Available online at www.sciencedirect.com
SCIENCE DIRECT
Reliability Engineering and System Safety 88 (2005) 301–309
RELIABILITY ENGINEERING & SYSTEM SAFETY
www.elsevier.com/locate/ress

Estimation of component failure probability from masked binomial system testing data

Zhibin Tan

Software and Systems Engineering Research Lab, Motorola Laboratories, 1303 E. Algonquin Road, Schaumburg, IL 60196, USA

Received 25 May 2004; accepted 10 August 2004
Available online 7 October 2004

Abstract

The component failure probability estimates from analysis of binomial system testing data are very useful because they reflect the operational failure probability of components in the field which is similar to the test environment. In practice, this type of analysis is often confounded by the problem of data masking: the status of tested components is unknown. Methods in considering this type of uncertainty are usually computationally intensive and not practical to solve the problem for complex systems. In this paper, we consider masked binomial system testing data and develop a probabilistic model to efficiently estimate component failure probabilities. In the model, all system tests are classified into test categories based on component coverage. Component coverage of test categories is modeled by a bipartite graph. Test category failure probabilities conditional on the status of covered components are defined. An EM algorithm to estimate component failure probabilities is developed based on a simple but powerful concept: equivalent failures and tests. By simulation we not only demonstrate the convergence and accuracy of the algorithm but also show that the probabilistic model is capable of analyzing systems in series, parallel and any other user defined structures. A case study illustrates an application in test case prioritization.

Keywords: Reliability; Masked data; Equivalent failures; Bayes theorem; EM algorithm

1. Introduction

The component failure probability estimates from analysis of binomial system testing data are very useful because they reflect the operational failure probability of components in the field which is similar to the test environment. In practice, however, this type of analysis is often confounded by data masking. That is, components to cause the test failure are unknown.

Estimating component reliability or failure probability from masked system life data has received a lot of attention in the literature [1–6]. As stated in [2], component reliability is often estimated by using a series system assumption and applying a competing-risk model. The observable quantities are the system-life (failure or censor time) and partial information on the cause of failure. The analysis output is the maximum likelihood estimate (MLE) for component-life distribution parameters. The analysis is mostly from a classical statistics perspective. For example, Miyakawa considers a 2-component series system of 'exponential' components and derives closed-form expressions for the MLE [7]. Under the same exponential assumption, Ref. [4] extended the Miyakawa results to a 3-component system; in all but a few special cases, close-form MLE is intractable, and a simple iterative solution was proposed. In [1] an iterative pseudo-graphical approach was presented to estimate Weibull component reliability from masked system life data. Ref. [6] further developed a procedure for finding the exact MLE in the 3-component case. Most recently a Bayes methodology for estimating component reliability from masked system-life data was proposed [2]. It allows the analyst to directly quantify the prior engineering judgment in the development of the component reliability estimates.

In this paper we consider binomial system testing data with totally unknown component causes of system test failure. We aim at estimating component failure probabilities from such data. The observable quantities are the number of

E-mail address: zhibin.tan@motorola.com

0951-8320/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2004.08.013

system failures in n system tests. The output of analysis is the MLE of component failure probabilities.

Estimating component failure probabilities from this type of masked data has not received much attention. Methods in solving this type of problem are usually computationally intensive and not practical for complex systems in practice. In this paper, we develop an EM algorithm to efficiently estimate component failure probability from masked system testing data. The Expectation–Maximization (EM) algorithm, first proposed by Dempster [8], is an elaborate technique for solving the MLE problem when the observed data is incomplete or has missing values. It is remarkable because of its simplicity, generality and convergence. The EM algorithm is a general approach to iterative computation of maximum likelihood estimates. Each iteration consists of an expectation step followed by a maximization step. The more detailed information is available in [9–11]. In this paper the idea of the developed EM algorithm can be explained by a simple, intuitive but useful concept of equivalent failures and tests: Assume all component failure probabilities are initially known. Based on Bayes theorem, the failure probability of a component conditional on a system test result can be calculated. The conditional failure probability is defined as the equivalent failure for the component per test. The total equivalent failures for a component are obtained by summing up all the equivalent failure per test over all tests. The equivalent tests for a component are all system tests that cover the component. Assume component failures are independent in all covered system tests and follow a Binomial distribution. Then, the MLE of the failure probability of the component is the total number of equivalent failures divided by the total number of equivalent tests for that component. Component failure probabilities are updated with the obtained MLE and the process is repeated until the probabilities converge.

To make the approach even more efficient, we classify all system tests into test categories based on component coverage. A system test may involve one or more components. The set of components involved in a test defines the component coverage of the test. System tests with same component coverage are put into the same test category and have the same failure probability, which is also defined as the failure probability for that test category. Formally, we model the component coverage of test categories with a bipartite graph, in which nodes represent test categories and components, and links define component coverage of test categories. The graphical model can be viewed as a specific Bayesian network [12] with two layers of nodes. Failure probabilities of test categories conditional on components are defined based on some system structure assumption.

