불완전 디버깅 환경에서의 이항 반응 계수 초기하분포 소프트웨어 신뢰성 성장 모델

김 성 희† · 박 중 양†† · 박 재 흥†††

요 약

지금까지의 초기하분포 소프트웨어 신뢰성 성장 모델(HGDM)에 대한 연구들은 디버깅을 할 때 새로운 결함이 생기지 않는다고 가정하고 있다. 그러나 테스트와 디버그 단계에서도 결함이 도입될 수 있으므로 이 완전 디버깅의 가정은 완화되어야 한다. 이러한 시도의 일환으로 Hou, Kuo와 Chang [7]은 불완전 디버깅 환경에서의 HGDM을 개발하였지만 학습 인자를 상수로 가정하였다. 본 논문에서는 불완전 디버깅 환경에서의 랜덤 반응 계수를 도입하고 변수 학습 인자를 허용하는 두 가지 측면에서 기존 HGDM을 수정하고 보완한다. 그리고 제안된 모델의 특징을 살펴보고 제안된 모델을 실제 자료에 적용해 본다.

The Binomial Sensitivity Factor Hyper-Geometric Distribution Software Reliability Growth Model for Imperfect Debugging Environment

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

ABSTRACT

The hyper-geometric distribution software reliability growth model (HGDM) usually assumes that all the software faults detected are perfectly removed without introducing new faults. However, since new faults can be introduced during the test-and-debug phase, the perfect debugging assumption should be relaxed. In this context, Hou, Kuo and Chang [7] developed a modified HGDM for imperfect debugging environment, assuming that the learning factor is constant. In this paper we extend the existing imperfect debugging HGDM for two respects : introduction of random sensitivity factor and allowance of variable learning factor. Then the statistical characteristics of the suggested model are studied and its applications to two real data sets are demonstrated.

1. Introduction

Recently software systems have been widely applied to the control of many complex and critical systems. Since the breakdown of computer system, caused by software faults, may result in serious damage to social life, we cannot emphasize too much the importance of achieving a high reliability in software system. Software reliability is defined in statistical term as the probability of failure-free operation of a software system for a specified period of time in a specified environment [14]. In order to quantitatively assess the reliability of a software system during the testing and operational phases, many software reliability growth models (SRGMs) have been proposed in the literatures. See, for example, Goel [2], Ramamoorthy and Bastini [19] and Shanthikumar

† 준 회 원 : 경상대학교 대학원 컴퓨터과학과
†† 정 회 원 : 경상대학교 통계학과 교수
††† 정 회 원 : 경상대학교 컴퓨터과학과 교수
논문접수 : 1999년 9월 18일, 심사완료 : 2000년 3월 11일

[20]. The SRGMs are usually used to estimate the number of remaining faults, software reliability and other software quality assessment measures. This paper considers the hyper-geometric distribution software reliability growth model (HGDM), which was advocated by Tohma and Tokunaga [25]. The HGDM has been shown to be efficient for estimating the number of initial faults resident in a software system at the beginning of the test-and-debug phase. It has been successfully applied to real data sets. A series of studies on the HGDM has been made recently by Hou, Kuo and Chang [5-8], Jacoby and Tohma [9-10], Minohara and Tohma [13], Tohma et al. [20-24], Park, Kim and Park [16], Park, Yoo and Lee [17] and Park, Yoo and Park [18].

To make the existing HGDM more realistic, the *perfect debugging assumption has need to be relaxed.* This is because debugging actions during the test-and-debug phase are not always performed perfectly. Due to the complexity of the software systems and the incomplete understanding of the software requirements/specification/structure, the testing team may not able to remove the faults perfectly. So, Hou, Kuo and Chang [7] proposed the HGDM with imperfect debugging. However, they did not take into account the progress in testing. In this paper we propose an extended model based on the binomial sensitivity factor HGDM incorporating the notion of imperfect debugging, which will be called the binomial sensitivity factor HGDM for imperfect debugging environment. 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 and random sensitivity factor are briefly reviewed in Section 2. Section 3 develops and characterizes the binomial sensitivity factor HGDM for imperfect debugging environment. The parameter estimation problem is considered in Section 4. Section 5 gives the illustrative examples and results. The suggested model is compared to the imperfect debugging HGDM proposed by Hou, Kuo and Chang [7]. And conclusions are presented in Section 6.

