이항 반응 계수를 가진 연속 시간형 HGDM의 개발

박 중 양† · 김 성 희†† · 박 재 흥†††

요 약

초기하분포 소프트웨어 신뢰성 성장 모델(HGDM)은 최근에 개발되어 테스트와 디버그의 시작 단계에서 소프트웨어에 남아 있는 초기 결함 수를 추정할 수 있는 문제에 성공적으로 적용되고 있다. 그러나 HGDM은 시간 도메인 소프트웨어 신뢰성 성장 모델(SRGM)에 속하지만 시험자 수 등과 같은 시험에 투입하는 자원을 고려하는 과정에서 다른 시간 도메인 SRGM과 비교하기 곤란한 점을 내포하게 되었다. 특히, 시간 도메인 SRGM에서 일반적으로 사용하는 소프트웨어 신뢰성을 계산할 수 없다. 본 논문은 HGDM이 시간에 의해 기술되지 않음으로 인해 생기는 이러한 문제점을 해결하기 위해 이항 반응 계수를 가진 연속 시간형 HGDM을 개발하고 그 특성을 연구한다. 그리고 제안된 모델을 실제 자료에 적용해서 기존 HGDM을 대신하여 사용할 수 있음을 보인다.

Development of the Continuous-Time HGDM with Binomial Sensitivity Factor

Joong-Yang Park† · Seong-Hee Kim†† · Jae-Heong Park†††

ABSTRACT

The hyper-geometric distribution software reliability growth model (HGDM) was recently developed and successfully applied to the problem of estimating the number of initial faults residual in a software at the beginning of the test-and-debug phase. Though the HGDM is a time-domain software reliability growth model (SRGM), it is not possible to compare the HGDM with other time-domain SRGMs. Furthermore the usual software reliability can not be computed. These drawbacks are derived from fact that the HGDM is not described in terms of the execution time. Thus we develop a continuous-time HGDM with binomial sensitivity factor in order to remove these drawbacks. Statistical characteristics of the suggested model are studied and its applicability is then examined by analyzing real test data sets. It is empirically shown that the continuous-time HGDM with binomial sensitivity factor can be used as an alternative to the current HGDM.

1. Introduction

Computer systems play important roles in the highly computerized society. The failure of a computer system caused by software faults may result in enormous damage. It is therefore required to develop reliable software systems. Software reliability, defined as the probability of its failure-free operation during a specified time interval in a specified environment, has become a central software engineering concept. At the beginning of the test-and-debug phase nobody knows definitely how reliable the software system is. Test workers are unable to certify when the software system can be released into service. Many software reliability

† 정 회 원 : 경상대학교 통계학과 교수
†† 준 회 원 : 경상대학교 대학원 전자계산학과
††† 정 회 원 : 경상대학교 컴퓨터과학과 교수
논문접수 : 1999년 8월 19일, 심사완료 : 1999년 10월 28일

growth models (SRGMs) have been developed for measuring and projecting the software reliability during the test-and-debug phase. The SRGMs allow us to estimate the number of remaining faults, software reliability and other software quality assessment measures. Though some SRGMs usually works better than others, there is no single SRGM that works best for all software systems. Some of currently available SRGMs probabilistically predict the time to the next occurrence of failure or the number of failures during a specified time interval. Another class of SRGMs estimates the number of software faults still residual at the beginning of the test-and-debug phase. The hyper-geometric distribution software reliability growth model (HGDM), advocated by Tohma and Tokunaga [20], provides us with the number of software faults still residual. A series of studies on the HGDM has been published recently [2-7, 10, 13, 14, 19-23]. It was successfully applied to real data sets.

The HGDM belongs to the class of time-domain SRGMs. Most time-domain SRGMs are described in terms of time measurement associated with software usage. The software reliability, defined in the previous paragraph, can be computed directly from the time-domain SRGMs. However, HGDM granulates the time and is described in terms of sequence number of the granulated time even when a certain software usage-dependent time measure is observed. The observed time measurements are treated as the resource associated with the learning factor. As a result it is difficult to compare the HGDM with other time-domain SRGMs and compute the usual software reliability based on the HGDM. In this paper we will first develop a version of HGDM expressed in terms of the software usage-dependent time measurement (execution time in most cases). The newly developed version of HGDM will be called the continuous-time HGDM. Then we will study its characteristics and demonstrate its applications to real data sets. The remaining presentation is organized as follows. The basic concept and precise formulation of HGDM are briefly reviewed in Section 2. Section 3 develops the continuous-time HGDM with binomial sensitivity factor. Parameter estimation problem for the continuous-time HGDM will be considered in Section 4. Section 5 gives illustrative examples for real data sets. The last section concludes by some comment.