In the rest of paper, we present the probabilistic model in Section 2. Simulation studies are presented in Section 3. In Section 4, we present a case study and demonstrate an application of the approach in test case prioritization in system regression and integration testing. Finally, we conclude in Section 5.

2. The probabilistic model

In this paper, we consider systems that can be decomposed into multiple components. In system testing, a test covers one or more components. We classify tests into test categories based on component coverage. We can do so because the numbers of tests and failures in test categories are sufficient statistics in estimating component failure probabilities. The classification not only simplifies the model presentation but also improves the algorithm efficiency because the number of categories is usually much less than the number of tests. We treat tests in a category as a random sample for the category.

In this section, we first describe the graphical presentation. Next, we show how to define conditional failure probabilities of test categories based on system structures. Then we derive component equivalent failures and tests. Finally, we summarize the EM algorithm for evaluating component failure probabilities from masked system testing data.

2.1. Graphical presentation

Consider a system consisting of n components, denoted as $i=1,2,\ldots,n$. Assume these components are independent. We classify all system tests into m categories based on component coverage. A system test covers one or more components in the system. The set of covered components in a test defines the component coverage for the test. Assume all system tests with same component coverage have the same failure probability. Denote the m categories as $j=1,2,\ldots,m$. Assume two types of test results: failure and pass. Let $X_i=1(0)$ represent the event that component i fails (passes) and $Y_j=1(0)$ represent the event that a test in category j, or simply category j, fails (passes). Denote $\Pr(X_i=1)$ as p_i, the failure probability of component i, for $i=1,2,\ldots,n$. Denote $\Pr(Y_j=1)$ as q_j, the failure probability of a test in category j, or simply category j, for $j=1,2,\ldots,m$. A probabilistic model for variables $\{X_1, X_2,\ldots,X_n, Y_1, Y_2,\ldots,Y_m\}$ consists of (1) a bipartite graph $\boldsymbol{G}=(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{E})$ that models component coverage of test categories, and (2) conditional dependence defined between $\boldsymbol{X}$ and $\boldsymbol{Y}$. In $\boldsymbol{G}=(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{E})$, $\boldsymbol{X}$ is the set of nodes representing components $1,2,\ldots,n$; $\boldsymbol{Y}$ is the set of nodes representing categories $1,2,\ldots,m$. $\boldsymbol{E}$ is the set of links between $\boldsymbol{X}$ and $\boldsymbol{Y}$. If nodes i and j are connected, it means that category j covers component i. As an example, Fig. 1 is the bipartite graph representing component coverage of categories in a system with three components and four categories.

Define C_j as component coverage of category j, represented as a set of covered components. In Fig. 1, $C_1=(1, 2)$, $C_2=(2, 3)$, $C_3=(1, 3)$, and $C_4=(1, 2, 3)$.

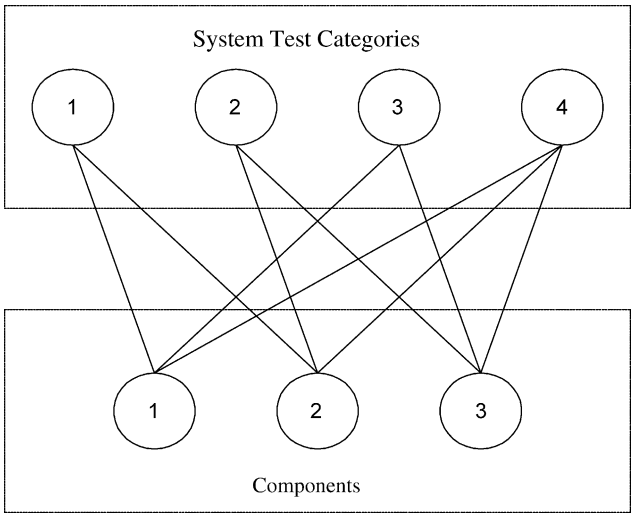

Fig. 1. Component coverage of test categories in a system with three components.

Table 1
Configuration definitions for C_1

Configurations	Variables and values	
	X_1	X_2
D_1^1	1	1
D_1^2	1	0
D_1^3	0	1
D_1^4	0	0

Conditional dependence between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is described by probabilities of Y_j conditional on its all configurations of C_j, for $j=1,2,\ldots,m$. A configuration of C_j is that all components in C_j are in a certain state. Since each component can have only two states (pass or fail), a coverage C_j with $\|C_j\|$ components has totally $r_j = 2^{\|Cj\|}$ configurations, denoted as $D_j^1, D_j^1, \ldots D_j^{rj}$. For instance, consider coverage C_1 in Fig. 1. Since C_1 has two components 1 and 2, it has four configurations as listed in Table 1.