2. Review of HGDM

In this section, we concisely review the basic concept and formulation of HGDM. A software system is assumed to have m faults initially when the test-and-debug phase starts. Test operations performed for a given period, in a day or a week, are called a test instance. Test instances are denoted by t_i, $i = 1, 2, \cdots, n$ in accordance with the order of applying them. The sensitivity factor w_i represents how many faults are newly discovered or rediscovered during the application of t_i. Some of the faults sensed by t_i may have been sensed previously by the application of test instances t_j, $j = 1, 2, \cdots, i-1$.

Let N_i be the number of faults newly detected by t_i and $C_i = \sum_{j=1}^{i} N_j$. 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, i.e., $w_i = mp_i$,

where $0 \le p_i \le 1$, 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

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

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 conditional expected value of N_i is obtained as

$$E(N_i \mid C_{i-1}) = (m - C_{i-1})p_i. \quad (2)$$

The mean value function, the expected value of C_t, was obtained by Jacoby and Tohma [9] as

$$E(C_t) = m[1 - \prod_{j=1}^{t}(1-p_j)]. \qquad (3)$$

The sensitivity factor w_i plays a key role in HGDM. Various plausible deterministic functions for w_i, equivalently p_i, have been devised and successfully applied to real data sets [9, 13] Recently Hou, Kuo and Chang [8] suggested the exponential and logistic learning factors based on the exponential and logistic learning curves

Suppose that F_i represents the set of faults detected by t_i. The HGDM assumes that $|F_i|$ is deterministic, where $|\cdot|$ is the cardinality of a set, but the elements of F_i are randomly chosen from the initial m faults. This assumption does not reflect enough the random behavior of testing process. 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. Such an attempt was made by Park, Yoo and Park [18]

They assumed that W_i is a binomial random variable with parameters m and p_i, that is, for $w_i = 0, 1, \cdots, m$

$$P(W_i = w_i) = \binom{m}{w_i} p_i^{w_i}(1-p_i)^{m-w_i}. \qquad (4)$$

They obtained the following equations,

$$P(C_t = c_t) = \binom{m}{c_t}\left[1 - \prod_{j=1}^{t}(1-p_j)\right]^{c_t}\left[\prod_{j=1}^{t}(1-p_j)\right]^{m-c_t},$$

$$P(N_t = x_t \mid C_{t-1}) = \binom{m - C_{t-1}}{x_t} p_t^{x_t}(1-p_t)^{m-C_{t-1}-x_t},$$

and

$$P(N_i = x_i,\ i = 1, 2, \cdots, n) = \binom{m}{x_1, \cdots, x_n}\prod_{i=1}^{n}\left[p_i\prod_{j=1}^{i-1}(1-p_j)\right]^{x_i} \cdot \left[\prod_{i=1}^{n}(1-p_i)\right]^{m-\sum_{i=1}^{n}x_i}.$$

It is not difficult to verify that $E(N_t \mid C_{t-1})$ and $E(C_t)$ for the binomial sensitivity factor HGDM are also obtained as Equations (2) and (3). They further showed that if the least squares method was employed, the estimation and prediction results of the binomial sensitivity factor HGDM are identical with those of the deterministic sensitivity factor 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.

3. Binomial Sensitivity Factor HGDM for Imperfect Debugging Environment

Most SRGMs including HGDM assume that debugging is perfect, that is, a fault is completely removed after it is detected. This implies that no new faults are introduced when a fault is removed. This assumption significantly contributes to the simplicity of SRGMs. The perfect debugging assumption does not usually hold for most practical projects. In order to make existing SRGMs more realistic, this perfect debugging assumption need to be relaxed It is therefore necessary to develop SRGMs in which the faults detected by testing are not always corrected or removed Such imperfect debugging SRGMs are expected to estimate reliability assessment measures more accurately SRGMs taking account of imperfect debugging were considered in several literature [1, 3, 4, 12, 26]. The HGDM for imperfect debugging environment was first proposed by Hou, Kuo and Chang [7]. The primary aim of this paper is to extend the HGDM with imperfect debugging proposed by Hou, Kuo and Chang [7] by introducing the binomial sensitivity factor and more general learning factor. Hou, Kuo and Chang [7] modified Assumption (A1) to :

(A1.1)′ When the detected faults are removed, it is possible to introduce new faults

(A1.2)′ When a fault is newly discovered during the application of t_i, removal of the fault is instantaneous and the following may occur :

(a) the fault is corrected with probability $1-\theta_i$;

(b) a new fault is introduced with probability θ_i.