2. Review of HGDM

This section briefly reviews the basic idea and formulation of HGDM. At the beginning of the test-and-debug phase a software system is assumed to have m initial faults. Test operations performed for a certain period, in a day or a week, may be called a test instance. Test instances are denoted by t_i, $i=1, 2, \cdots$, in accordance with the order of applying them. With the application of test instances t_i, errors caused by faults manifest themselves as failures. The sensitivity factor, w_i, represents the number of faults sensed by the application of test instance t_i. Some of the faults discovered by t_i may have been defected by the application of previous test instances t_j, $j=1, 2, \cdots, i-1$. Therefore the number of faults newly discovered by t_i is not necessarily equal to w_i. That is, each detected faults can be classified into two categories, newly discovered faults and rediscovered faults. Let N_i be the number of faults newly detected by t_i and C_i be the number of faults already detected up to t_i. Then $C_i = \sum_{j=1}^{i} N_j$. The following assumptions are made on the HGDM.

(A1) No new faults are introduced into the software system during the debugging process.

(A2) Sensitivity factor w_i is the faults taken randomly out of m initial faults.

(A3) Sensitivity factor w_i is represented as a function of the number of initial faults m and the progress in testing p_i, that is, $w_i = mp_i$.

Note that $0 \le p_i \le 1$, since $0 \le w_i \le m$. The term p_i is usually referred to as the learning factor. Due

to assumption (A1), the probability that x_i faults are newly discovered by t_i on condition that c_{i-1} faults has been discovered up to t_{i-1} is then formulated as

$$P(N_i = x_i \mid C_{i-1} = c_{i-1}) = \frac{\binom{m-c_{i-1}}{x_i}\binom{c_{i-1}}{w_i - x_i}}{\binom{m}{w_i}}, \quad (1)$$

where $\max(0, w_i - c_{i-1}) \le x_i \le \min(w_i, m - c_{i-1})$, $c_0 = 0$, $x_0 = 0$ and $i = 1, 2, \cdots$. Thus the conditional expected value of N_i is

$$E(N_i \mid C_{i-1} = c_{i-1}) = (m - c_{i-1})p_i.$$

The mean value function, the expected value of C_i, was obtained by Jacoby and Tohma [6] as

$$E(C_i) = m[1 - \prod_{j=1}^{i}(1 - p_j)]. \quad (2)$$

The sensitivity factor w_i plays the key role in HGDM. The sensitivity factor represents the number of faults discovered or rediscovered during the application of test instance t_i. It is usually assumed that w_i, ultimately p_i, depends on the resources used for executing test instance such as software usage, number of test workers and test items. It is further assumed that p_i depends on the improvement of test worker's skill along with the progress of testing. The term p_i is thus called the learning factor. Various plausible deterministic functions for p_i have been devised and successfully applied to real data sets [6, 10]. Recently Hou, Kuo and Chang [5] suggested the exponential and logistic learning factor based on the exponential and logistic learning curves.

Suppose that F_i represents the set of faults detected by t_i. Then $w_i = |F_i|$, where $|\cdot|$ is the cardinality of a set. Assumption (A2) implies that $|F_i|$ is deterministic, but the elements of F_i are randomly chosen from the initial m faults. This does not reflect enough the random behavior of testing process. Test items for a test instance are usually selected randomly from the input domain. The number of faults detected by the randomly selected test items would not be deterministic. That is, if different test items are executed for t_i, different number of faults would be discovered. It is therefore more reasonable to postulate that the sensitivity factor is a random variable. Henceforth we denote the random sensitivity factor by W_i. Tohma and Tokunaga [20] attempted the normally distributed sensitivity factor irrespective of i or the resources used for executing t_i. However, the results were not satisfactory. Park, Yoo and Park [14] assumed that W_i is a binomial random variable with parameters m and p_i, that is, for $w_i = 0, 1, \cdots, m$

$$P(W_i = w_i) = \binom{m}{w_i} p_i^{w_i}(1 - p_i)^{m - w_i}. \quad (3)$$

Then $P(N_i = x_i \mid C_{i-1} = c_{i-1}, W_i = w_i)$ is given by Equation (1) and

$$P(N_i = x_i \mid C_{i-1} = c_{i-1}) = \binom{m-c_{i-1}}{x_i} p_i^{x_i}(1 - p_i)^{m - c_{i-1} - x_i}. \quad (4)$$

It was further shown that if the least squares method was employed, estimation and prediction results of the binomial sensitivity factor HGDM are identical with those of HGDM. This implies that the binomial sensitivity factor HGDM performs at least as well as the deterministic sensitivity factor HGDM. The maximum likelihood (ML) estimation was further suggested as an alternative to the least squares estimation. We thus employ the binomial sensitivity factor in the rest of this paper.