For configuration D_l^j, $j=1,\ldots,m$, $l=1,\ldots,r_j$, we define $\theta_{jl} = \Pr(Y_j = 1|D_l^j)$, the failure probability of category j conditional on configuration D_j^l. For example, θ_{11} is the failure probability of the first category when both components 1 and 2 fail.

2.2. Component conditional failure probabilities

Based on component coverage and probabilities of categories conditional on their configurations, we derive component failure probabilities conditional on test results, meaning Pr $(X_i=1|Y_j=1)$ and Pr $(X_i=1|Y_j=0)$ for $i=1,\ldots,n$, $j=1,2,\ldots,m$

From probability calculus, we have

$$q_j = \Pr(Y_j = 1) = \sum_{l=1}^{r_j} \Pr(Y_j = 1|D_j^l)\Pr(D_j^l) = \sum_{l=1}^{r_j} \theta_{jl}\Pr(D_j^l) \tag{1}$$

where $\Pr(D_j^l)$ is the probability of covered component to take values defined in configuration D_j^l. For example, $\Pr(D_j^l) = \Pr(X_1 = 1, X_2 = 1)$. With assumption that X_1, $X_2,\ldots,X_n$ are independent, we have

$$\Pr(D_j^l) = \prod_{i\in C_j} \Pr(X_i = D_j^l(X_i)) \tag{2}$$

where $D_j^l(X)$ is the value of X in configuration D_j^l. For instance, $D_1^1(X_1) = D_1^2(X_1) = 1$, $D_1^3(X_1) = D_1^4(X_1) = 0$.

From Eq. (1), we have

$$\Pr(Y_j = 0) = \sum_{l=1}^{r_j}(1 - \theta_{jl})\Pr(D_j) = 1 - \Pr(Y_j = 1) \tag{3}$$

Define a function $1\{\}$ as $1\{\text{True}\}=1$ and $1\{\text{False}\}=0$. Consider any $i \in C_j$. By marginating out other components in C_j, the joint probabilities of Y_j and X_i are

$$\Pr(Y_j = 1,\ X_i = 1) = \sum_{l=1}^{r_j} \Pr(Y_j = 1|D_j^l)\Pr(D_j^l)1\{D_j^l(X_i = 1)\}$$

$$\Pr(Y_j = 0,\ X_i = 1) = \sum_{l=1}^{r_j} \Pr(Y_j = 0|D_j^l)\Pr(D_j^l)1\{D_j^l(X_i = 1)\}$$

From Bayes theorem, we have

$$\Pr(X_i = 1|Y_j = 1) = \frac{\sum_{l=1}^{r_j} \Pr(Y_j = 1|D_j^l)\Pr(D_j^l)1\{D_j^l(X_i = 1)\}}{\Pr(Y_j = 1)} \tag{4}$$

$$\Pr(X_i = 1|Y_j = 0) = \frac{\sum_{l=1}^{r_j} \Pr(Y_j = 0|D_j^l)\Pr(D_j^l)1\{D_j^l(X_i = 1)\}}{\Pr(Y_j = 0)} \tag{5}$$

where $\Pr(Y_j=1)$ and $\Pr(Y_j=0)$ are defined in Eqs. (1) and (3) and $\Pr(D_j^l)$ is given in Eq. (2).

For example, consider C_1 in Fig. 1. We have

$$q_1 = \Pr(Y_1 = 1) = \theta_{11}p_1p_2 + \theta_{12}p_1(1 - p_2) + \theta_{13}(1 - p_1)p_2 + \theta_{14}(1 - p_1)(1 - p_2)$$

$$\Pr(X_1,\ Y_1 = 1) = \theta_{11}p_1p_2 + \theta_{12}p_1(1 - p_2)$$

$$\Pr(X_1 = 1,\ Y_1 = 0) = (1 - \theta_{11})p_1p_2 + (1 - \theta_{12})p_1(1 - p_2)$$

$$\Pr(X_1 = 1|Y_1 = 1) = \frac{(\theta_{11}p_1p_2 + \theta_{12}p_1(1 - p_2))p_1}{\theta_{11}p_1p_2 + \theta_{12}p_1(1 - p_2) + \theta_{13}(1 - p_1)p_2 + \theta_{14}(1 - p_1)(1 - p_2)}$$

$$\Pr(X_1 = 1|Y_1 = 0) = \frac{(1-\theta_{11})p_1p_2 + (1-\theta_{12})p_1(1-p_2)}{1-(\theta_{11}p_1p_2 + \theta_{12}p_1(1-p_2) + \theta_{13}(1-p_1)p_2 + \theta_{14}(1-p_1)(1-p_2))}$$

2.3. Component equivalent failures and tests

To evaluate component failure probability, we can test the component n times. When k failures are observed, the MLE of the component failure probability is k/n. For masked data, failures are only observable at system level and the real number of failures for components may not be known. However, based on the derived conditional failure probabilities for components, we can define component equivalent failures and tests from system testing data.