The parameter θ_i is referred to the fault introduction rate. Let m_i be the expected number of faults including the initial faults and all the faults introduced so far by t_j, $j=1, 2, \cdots, i-1$. Then $m_i = m + \sum_{j=1}^{i-1} \theta_j N_j$ and Equation (1) is modified to

$$P(N_i = x_i \mid C_{i-1}) = \frac{\binom{m_{i-1} - C_{i-1}}{x_i}\binom{C_{i-1}}{w_i - x_i}}{\binom{m_{i-1}}{w_i}}, \quad (5)$$

where $\max(0, w_i - C_{i-1}) \le x_i \le \min(w_i, m_{i-1} - C_{i-1})$. Further assuming that the learning factor p_i is constant, i. e., $p_i = p$ for all i, they obtained the mean value function as $E(C_1) = mp$ and

$$E(C_i) = mp\left[1 + \sum_{j=2}^{i} \prod_{k=1}^{j-1} \{1-(1-\theta_k)p\}\right] \quad (6)$$

for $i = 2, \cdots, n$.

Let us discuss drawbacks of the above imperfect debugging HGDM. First, it assumes that the learning factor is constant. In order to model the human learning process, the learning factor should vary as the testing proceeds. Second, m_{i-1}, the expected number of total faults, is used in Equation (5). Due to Assumption (A1.2)′ the number of total faults should be regarded as a random variable. Henceforth, we will treat the number of faults as a random variable and denote it by M_i. Since M_{i-1} and C_{i-1} are realized at the application of ith test instance and the number of faults sensed by ith test instance depends on only M_{i-1}, we further assume that the sensitivity factor W_i is a binomial random variable with parameters M_{i-1} and p_i on condition that M_{i-1} and C_{i-1} are given. That is,

$$P(W_i = w_i \mid M_{i-1}, C_{i-1}) = \binom{M_{i-1}}{w_i} p_i^{w_i} (1-p_i)^{M_{i-1}-w_i}. \quad (7)$$

N_i newly detected faults of W_i sensed faults are subjected to debugging. If C_{i-1}, M_{i-1} and W_i are given, the probability that N_i faults are newly detected is

$$P(N_i = x_i \mid M_{i-1}, C_{i-1}, W_i) = \frac{\binom{M_{i-1} - C_{i-1}}{x_i}\binom{C_{i-1}}{W_i - x_i}}{\binom{M_{i-1}}{W_i}}. \quad (8)$$

Let R_i be the number of faults introduced into the software system during debugging of N_i newly detected faults. According to Assumption (A1.2)′, R_i is a binomial random variable with parameters N_i and θ_i, i. e.,

$$P(R_i = r_i \mid N_i) = \binom{N_i}{r_i} \theta_i^{r_i} (1-\theta_i)^{N_i - r_i}. \quad (9)$$

Let L_i be the number of faults successfully corrected during debugging of N_i newly detected faults. Then

$$C_{i-1} = \sum_{j=1}^{i-1} N_j = \sum_{j=1}^{i-1} R_j + \sum_{j=1}^{i-1} L_j$$

and

$$M_{i-1} = m + \sum_{j=1}^{i-1} R_j$$

We first derive the conditional distribution of N_i given C_{i-1} and M_{i-1}. Multiplying Equation (7) and (8),

$$\begin{aligned} P(N_i = x_i, W_i = w_i \mid M_{i-1}, C_{i-1}) &= \binom{M_{i-1} - C_{i-1}}{x_i}\binom{C_{i-1}}{w_i - x_i} p_i^{w_i}(1-p_i)^{M_{i-1}-w_i} \\ &= \binom{M_{i-1} - C_{i-1}}{x_i} p_i^{x_i}(1-p_i)^{M_{i-1}-C_{i-1}-x_i} \\ &\quad \cdot \binom{C_{i-1}}{w_i - x_i} p_i^{w_i - x_i}(1-p_i)^{C_{i-1}-w_i-x_i} \end{aligned} \quad (10)$$

where $0 \le x_i \le M_{i-1} - C_{i-1}$ and $n_i \le w_i \le C_{i-1} + x_i$. Therefore, by summing (10) over w_i,

$$P(N_i = x_i \mid M_{i-1}, C_{i-1}) = \binom{M_{i-1} - C_{i-1}}{x_i} p_i^{x_i} (1-p_i)^{M_{i-1} - C_{i-1} - x_i}. \quad (11)$$