A continuous-time HGDM will be developed in the next section. Thus the HGDM reviewed in this section will be referred to as the discrete HGDM.

3. Continuous-Time HGDM with Binomial Sensitivity Factor

The discrete HGDM is applicable to all kind of test data. It enables us to estimate the number of initial faults residual in a software at the beginning

of the test-and-debug phase. However, the HGDM has some drawbacks mentioned in Section 1. This is due to the fact that most time-domain SRGMs are expressed in software usage-dependent time measurement and the HGDM is not. The HGDM takes the time measurement into account as one of resources in modeling the learning factor p_i. Thus we modify the HGDM so that it is described in terms of the time and the learning factor is still dependent on the time. The modified HGDM will be called the continuous-time HGDM.

Suppose that the obtained software usage-dependent time measurement is the execution time. We denote the cumulative execution time by h. Then test operations performed during $[h_{i-1}, h_i)$ correspond to t_i, where h_i's are the cumulative execution times at which the test-and-debug process is observed and $h_0=0$. The cumulative number of faults newly detected during $[0, h)$ is denoted by $C(h)$. We further denote the binomial sensitivity factor by $W(h, \Delta h)$, which represents the number of faults sensed by the test operations during $[h, h+\Delta h)$. Here Δh is the execution time elapsed after h. Assumptions (A2) and (A3) are then replaced by the following assumption:

(A2)′ Sensitivity factor $W(h, \Delta h)$ is binomially distributed with parameters m and $p(h, \Delta h)$.

Here $p(h, \Delta h)$ is the probability that a fault is sensed by the test operations during $[h, h+\Delta h)$. Then the probability that x faults are newly discovered during $[h, h+\Delta h)$ on condition that $c(h)$ faults has been discovered up to h and w faults have been sensed during $[h, h+\Delta h)$ is then formulated as

$$P(C(h+\Delta h)-C(h)=x \mid C(h)=c(h), W(h, \Delta h)=w) = \frac{\binom{m-c(h)}{x}\binom{c(h)}{w-x}}{\binom{m}{w}}, \quad (5)$$

where $\max(0, w-c(h)) \le x \le \min(w, m-c(h))$ and $c(0)=0$. Since

$$P(W(h, \Delta h)=w) = \binom{m}{w} p(h, \Delta h)^w (1-p(h, \Delta h))^{m-w},$$

the probability that x faults are newly detected during $[h, h+\Delta h)$ on condition that $c(h)$ faults has been detected up to h becomes

$$\begin{aligned} &P(C(h+\Delta h)-C(h)=x \mid C(h)=c(h)) \\ &= \sum_w P(C(h+\Delta h)-c(h)=x \mid C(h)=c(h), \\ &\qquad W(h, \Delta h)=w) \cdot P(W(h, \Delta h)=w) \\ &= \binom{m-c(h)}{x} p(h, \Delta h)^x [1-p(h, \Delta h)]^{m-c(h)-x}. \end{aligned} \quad (6)$$

As discussed in the previous section, $p(h, \Delta h)$ depends on the learning, i. e., the growth of test team's potential during $[0, h+\Delta h)$. Thus we simply denote $p(h, \Delta h)$ by $p(h+\Delta h)$. Then Equation (6) is written as

$$\begin{aligned} &P(C(h+\Delta h)-C(h)=x \mid C(h)=c(h)) \\ &= \binom{m-c(h)}{x} p(h+\Delta h)^x [1-p(h+\Delta h)]^{m-c(h)-x}. \end{aligned} \quad (7)$$

Since $C(0) \equiv 0$, by replacing h and Δh in Equation (7) with 0 and h respectively, we have

$$\begin{aligned} P(C(h)=x) &= P(C(0+h)-C(0)=x \mid C(0)) \\ &= \binom{m}{x} p(h)^x [1-p(h)]^{m-x}. \end{aligned} \quad (8)$$

That is, $C(h)$ is binomially distributed with parameters m and $p(h)$. The expected value and variance of $C(h)$ are thus obtained as $mp(h)$ and $mp(h)[1-p(h)]$, respectively. If a software system has been tested up to h_n and $C(h_n)=c_n$, the software reliability is then computed from Equation (7) as

$$\begin{aligned} R(h \mid h_n) &= P(C(h_n+h)-C(h_n)=0 \mid C(h_n)=c_n) \\ &= [1-p(h_n+h)]^{m-c_n}. \end{aligned} \quad (9)$$

The continuous-time HGDM derived above may appear to be similar to the binomial-type SRGMs (Jelinski-Moranda [8], Shooman [18], Wagoner [24],

Schick-Wolverton [15, 16] and Littlewood [9] models are the binomial-type SRGMs). The binomial-type SRGMs are based on the following three assumptions:

(A4) Whenever a software failure occurs, the fault that caused it will be removed instantaneously.