For component $i \in C_j$, we define $\Pr(X_i=1|Y_j=1)$ as the equivalent failure for component i when category j fails; define $\Pr(X_i=1|Y_j=0)$ as the equivalent failure for component i when category j passes. When k_j failures are observed in n_j tests in category j, the equivalent failures for component $i \in C_j$ can be expressed as

$$f_{ij} = k_j \Pr(X_i = 1|Y_j = 1) + (n_j - k_j)\Pr(X_i = 1|Y_j = 0) \tag{6}$$

Since category j covers component i, each test in j tests i, the number of equivalent tests for i in j is n_j. With considering all categories, the total number of equivalent failures for component i can be expressed as

$$f_i = \sum_{j=1}^{m} f_{ij} 1\{i \in C_j\} \tag{7}$$

and the total number of equivalent tests is

$$t_i = \sum_{j=1}^{m} n_j 1\{i \in C_j\} \tag{8}$$

This is the E-step in the EM algorithm. Applying the M-step in the EM algorithm, we obtain the MLE of the failure probability for component i, $i=1,2,\ldots,n$ as

$$p_i = f_i/t_i \tag{9}$$

2.4. The EM algorithm

Based on the results obtained in previous sections, we summarize the EM algorithm for estimating component failure probabilities. In the algorithm, first we assume we know component failure probabilities by setting their initial values. With known component coverage, category probabilities conditional on configurations, and test sufficient statistics, we obtain the number of equivalent failures and tests for components. We update component failure probabilities using the equivalent failures and tests. We repeat this process until the component failure probabilities converge. The input to the algorithm includes the bipartite graph which models component coverage of test categories C_j, testing results in categories (k_j, n_j), and category failure probabilities conditional on configurations $\Pr(Y_j|D_j^l)$. The output is the component failure probabilities. The algorithm is listed in Table 2. In the algorithm, $\Pr(X_i=1|Y_j=1)$ and $\Pr(X_i=1|Y_j=0)$ are given in Eqs. (4) and (5).

In the algorithm, the major computation is calculation of category failure probabilities from component failure probabilities and component failure equivalent failures. From Eq. (1), for category j, the failure probability calculation involves $r_j \times \|C_j\|$ multiplications and $r_j - 1$ additions. In practice, the numbers of multiplications and additions can be much less for some specific structure. For instance, for series structures, the number of multiplications is only $\|C_j\|$, and there is no addition operation involved, because the category failure probability is the product of covered component failure probabilities. To calculate equivalent failure for a component per test category, it involves two conditional probability computations. When the category failure probabilities are calculated, these two conditional probabilities are easy to calculate. Simply put, the algorithm is simple and not computationally intensive.

Table 2
Algorithm for estimation of component failure probability from masked binomial system testing data

Input: C_j, (k_j, n_j), $\Pr(Y_j|D_j^l)$, for $l=1,2,\ldots,r_j$, $j=1,\ldots,m$
Output: p_i, for $i=1,\ldots,n$, component failure probabilities
Steps:
1. Initialize component failure probabilities $p_1,p_2,\ldots,p_n$
2. Calculate category failure probabilities $q_1,q_2,\ldots,q_m$, as defined in Eq. (1)
3. E-step: Calculate component equivalent failures f_i and tests t_i. For component i, $i=1,\ldots,n$,
   (a) Initialize $f_i=0$ and $t_i=0$
   (b) For $j=1$ to m
      If $i \in C_j$ then
         $f_i = f_i + k_j \times \Pr(X_i = 1|Y_j = 1) + (n_j - k_j) \times \Pr(X_i = 1|Y_j = 0)$
         $t_i = t_i + n_j$
4. M-step: Update component failure probabilities by letting $p_i = f_i/t_i$, for $i=1,2,\ldots,n$
5. Repeat 2–4 until $p_1,p_2,\ldots,p_n$ converge

Table 3
Configurations and conditional probabilities in category 1

	Configurations		$\theta_{1l} = \Pr(Y_1 = 1 \mid D_1^l)$		
	X_1	X_2	Case I	Case II	Case III
D_1^1	1	1	1	1	0.9
D_1^2	1	0	1	0	0.6
D_1^3	0	1	1	0	0.6
D_1^4	0	0	0	0	0.1

3. Simulations

In this section, we conduct simulations to demonstrate convergence and accuracy of the proposed algorithm in Section 2.4. In simulations, we consider a system with component coverage as defined in Fig. 1. In the following, we first define category conditional probabilities $\Pr(Y_j|D_j^l)$ with three different system structures. Next, we generate sample data (k_j, n_j) for four categories. Finally, we apply the algorithm to the generated data and obtain estimated component probabilities. Some observations are made based on the simulation results.

