMs under incomplete knowledge of SRM formulation and apply our

refined bootstrap method, because we just concerned the bootstrap method for only parametric NHPP-based SRMs with known parametric mean value functions. This will be a challenging issue because the non-parametric model does not enable us to make the prediction for future, especially, in the long-run.

CRedit authorship contribution statement

Jingchi Wu: Writing - original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Tadashi Dohi:** Writing - review & editing, Supervision, Funding acquisition, Conceptualization; **Junjun Zheng:** Writing - review & editing; **Hiroyuki Okamura:** Writing - review & editing.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- AaA., 2011. <https://github.com/twbs/bootstrap>.
- AaA., 2012a. <https://github.com/hexojs/hexo>.
- AaA., 2012b. <https://github.com/httpie/cli>.
- AaA., 2013. <https://github.com/BVLC/caffe>.
- AaA., 2018a. <https://github.com/tabler/tabler>.
- AaA., 2018b. <https://github.com/notable/notable>.
- AaA., 2018c. <https://github.com/MonitorControl/MonitorControl>.
- AaA., 2021. <https://github.com/withastro/astro>.
- Abdel-Ghaly, A.A., Chan, P.Y., Littlewood, B., 1986. Evaluation of competing software reliability predictions. *IEEE Trans. Softw. Eng.* 12 (9), 950–967.
- Achcar, J.A., Dey, D.K., Nivethi, M., 1998. A bayesian approach using nonhomogeneous poisson processes for software reliability models. *World Scientific, Singapore*, pp. 1–18.
- Cowling, A., Hall, P., Phillips, M.J., 1996. Bootstrap confidence regions for the intensity of a poisson point process. *J. Am. Stat. Assoc.* 91 (436), 1516–1524.
- Dalal, S.R., McIntosh, A.A., 1994. When to stop testing for large software systems with changing code. *IEEE Trans. Softw. Eng.* 20 (4), 318–323.
- Efron, B., 1979. Bootstrap methods: another look at the jackknife. *Ann. Stat.* 7 (1), 1–26.
- Efron, B., 1981. Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods 68, 589–599.
- Efron, B., Tibshirani, R.J., 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.
- Goel, A.L., 1985. Software reliability models: assumptions, limitations and applicability. *IEEE Trans. Softw. Eng.* 11 (12), 1411–1423.
- Goel, A.L., Okumoto, K., 1979. Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans. Reliab.* 28 (3), 206–211.
- Gokhale, S.S., Trivedi, K.S., 1998. Log-logistic software reliability growth model. In: *Proceedings of the 3Rd IEEE International High-Assurance Systems Engineering Symposium (HASE-1998)*. IEEE CPS, pp. 34–41.
- Hossain, S.A., Dahiya, R.C., 1992. Estimation of parameters of yamada et al. model and modification of the model. *Communications in Statistics? Theory Methods* 21 (10), 2881–2898.
- Hossain, S.A., Dahiya, R.C., 1993. Estimating the parameters of a non-homogeneous poisson-process model for software reliability. *IEEE Trans. Reliab.* 42 (4), 604–612.
- Jeske, D.R., Pham, H., 2001. On the maximum likelihood estimates for the goel-okumoto software reliability model. *Am. Stat.* 55, 219–222.
- Joe, H., 1989. Statistical inference for general-order statistics and nonhomogeneous-poisson process software reliability model. *IEEE Trans. Softw. Eng.* 15 (2), 1485–1490.
- Kaneishi, T., Dohi, T., 2011. Parametric bootstrapping for assessing software reliability measures. In: *Proceedings of the IEEE 17th Pacific Rim International Symposium on Dependable Computing (PRDC-2011)*. IEEE CPS, pp. 1–9.
- Kaneishi, T., Dohi, T., 2013. Software reliability modeling and evaluation under incomplete knowledge on fault distribution. In: *Proceedings of the IEEE 7th International Conference on Software Security and Reliability (SERE-2013)*. IEEE CPS, pp. 3–12.
- Kimura, M., Fujiwara, T., 2011. A note on software reliability monitoring by moving-average method based on a bootstrap scheme. *Int. J. Syst. Assurance Eng. Manag.* 1 (4), 293–298.
- Knafl, G., Morgan, J., 1996. Solving ml equations for 2-parameter poisson-process model for ungrouped software failure data. *IEEE Trans. Reliab.* 45 (1), 42–53.
- Koch, H.S., Kubat, P., 1983. Optimal release time for computer software. *IEEE Trans. Softw. Eng.* 9 (3), 323–327.