(A5) There are m initial faults in the software system.

(A6) Each failure, caused by a fault, occurs independently and randomly in time according to the per-fault hazard rate $z(h)$.

Then the times to failure of each fault are independently and identically distributed as $F(h)$, where $F(h)=1-\exp\left(-\int_0^h z(y)dy\right)$. It can be shown that

$$P(C(h)=x)=\binom{m}{x}F(h)^x[1-F(h)]^{m-x}. \quad (10)$$

If $F(h)$ is replaced by $p(h)$, Equation (10) becomes identical to Equation (8). However these two types of SRGMs are different in some respects. First the conditional distribution of $C(h+\Delta h)$ given that $C(h)=c(h)$ is different. The conditional distribution for the binomial type SRGMs is obtained as

$$P(C(h+\Delta h)-C(h)=x \mid C(h)=c(h)) = \binom{m-c(h)}{x}F(h+\Delta h \mid h)^x \cdot [1-F(h+\Delta h \mid h)]^{m-c(h)-x}, \quad (11)$$

where

$$F(h+\Delta h \mid h)=\frac{F(h+\Delta h)-F(h)}{1-F(h)}.$$

Comparing Equations (7) and (11) with $F(h)$ replaced by $p(h)$, we can find the difference between these two types of SRGMs. Second the interpretation of $p(h)$ and $F(h)$ are quite different. $F(h)$ represents the failure time distribution of a fault, while $p(h)$ describes the learning process of testers.

4. Estimation for the Continuous-Time HGDM with Binomial Sensitivity Factor

Suppose that the software system has been tested up to h_n in time or equivalently nth test instance. Let c_i be the observed values of C_i or $C(h_i)$, $x_i=c_i-c_{i-1}$ and $p_i=p(h_i)$ for $i=1, 2, \cdots$. (Note that c_i, x_i and p_i are used for both discrete and continuous-time HGDMs.) Then the available test data consists of pairs (h_i, c_i) or (h_i, x_i), $i=1, 2, \cdots, n$. In order to evaluate the quality of the software system under testing, we need to estimate m and the parameters associated with p_i from the data. Most previous researches on the discrete HGDM obtained the estimates by the least squares (LS) method. However, the LS estimates are of two types. The first type of LS estimates minimizing

$$\sum_{i=1}^{n}[c_i-E(C_i))]^2 \quad (12)$$

was first considered in [21]. This criterion was also employed in Hou, Kuo and Chang [4, 5] and Jacoby and Tohma [6]. The second type of LS estimates minimizing

$$\sum_{i=1}^{n}[x_i-E(N_i \mid C_{i-1}=c_{i-1})]^2 \quad (13)$$

was suggested by Tohma et al. [23]. This is equivalent to the minimization of

$$\sum_{i=1}^{n}[c_i-E(C_i \mid C_{i-1}=c_{i-1})]^2, \quad (14)$$

The discrete HGDM assumes that the number of faults newly detected by t_i depends on c_{i-1}. At the application of t_i, c_{i-1} is already observed. Therefore the minimization of (13) or (14) is more appropriate than the minimization of (12).

The above LS criteria are generally employed when the corresponding variance, $Var(C_i)$ or $Var(N_i \mid C_{i-1}=c_{i-1})$, is constant. According to Equation (4), the conditional variance of N_i is

$$Var(N_i \mid C_{i-1}=c_{i-1})=(m-c_{i-1})p_i(1-p_i),$$

which is apparently nonhomogeneous. $[x_i-E(N_i \mid C_{i-1}=c_{i-1})]^2$ is likely to be large if $Var(N_i \mid C_{i-1}=c_{i-1})$ is large. Therefore the larger the con-

ditional variance of N_i is, the less weight should be allocated to x_i. We thus consider the weighted least squares (WLS) estimation method minimizing

$$\sum_{i=1}^{n} \frac{[x_i - E(N_i \mid C_{i-1}=c_{i-1})]^2}{Var(N_i \mid C_{i-1}=c_{i-1})}, \tag{15}$$

in which the weights are the reciprocal of the conditional variance. The WLS criterion with equal weights becomes the LS criterion. The LS and WLS criteria for the continuous-time HGDM are obtained by replacing N_i and C_{i-1} in Equations (12)~(15) with $N(h_i)=C(h_i)-C(h_{i-1})$ and $C(h_{i-1})$.