3.1. Definition of probabilities conditional on configurations

For the same component converge, we may have different definitions for category failure probabilities when system structures are different. In this study, we consider three cases regarding the failure relations between components and categories. In Case I, components are in series and a failure of any covered component causes the category to fail. In Case II, components are in parallel and a category fails only when all its covered components fail. We take Case III as a general one. The conditional probabilities can be defined by experts as any valid values. For completeness, we list conditional failure probabilities for the three cases in Tables 3–6.

3.2. Data generation

Two steps are taken to generate a test sample. The first step is to determine the category to which a test belongs. The second step is to determine the test result. In all simulations, we set probabilities of a test to belong to the four categories are 0.1, 0.3, 0.4, and 0.2. To determine the category for a test, we randomly generate a number between 0 and 1. If the number falls between 0 and 0.1, the test belongs to the first test category; if the number falls between 0.1 and 0.4, the sample belongs to the second category; if the number falls between 0.4 and 0.8, the sample belongs to the third category; and if the number falls between 0.8 and 1.0, the test belongs to the fourth category.

To determine the result of a test in a category, we first determine the configuration by determining the status of each component in that category. That is, we need to know if each component covered by the test fails or not in that test. We set component failure probabilities as $p_1 = 0.4$, $p_2 = 0.1$, $p_3 = 0.2$ in all simulations. To determine if a component fails in a test, we generate a random number between 0 and 1. If the number is less than the set probability of the component, we say the component fails; otherwise, the component passes. The status of all components covered by the test defines the configuration. Next we determine the test result with determined configuration by generating a random number between 0 and 1. If the number is less than the conditional probability defined for that configuration, we say that the test fails; otherwise, the test passes.

In a simulation, we set sample size as n. Using the above two steps, we generate n test samples. For category j, $j = 1,2,3,4$, we count the numbers of tests and failures n_j and k_j in that category, which are the inputs to the algorithm.

Table 4
Configurations and conditional probabilities in category 2

	Configurations		$\theta_{2l} = \Pr(Y_2 = 1 \mid D_2^l)$		
	X_1	X_2	Case I	Case II	Case III
D_2^1	1	1	1	1	0.9
D_2^2	1	0	1	0	0.6
D_2^3	0	1	1	0	0.6
D_2^4	0	0	0	0	0.1

Table 5
Configurations and conditional probabilities in category 3

	Configurations		$\theta_{3l} = \Pr(Y_3 = 1 \mid D_3^l)$		
	X_1	X_2	Case I	Case II	Case III
D_3^1	1	1	1	1	0.9
D_3^2	1	0	1	0	0.6
D_3^3	0	1	1	0	0.6
D_3^4	0	0	0	0	0.1

3.3. Simulation results and analysis

For each case, we run simulation multiple trials. We study the possible impact of sample size, initial settings of

Table 6
Configurations and conditional probabilities in category 4

	Configurations			$\theta_{4l} = \Pr(Y_4 = 1 \mid D_4^l)$		
	X_1	X_2	X_3	Case I	Case II	Case III
D_4^1	1	1	1	1	1	0.9
D_4^2	1	1	0	1	0	0.7
D_4^3	1	0	1	1	0	0.7
D_4^4	1	0	0	1	0	0.7
D_4^5	0	1	1	1	0	0.3
D_4^6	0	1	0	1	0	0.3
D_4^7	0	0	1	1	0	0.3
D_4^8	0	0	0	0	0	0.1

Table 7
Simulation results in Case I with sample size of 1000 and iteration step of 100

Variables	Settings	From data	From algorithm
X_1	0.400	0.406	0.427
X_2	0.100	0.102	0.090
X_3	0.200	0.201	0.190
Y_1	0.460	0.472	0.479
Y_2	0.280	0.261	0.263
Y_3	0.520	0.533	0.538
Y_4	0.568	0.588	0.578

component probabilities and randomness of data generation on convergent and accuracy of the algorithm.

To study the algorithm accuracy, we compare three different sets of probabilities. The first set of probabilities are true values from settings. Giving components probabilities and conditional probabilities on configurations, we can calculate category failure probabilities using Eq. (1). For example, in Case I where components are in series, we have failure probabilities for the four categories as $q_1=0.36$, $q_2=0.28$, $q_3=0.52$, and $q_4=0.568$ when $p_1=0.4$, $p_2=0.1$, $p_3=0.2$. The second set of probabilities are from generated data. The point estimate of failure probability for category j is obtained by k_j/n_j, $j=1,2,3,4$. Similarly, we count test and failure numbers for each component in the process of data generation. The point estimate of failure probability of that component is the ratio of the failure number to the test number. The third set of probabilities are obtained by using the developed algorithm. For components, the probabilities are directly obtained from the algorithm. For categories, the probabilities are calculated using obtained component probabilities and setting conditional probabilities based on Eq. (1).

To show the accuracy of the algorithm, we list simulation results for the three cases in Tables 7–9. Though there is variation between trials, the variation is small and the listed results should reflect the behavior of the algorithm in general.