- Kuo, L., Yang, T.Y., 1996. Bayesian computation for nonhomogeneous poisson processes in software reliability. *J. Am. Stat. Assoc.* 91, 763–773.
- Lei, S., Smith, M.R., 2002. Evaluation of several Efron nonparametric bootstrap methods to estimate error measures for software metrics. In: *Proceedings of the 2002 IEEE Canadian Conference on Electrical and Computer Engineering*. IEEE CPS, pp. 703–707.
- Lei, S., Smith, M.R., 2003. Evaluation of several nonparametric bootstrap methods to estimate confidence intervals for software metrics. *IEEE Trans. Softw. Eng.* 29 (11), 996–1004.
- Leskelä, L., 2024. Information divergences and likelihood ratios of poisson processes and point patterns. *IEEE Trans. Inf. Theory* 70 (12), 9084–9101.
- Lewis, P. A.W., Shedler, G.S., 1979. Simulation of nonhomogeneous poisson processes by thinning. *Naval Res. Logist. Q.* 26 (3), 403–413.
- Lyu, M.R., 1996. *Handbook of Software Reliability Engineering*. New York, McGraw-Hill.
- Musa, J.D., Iannino, A., Okumoto, K., 1987. *Software Reliability: Measurement, Prediction, Application*. McGraw-Hill, New York.
- Musa, J.D., Okumoto, K., 1984. A logarithmic poisson execution time model for software reliability measurement. In: *Proceedings of the 7Th International Conference on Software Engineering (ICSE-1984)*. ACM, pp. 230–238.
- Nayak, T.K., Bose, S., Kundu, S., 2008. On inconsistency of estimators of parameters of non-homogeneous poisson process models for software reliability. *Stat. Probab. Lett.* 78, 2217–2221.
- Ohba, M., 1984. Inflection S-shaped software reliability growth model, *Stochastic Models in Reliability Theory*. Springer, Berlin/Heidelberg.
- Ohishi, K., Okamura, H., Dohi, T., 2009. Gompertz software reliability model: estimation algorithm and empirical validation. *J. Syst. Softw.* 82 (3), 535–543.
- Okamura, H., Dohi, T., 2013a. Application of EM algorithm to NHPP-based software reliability assessment? *Stochastic Reliability and Maintenance Modeling*. London, Springer.
- Okamura, H., Dohi, T., 2013b. SRATS: software reliability assessment tool on spreadsheet, *Proceedings of the 24th IEEE International Symposium on Software Reliability Engineering (ISSRE-2013)*. IEEE CPS.
- Okamura, H., Dohi, T., 2021. Application of EM algorithm to NHPP-based software reliability assessment with generalized failure count data. *Mathematics* 9 (9), 985. <https://doi.org/10.3390/math9090985>
- Okamura, H., Dohi, T., Osaki, S., 2013. Software reliability growth models with normal failure time distributions. *Reliab. Eng. Syst. Saf.* 116, 135–141.
- Okamura, H., Grottko, M., Dohi, T., Trivedi, K.S., 2007. Variational bayesian approach for interval estimation of nhpp-based software reliability models. In: *Proceedings of the 37Th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-2007)*. IEEE CPS, pp. 698–707.
- Okamura, H., Watanabe, T., Dohi, T., 2003. An iterative scheme for maximum likelihood estimation in software reliability modeling. In: *Proceedings of the 14Th IEEE International Symposium on Software Reliability Engineering (ISSRE-2003)*. IEEE CPS, pp. 479–490.
- Okumoto, K., Goel, A.L., 1979. Optimum release time for software systems based on reliability and cost criteria. *J. Syst. Softw.* 1, 315–318.
- Pham, H., 2006. *System Software Reliability*. London, Springer-Verlag.
- Pul, M. C.V., 1992. Asymptotic properties of a class of statistical models in software reliability. *Scand. J. Stat.* 19, 235–253.
- Xiao, X., Dohi, T., 2014. Interval estimation method for decision making in wavelet-based software reliability assessment. *IEICE Trans. Inf. Syst.* 97 (5), 1058–1068.
- Xiao, X., Dohi, T., 2021. Interval estimation for non-parametric nhpp-based software reliability model via simulation-based bootstrap. In: *Proceedings of the 8Th International Conference on Dependable Systems and Their Applications (DSA 2021)*. IEEE CPS, pp. 265–270.
- Yamada, S., Ohba, M., Osaki, S., 1983. S-Shaped reliability growth modeling for software error detection. *IEEE Trans. Reliab.* 32 (5), 475–484.
- Yamada, Y., Osaki, S., 1985a. Cost-reliability optimal release policies for software systems. *IEEE Trans. Reliab.* 34 (5), 422–424.
- Yamada, Y., Osaki, S., 1985b. Software reliability growth modeling: models and applications. *IEEE Trans. Softw. Eng.* 11 (12), 1431–1437.
- Yang, B., Hu, H., Jia, L., 2008. A study of uncertainty in software cost and its impact on optimal software release time 34, 813–825.
- Yano, H., Dohi, T., Okamura, H., 2024. Performance comparison of software reliability estimation algorithms. *IEEE Comput.* 57 (4), 26–36.
- Yin, L., Trivedi, K.S., 1999. Confidence interval estimation of nhpp-based software reliability models. In: *Proceedings of the 10Th International Symposium on Software Reliability Engineering (ISSRE-1999)*. IEEE CSP, pp. 6–11.
- Zhao, M., Xie, M., 1996. On maximum likelihood estimation for a general non-homogeneous poisson process. *Scand. J. Stat.* 23 (4), 597–607.
- Zhao, Y., Dohi, T., Okamura, H., 2018. Software Test-Run Reliability Modeling with Non-homogeneous Binomial Processes, *Proceedings of the 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*.