We next derive the joint distribution of $N(h_i)$, $i=1, 2, \cdots, n$.

$$\begin{aligned}
&P(N(h_i)=x_i,\ i=1,2,\cdots,n)\\
&= \prod_{i=1}^{n} P(N(h_i)=x_i \mid C(h_{i-1})=c_{i-1})\\
&= \prod_{i=1}^{n} \binom{m-c_{i-1}}{x_i} p_i^{x_i}(1-p_i)^{m-c_{i-1}-x_i}\\
&= \prod_{i=1}^{n} \binom{m-\sum_{j=1}^{i-1}x_j}{x_i} p_i^{x_i}(1-p_i)^{m-\sum_{j=1}^{i}x_j}\\
&= \binom{m}{x_1,\cdots,x_n} \cdot \prod_{i=1}^{n} p_i^{x_i} \cdot \prod_{i=1}^{n}(1-p_i)^{m}\\
&\quad \cdot \prod_{i=1}^{n}\prod_{j=i}^{n}(1-p_j)^{-x_i}\\
&= \binom{m}{x_1,\cdots,x_n} \cdot \prod_{i=1}^{n} \left[\frac{p_i}{\prod_{j=i}^{n}(1-p_j)}\right]^{x_i}\\
&\quad \cdot \left[\prod_{i=1}^{n}(1-p_i)\right]^{m}\\
&= \binom{m}{x_1,\cdots,x_n} \cdot \prod_{i=1}^{n} \left[\frac{p_i\prod_{j=1}^{i-1}(1-p_j)}{\prod_{j=i}^{n}(1-p_j)}\right]^{x_i}\\
&\quad \cdot \left[\prod_{i=1}^{n}(1-p_i)\right]^{m-\sum_{i=1}^{n}x_i}\\
&= \binom{m}{x_1,\cdots,x_n} \cdot \prod_{i=1}^{n} \left[p_i\prod_{j=1}^{i-1}(1-p_j)\right]^{x_i}\\
&\quad \cdot \left[\prod_{i=1}^{n}(1-p_i)\right]^{m-c_n}.
\end{aligned} \tag{16}$$

Noting that $\sum_{i=1}^{n} p_i \prod_{j=1}^{i-1}(1-p_j) = 1-\prod_{i=1}^{n}(1-p_i)$, we can find that $N(h_i)$, $i=1, 2, \cdots, n$ are multinomially distributed with parameters m and $p_i\prod_{j=1}^{i-1}(1-p_j)$, $i=1, 2, \cdots, n-1$. Park, Yoo and Park [14] showed that the joint distribution of N_i's for the discrete HGDM with the binomial sensitivity factor also derived as Equation (16).

Since the joint distribution of $N(h_i)$, $i=1, 2, \cdots, n$ is available, we are now able to use the ML estimation method for the HGDM. The ML method is the estimation method maximizing the (log) likelihood function. When n is sufficiently large, the ML method is generally preferred to the LS and WLS method. The LS method has been used mainly because it is mathematically easy to apply to the HGDM.

Once the estimates of m and parameters in p_i, the software reliability is obtained by substituting the parameters in Equation (9) with the estimates.

5. Application to Real Data Sets

This section applies the continuous-time HGDM with binomial sensitivity factor to two real data sets with software-usage dependent time measurement. We only consider the exponential and logistic learning factors, since the exponential and logistic curves reflect man's learning curve very well. They are respectively

$$p_i = p_{LT}\cdot(1-e^{-au_i})$$

and

$$p_i = p_{LT}\cdot\frac{1}{1+\beta e^{-au_i}}$$

for the discrete HGDM, and

$$p(h) = p_{LT}\cdot(1-e^{-ah})$$

and

$$p(h) = p_{LT}\cdot\frac{1}{1+\beta e^{-ah}}$$

for the continuous-time HGDM, where $u_i=h_i-h_{i-1}$ represents the execution time corresponding to t_i.

Two data sets are analyzed by the LS, WLS and

ML methods. The first data set is the test data of a PL/I database application program [12]. The size of the software is about 1.317 kilolines of code. The test operations in a week were regarded as a test instance. The data consists of 19 weeks of reported detected failures. The execution times and the number of observed faults are also reported. The total number of observed failures is 328. The second data sets is the failure data collected on a computer program designed to solve a navigation program and also contains the execution times and the number of detected faults [24]. The number of the cumulative failures is 107 during 15 weeks.