To demonstrate the convergent of the algorithm, we display component and category failure probabilities along the iterative process for each case in Figs. 2–4. Category failure probabilities are calculated based on the obtained component probabilities. Since component probabilities converge, the category probabilities converge too.

Table 8
Simulation results in Case II with sample size of 10,000 and iteration step of 500

Variables	Settings	From data	From algorithm
X_1	0.400	0.400	0.437
X_2	0.100	0.104	0.106
X_3	0.200	0.198	0.179
Y_1	0.040	0.049	0.046
Y_2	0.020	0.020	0.019
Y_3	0.080	0.079	0.078
Y_4	0.008	0.006	0.008

Table 9
Simulation results in Case III with sample size of 10,000 and iteration step of 200

Variables	Settings	From data	From algorithm
X_1	0.400	0.409	0.423
X_2	0.100	0.096	0.089
X_3	0.200	0.203	0.212
Y_1	0.342	0.351	0.349
Y_2	0.246	0.248	0.247
Y_3	0.384	0.401	0.399
Y_4	0.265	0.264	0.271

From simulations conducted on the system in Fig. 1, following observations can be made:

1. The algorithm converges. In fact, the convergence of EM algorithms has been strictly proved [8]. The initial setting of component probabilities has no impact on convergence, though it may have impact on the converging speed.
2. The converged probabilities are close to probabilities calculated from data. With large sample size, the probabilities from data are very close to the true probabilities. Therefore, the converged probabilities are very close to the true values.

4. Case study

In this section, we present a case study and demonstrate a real application of the proposed approach. We applied the approach to test case prioritization in system regression and integration testing. Regression testing is to verify that modifications have not caused unintended effects and that the system still complies with its specified requirements. In integration testing, both software and hardware are combined and tested to evaluate the interaction between

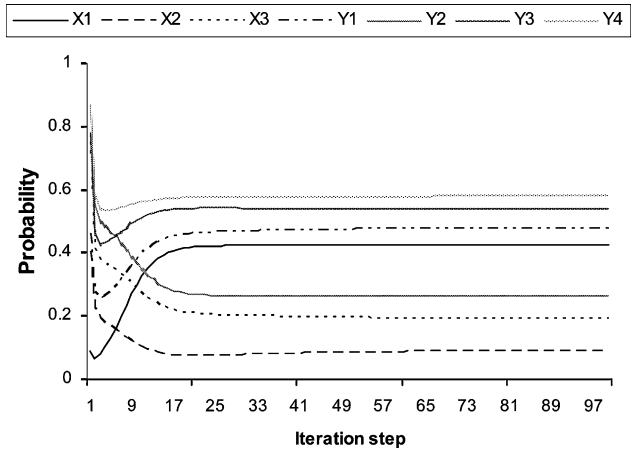

Fig. 2. Demonstration of convergence in Case I. The sample size is 1000.

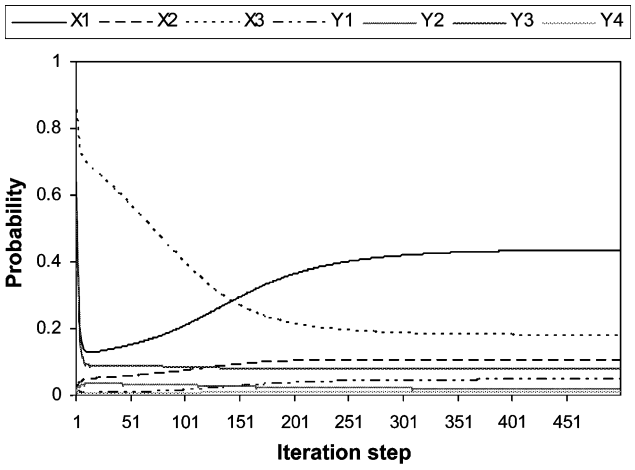

Fig. 3. Demonstration of convergence in Case II. The sample size is 10,000.

them. Test case prioritization techniques help engineers execute test cases in an order that achieve testing objectives earlier in the testing process.

One testing objective involves defect arrival rate—a measure of how quickly a test order finds defects. An improved defect arrival rate can provide earlier feedback on the system under test, enable earlier debugging, and increase the likelihood that, if testing is prematurely halted, those test cases that offer the greatest defect detection ability in the available testing time will have been executed. Numerous prioritization techniques have been described in the research literature [13–15]. To date, most proposed techniques have been code-based, relying on information relating test cases to coverage of code elements and changes made in code elements. Kim and Porter [16] present a 'history based prioritization' technique, in which information from previous regression testing cycle is used to better inform the selection of a subset of an existing test suite for use on a modified version of a system. This technique is not, however, a 'prioritization technique' because it imposes no ordering on test cases. Rather, the approach is more accurately described as 'test case selection technique' [17]