The discrete and continuous-time HGDMs are applied to these data sets. LS, WLS and ML estimation results were computed and summarized in <Table 1~8>. The estimates were obtained by using the nonlinear least squares procedure of SAS system. Based on the values of the sum of squares of errors (SSE), weighted SSE and log likelihood function, we can say that the continuous-time HGDM with logistic learning factor is appropriate for the two data sets. (There might be some controversies for the first data set. The (weighted) LS criterion favors the continuous-time HGDM with the logistic

〈Table 1〉 LS, WLS and ML estimates of the continuous-time HGDM with the exponential learning factor (first data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	385.7199	360.0671	371.6025
a	0.1207	0.0886	0.1099
p_{LT}	0.1139	0.1515	0.1294
SSE	1846.3934		
Weighted SSE		97.9797	
Log Likelihood function			-101.9006

〈Table 2〉 LS, WLS and ML estimates of the discrete HGDM with the exponential learning factor (first data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	497.3346	386.5658	406.0551
a	0.5976	0.1833	0.2285
p_{LT}	0.0568	0.1065	0.0920
SSE	1969.7523		
Weighted SSE		109.3967	
Log Likelihood function			-105.9766

〈Table 3〉 LS, WLS and ML estimates of the continuous-time HGDM with the logistic learning factor (first data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	331.9789	331.5892	330.0000
a	0.0569	0.0587	0.0600
β	19.6342	20.6839	17.0000
p_{LT}	1.0000	1.0000	0.9000
SSE	1530.7446		
Weighted SSE		21.6428	
Log Likelihood function			-97.7405

〈Table 4〉 LS, WLS and ML estimates of the discrete HGDM with the logistic learning factor (first data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	346.5697	343.6328	349.8553
a	0.0389	0.0510	0.0519
β	8.4621	6.1057	4.1136
p_{LT}	0.5273	0.3453	0.2762
SS	1587.6275		
Weighted SS		22.9023	
Log Likelihood function			-95.3299

〈Table 5〉 LS, WLS and ML estimates of the continuous-time HGDM with the exponential learning factor (second data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	108.8947	112.6791	111.9783
a	0.1667	0.2319	0.2257
p_{LT}	0.2626	0.2213	0.2270
SSE	109.6058		
Weighted SSE		24.2425	
Log Likelihood function			-37.4310

〈Table 6〉 LS, WLS and ML estimates of the discrete HGDM with the exponential learning factor (second data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	110.4614	114.4543	113.6587
α	0.1043	0.2017	0.1943
p_{LT}	0.2436	0.2002	0.2056
SSE	129.5066		
Weighted SSE		31.7969	
Log Likelihood function			-39.8853

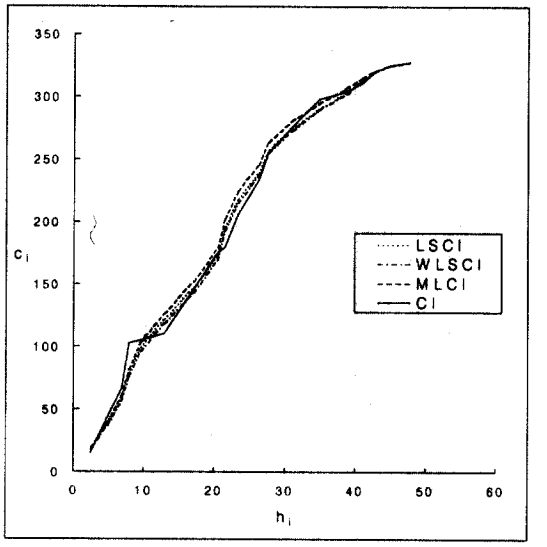
〈Table 7〉 LS, WLS and ML estimates of the continuous-time HGDM with the logistic learning factor (second data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	109.2321	111.2738	111.3619
α	0.4603	0.3311	0.3907
β	6.9413	4.4966	5.1894
p_{LT}	0.2546	0.2408	0.2371
SSE	74.8797		
Weighted SSE		5.0214	
Log Likelihood function			-33.4244

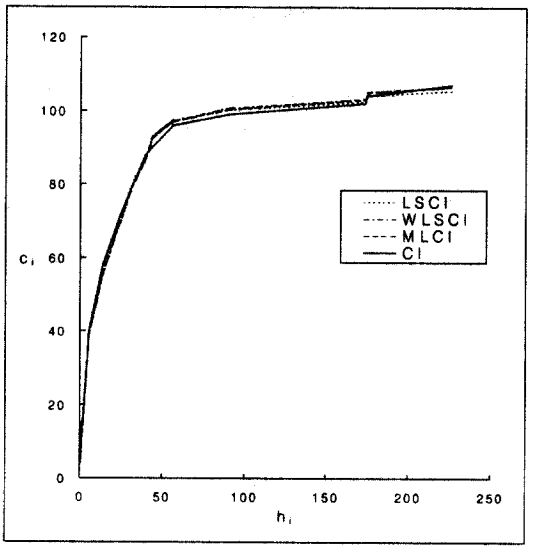
〈Table 8〉 LS, WLS and ML estimates of the discrete HGDM with the logistic learning factor (second data set)

Parameter	LS estimates	WLS estimates	ML estimates
m	110.0833	112.3437	112.0672
α	0.2087	0.1512	0.2027
β	4.6279	3.5150	4.1656
p_{LT}	0.2464	0.2312	0.2304
SSE	90.5119		
Weighted SSE		5.0348	
Log Likelihood function			-33.7644

learning factor while the ML criterion does the discrete HGDM with the logistic learning factor.) The corresponding estimates seem reasonable. One noteworthy point is that estimates are more sensitive to the model than to the estimation method. In order to show performance of the selected model, the continuous-time HGDM with the logistic learning factor, its LS, WLS and ML estimates of $E(C_i)$ and c_i



(Figure 1) Plots of c_i and LS, WLS and ML estimates of $E(C_i)$ for the continuous-time HGDM with the logistic learning factor. (first data set)



(Figure 2) Plots of c_i and LS, WLS and ML estimates of $E(C_i)$ for the continuous-time HGDM with the logistic learning factor. (second data set)

(denoted by LSCI, WLSCI, MLCI) are plotted in (Figures 1~2), respectively. The selected model works well enough for both data sets.

6. Conclusions

Most time-domain SRGMs are expressed in the execution time. However, the current discrete HGDM is not described in terms of the execution time, even when the execution time is observed and recorded during the test-and-debug phase. We discussed the drawbacks stemming from this. In order to eliminate the drawbacks, this paper developed the continuous-time HGDM. It was shown that the continuous-time HGDM retains the desirable characteristics of the discrete HGDM. Estimation problem was studied and its practical applicability has been illustrated empirically. Future research will be directed to usefulness and efficiency of the continuous-time HGDM.

References

[1] A. L. Goel and K. Okumoto, "Time-Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Trans. Rel., Vol.R-28, No.3, pp.206-211. Aug. 1979.

[2] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Optimal Release Times for Software Systems with Scheduled Delivery Time Based on the HGDM," IEEE Trans. Computers, Vol.46, pp.216-221, 1997.

[3] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Optimal Release Policy for Hyper-Geometric Distribution Software Reliability Growth Model," IEEE Trans. Reliability, Vol.45, pp.645-651, 1996.

[4] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Hyper-Geometric Distribution Software Reliability Growth Model with Imperfect Debugging," Proc. 6th Int' l Symp. Software Rel. Eng., pp.195-200, Oct. 1995.

[5] R. H. Hou, S. Y. Kuo and Y. P. Chang, "Applying Various Learning Curves to Hyper-Geometric Distribution Software Reliability Growth Model," Proc. 5th Int' l Symp. Software Rel. Eng., pp. 7-16, Nov. 1994.

[6] R. Jacoby and Y. Tohma, "Parameter Value Computation by Least Square Method and Evaluation of Software Availability and Reliability at Service-Operation by the Hyper-Geometric Distribution Software Reliability Growth Model (HGDM)," Proc. 13th Int. Conf. Software Eng., pp.226-237, 1991.

[7] R. Jacoby and Y. Tohma, "The Hyper-Geometric Distribution Software Reliability Growth Model (HGDM) : Precise Formulation and Applicability," Proc. COMPSAC90, Chicago, pp.13-19, October 1990.

[8] Z. Jelinski and P. B. Moranda, "Software Reliability Research," Statistical Computer Performance Evaluation, Academic, New York, pp.465-484, 1972.

[9] B. Littlewood, "Stochastic Reliability-Growth : A Model for Fault-Removal in Computer-Programs and Hardware-Design," IEEE Trans. on Reliability, R-30(4), pp.313-320, 1981.

[10] T. Minohara and Y. Tohma, "Parameter Estimation of Hyper-Geometric Distribution Software Reliability Growth Model by Genetic Algorithms," Proc. 6thth Int. Symp. Software Reliab. Eng., pp.324-329, 1995.