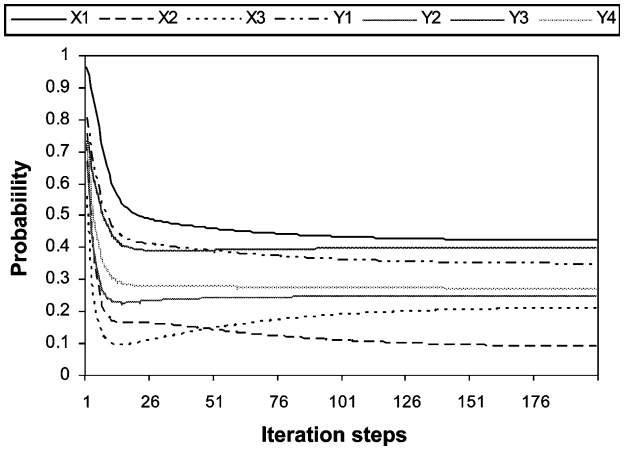

Fig. 4. Demonstration of convergence in Case III. The sample size is 10,000.

Table 10
Summary of testing data in five releases

Releases	R_1	R_2	R_3	R_4	R_5
No. of executions	8765	16,607	24,663	15,392	14,890
No. of test cases	4089	5484	11,588	9050	7949
No. of failures	70	97	271	279	141
No. of defects	55	89	280	141	98

because it uses history information to determine which test cases should be selected.

Our approach to prioritizing test cases is also history based. We prioritize test cases based on their failure probabilities calculated from historical testing information. The system we worked on is a telecommunication system which combines the capabilities of a digital cellular telephone, two-way radio, alphanumeric pager, and data/fax modem in a single network. It is based on Global System for Mobile communication (GSM) architecture. The network can be decomposed into 24 functional entities, each of which is a system. Those entities, also called components in this context, are functionally independent. A test case, usually representing a use case, covers one or more components. With a series assumption, for an execution of a test case to pass, all tested components should pass.

We applied the EM method to the testing data in five releases. For convenience, we denote the releases as R_1, R_2, R_3, R_4 and R_5. Every release has a set of test cases and a set of executions. A test case may be executed multiple times in a release and may have different results in different tests. The execution result is either pass or fail. When an execution fails, the corresponding test case is associated with a defect. A test case may find different defects in different executions and a defect may be found by multiple test cases. In each release, both old and new test cases are included. However, only a portion of old test cases from all previous releases are selected by test engineers. The components covered by each test case are identified. The component coverage of a test case does not change over releases. Table 10 is a summary of testing data for the five

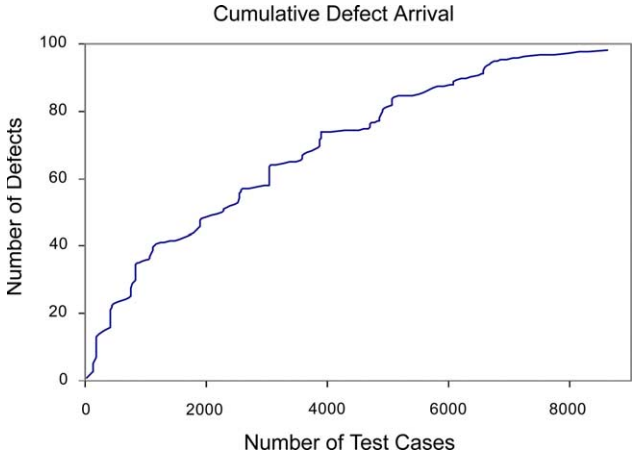

Fig. 5. Cumulative defect arrival for R_5.

Table 11
Average component probabilities in the fifth release

Components	1	2	3	4	5	6	7	8
Probabilities	0.012	0.0060	0.0029	0.016	0.0069	0.012	0.0050	0.010
Components	9	10	11	12	13	14	15	16
Probabilities	0.0090	0.011	0.013	0.018	0.013	0.011	0.038	0.0030
Components	17	18	19	20	21	22	23	24
Probabilities	0.0040	0.0061	0.0061	0.049	0.023	0.018	0.031	0.019

releases. For prioritizing test cases in R_s for $s=2,\ldots,5$, historical execution information in $R_1,\ldots,R_{s-1}$ is available. We could use information only in R_{s-1}, which is the latest to R_s, or we could use information in all releases equally. Here we applied the Exponential Weighted Moving Average (EWMA) model [18] to use all historical execution information but assign geometrically decreasing weights for past releases.

Denote $\boldsymbol{P}_s=(p_{s1},p_{s2},\ldots,p_{s24})$ as the set of component failure probabilities obtained by using execution information only in R_s; denote $\boldsymbol{Q}_s=(q_{s1},q_{s2},\ldots,q_{s24})$ as the set of component failure probabilities obtained by exponentially weighted average of $\boldsymbol{P}_1,\boldsymbol{P}_2,\ldots,\boldsymbol{P}_s$. Then we have

$$\boldsymbol{Q}_s = \lambda \boldsymbol{P}_s + (1-\lambda)\boldsymbol{Q}_{s-1} \tag{10}$$

in which λ is the discount factor. Using a larger value for λ results in weights that die out more quickly and place more emphasis on recent information. A value of 0.4 for λ is often used in practice.