[11] J. D. Musa, A. Iannino and K. Okumoto, 'Software Reliability : Measurement, Prediction, Application,' McGraw-Hill, p.413, 1987.

[12] M. Ohba, "Software Reliability Analysis Models," IBM J. Res. Develop., Vol.28, No.4, pp.428-443, July 1984.

[13] J. Y. Park, C. Y. Yoo and B. K. Lee, "Parameter Estimation and Prediction for Hyper-Geometric Distribution Software Reliability Growth Model," Transactions of Korea Information Processing Society, Vol.5, No.9, pp.2345-2352.

[14] J. Y. Park, C. Y. Yoo and J. H. Park, "Hyper-Geometric Distribution Software Reliability Growth Model : Generalization, Estimation and Prediction,"

Transactions of Korea Information Processing Society, Vol.6, No.9, pp.2342-2349.

[15] G. J. Schick and R. W. Wolverton, "Assessment of Software Reliability," Proceedings Operations Research, Physica-Verlag, Wurzburg-Wien, pp. 395-422, 1973.

[16] G. J. Schick and R. W. Wolverton, "An Analysis of Competing Software Reliability Models," IEEE Trans. Software Eng., Vol.SE-4(2), pp. 104-120, March, 1978.

[17] J. G. Shanthikumar, "Software Reliability Models : A Review," Microelectron. Reliab., Vol.23, pp. 903-943, 1983.

[18] M. L. Shooman, "Probabilistic Models for Software Reliability Prediction," Statistical Computer Performance Evaluation, Academic, New York, pp.485-502, 1972.

[19] Y. Tohma, R. Jacoby, M. Murata, and M. Yamamoto, "Hyper-Geometric Distribution Model to Estimate the Number of Residual Software Faults," Proc. COMPSAC-89, Orlando, pp.610-617, September 1989.

[20] Y. Tohma, K. Tokunaga, "A Model for Estimating the Number of Software Faults," Inst. Electron. Commun. Eng.(IECE) Japan. Tech. Rep. FTS86-14. Sept. 1986.

[21] Y. Tohma, K. Tokunaga, S. Nagase, and Y. Murata, "Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hyper-Geometric Distribution," IEEE Trans. Software Eng., Vol.15, No.3, pp.345-355, March 1989.

[22] Y. Tohma, H. Yamano, M. Ohba, and R. Jacoby, "Parameter Estimation of the Hyper-Geometric Distribution Model for Real Test/Debug Data," Proc. Int. Symposium on Software Reliability Engineering, Austin, Texas, May 17-18, pp.28-34, 1991.

[23] Y. Tohma, H. Yamano, M. Ohba, and R. Jacoby, "The Estimation of Parameters of the Hyper-Geometric Distribution and Its Application to the Software Reliability Growth Model," IEEE Trans. Software Eng., Vol.SE-17, No.5, pp.483-489, May 1991.

[24] W. L. Wagoner, "The Final Report on a Software Reliability Measurement Study," Technol. Div., The Aerospace Corp., El Segundo, CA, Aug. 1973.

박 중 양

e-mail : parkjy@nongae.gsnu.ac.kr

1982년 연세대학교 응용통계학과(학사)

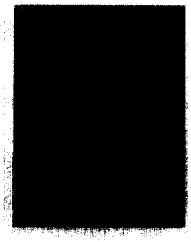
1984년 한국과학기술원 산업공학과 응용통계전공(석사)

1994년 한국과학기술원 산업공학과 응용통계전공(박사)

1984년~1989년 경상대학교 전산통계학과 교수

1989년~현재 경상대학교 통계학과 교수

관심분야 : 소프트웨어 신뢰성, 신경망, 선형 통계 모형, 실험계획법 등



김 성 희

e-mail : ksh3029@netian.com

1991년 경상대학교 수학교육과(학사)

1994년 경상대학교 대학원 전자계산학과(석사)

1996년~현재 경상대학교 대학원 전자계산학과(박사수료)

관심분야 : 소프트웨어 공학(특히, 소프트웨어 신뢰성, 소프트웨어 테스팅), 신경망 등

박 재 흥

e-mail : pjh@nongae.gsnu.ac.kr

1978년 충북대학교 수학교육과 (학사)

1980년 중앙대학교 대학원 전산학과(석사)

1988년 중앙대학교 대학원 전산학과(박사)

1983년~현재 경상대학교 컴퓨터과학과 교수

관심분야 : 소프트웨어 신뢰성, 시험도구 자동화 등