To obtain $\boldsymbol{P}_s$, we applied the EM Algorithm in Section 2.4 to the testing data in R_s. For the first release R_1, the failure probabilities were randomly initialized. For releases R_s, $s>1$, the failure probabilities $\boldsymbol{P}_s$ were initialized as $\boldsymbol{P}_{s-1}$.

Once $\boldsymbol{P}_s$ and $\boldsymbol{Q}_s$ were obtained, we calculated the failure probabilities of test cases in the release of R_s. Denote C_{test} as the set of components covered by a test case. With the series assumption, the failure probability of that test case was calculated as

$$p_{\text{test}} = 1 - \prod_{i \in C_{\text{test}}} (1-q_{si}) \tag{11}$$

We sorted test cases by their failure probabilities in a decreasing order, meaning we would execute test cases with higher failure probability first. With the obtained order of test cases, we plotted defect arrival curves to demonstrate testing effectiveness in that order. For example, the cumulative defect arrival curve in R_5 when $\lambda=0.6$ is displayed in Fig. 5. In that release, totally 7949 test cases were executed and 98 unique defects were found. The calculated $\boldsymbol{Q}_5$ is listed in Table 11. Notice that failure probability of test cases is one of many factors that could be used to prioritize test cases. Other factors from historical information may include defect severity, execution time, etc. Discussion on optimizing these factors is beyond this study.

5. Conclusions

In this paper, we presented a probabilistic model and an EM algorithm for evaluating component failure probability from masked binomial system testing data. By classifying all system tests into test categories based on component coverage, we not only simplify the model presentation but also greatly improve the efficiency of the algorithm. The approach was demonstrated to be very efficient and capable of analyzing complex systems with large number of components.

The concept of the equivalent failures and tests from the EM algorithm is simple, intuitive and powerful. In the future, we will show that this concept can be easily applied to estimate component reliability from masked system life data when the failure time of components follows exponential distribution.

Acknowledgements

The author thanks the two anonymous referees for their helpful suggestions and comments. He also thanks Wei Xi for useful discussions.

References

[1] Usher JS, Guess FG. An iterative approach for estimating component reliability from masked system life data. Qual Reliab Eng Int 1989; 5(4):257–61.
[2] Lin DKJ, Usher JS, Guess FM. Bayes estimation of component-reliability from masked system-life data. IEEE Trans Reliab 1996; 45(2):233–7.
[3] Usher JS. Weibull component reliability-prediction in the presence of masked data. IEEE Trans Reliab 1996;45(2):229–32.
[4] Usher JS. Maximum likelihood analysis of component reliability using masked system life-testing data. 1988;37(5):550–5.
[5] Doganaksoy N. Interval estimation from censored and masked system failure data. IEEE Trans Reliab 1991;40(3):280–6.
[6] Lin DKJ, Usher JS, Guess FM. Exact maximum likelihood estimation using masked system data. IEEE Trans Reliab 1993;42(4):631–5.
[7] Miyakawa M. Analysis of incomplete data in competing risk model. IEEE Trans Reliab 1984;R-33:293–6.
[8] Dempster AP, Laird NM, Rubin DB. Maximum-likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B 1977;39.
[9] Bilmes J. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report. University of Berkeley; 1997 [ICSI-TR-97-021].

[10] Calder K, Holloman C. The EM algorithm and missing data problem; Apr 12, 2000. [Online article, http://www.stat.duke.edu/∼chris/research/EM.pdf]
[11] Kuo L, Peng F. A mixture-model approach to the analysis of survival data. [Online article, http://citeseer.nj.nec.com/363619.html]
[12] Heckerman D. Bayesian network for knowledge discovery. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthrurusamy R, editors. Advances in knowledge discovery and data mining. Menlo Park, CA: AAAI Press; 1996.
[13] Elbaum S, Malishevsky A, Rothermel G. Incorporating varying test costs and fault severities into test case prioritization. Int Conf Software Eng 2001;329–38.
[14] Elbaum S, Malishevsky A, Rothermel G. Test case prioritization: a family of emprirical studies. IEEE Trans Software Eng 2002;28(2):159–82.
[15] Srivastava A, Thiagarajan J. Effectively prioritizing tests in development environment. Proc Int Symp Software Test Anal 2002;97–106.
[16] Kim JM, Porter A. A history-based test prioritization technique for regression testing in resource constrained environments. Proc Int Conf Software Eng 2002.
[17] Rothermel G, Harrold M. Analyzing regression test selection techniques. IEEE Trans Software Eng 1996;22(8):529–51.
[18] Box G, Luceno A. Statistical control by monitoring and feedback adjustment. New York: Wiley; 1997.