This is a binomial distribution with parameters $(M_{i-1} - C_{i-1})$ and p_i. Multiplying Equations (9) and (11),

$$P(R_i = r_i, N_i = x_i \mid M_{i-1}, C_{i-1}) = \frac{(M_{i-1} - C_{i-1})!}{r_i!(x_i - r_i)!(M_i - C_{i-1} - x_i)!} (p_i \theta_i)^{r_i} \cdot [(p_i(1-\theta_i)]^{x_i - r_i} (1-p_i)^{M_{i-1} - C_{i-1} - x_i}. \quad (12)$$

Since $N_i = R_i + L_i$,

$$P(R_i = r_i, L_i = l_i \mid M_{i-1}, C_{i-1}) = \frac{(M_{i-1} - C_{i-1})!}{r_i! \, l_i! \, (M_{i-1} - C_{i-1} - r_i - l_i)!} (p_i \theta_i)^{r_i} \cdot [(p_i(1-\theta_i)]^{l_i} (1-p_i)^{M_{i-1} - C_{i-1} - r_i - l_i}. \quad (13)$$

Thus R_i is binomially distributed with parameters $M_{i-1} - C_{i-1}$ and $p_i \theta_i$. And L_i is binomially distributed with parameters $M_{i-1} - C_{i-1}$ and $p_i(1-\theta_i)$. These distribution results enable us to have the mean value function. Since $M_i - C_i = (M_{i-1} - C_{i-1}) - L_i$,

$$\begin{aligned} E(M_i - C_i) &= E(M_{i-1} - C_{i-1}) - E[E(L_i \mid M_{i-1}, C_{i-1})] \\ &= E(M_{i-1} - C_{i-1}) - E[(M_{i-1} - C_{i-1}) p_i (1-\theta_i)] \\ &= E(M_{i-1} - C_{i-1}) \{1 - p_i(1-\theta_i)\}. \end{aligned} \quad (14)$$

The solution to the difference equation (14) is

$$E(M_i - C_i) = m \prod_{k=1}^{i} \{1 - p_i(1-\theta_i)\} \quad (15)$$

for $i = 1, 2, \cdots$. Since $C_i = C_{i-1} + N_i$,

$$\begin{aligned} E(C_i) &= E(C_{i-1}) + E[E(N_i \mid M_{i-1}, C_{i-1})] \\ &= E(C_{i-1}) + E[(M_{i-1} - C_{i-1}) p_i] \\ &= E(C_{i-1}) + m \prod_{k=1}^{i-1} \{1 - p_k(1-\theta_k)\} p_i \\ &= E(C_{i-1}) + p_i \cdot m \prod_{k=1}^{i-1} \{1 - p_k(1-\theta_k)\}. \end{aligned} \quad (16)$$

The solution to the difference equation (16) is obtained as $E(C_1) = mp_1$ and

$$E(C_i) = m\left[p_1 + \sum_{j=2}^{i} p_i \cdot \prod_{k=1}^{j-1} \{1 - p_k(1-\theta_k)\}\right] \quad (17)$$

for $i = 2, \cdots, n$. Also $E(N_i)$ is obtained from Equation (16) as

$$E(N_i) = p_i \cdot m \prod_{k=1}^{i-1} \{1 - p_k(1-\theta_k)\}. \quad (18)$$

In general, p_i tends to increase as the testing proceeds while θ_i tends to decrease in the overall viewpoints. We thus suppose that p_i increases in i and θ_i decreases in i. Then

$$\begin{aligned} \lim_{i \to \infty} E(C_i) &= \lim_{i \to \infty} m\left[p_1 + \sum_{j=2}^{i} p_j \prod_{k=1}^{j-1} \{1 - p_k(1-\theta_k)\}\right] \\ &\le \lim_{i \to \infty} m\left[1 + \sum_{j=2}^{i} \prod_{k=1}^{j-1} \{1 - p_1(1-\theta_1)\}\right] \\ &= \lim_{i \to \infty} m\left[\sum_{j=1}^{i} \{1 - p_1(1-\theta_1)\}^{j-1}\right] \\ &= \frac{m}{p_1(1-\theta_1)} \end{aligned}$$

and

$$\begin{aligned} \lim_{i \to \infty} E(M_i - C_i) &= \lim_{i \to \infty} m \prod_{k=1}^{i} \{1 - p_k(1-\theta_k)\} \\ &\le \lim_{i \to \infty} m \prod_{k=1}^{i} \{1 - p_1(1-\theta_1)\} \\ &= \lim_{i \to \infty} m\{1 - p_1(1-\theta_1)\}^i \\ &= 0. \end{aligned}$$

Therefore,

$$\lim_{i \to \infty} E(M_i) = \lim_{i \to \infty} E(C_i) \le \frac{m}{p_1(1-\theta_1)}.$$

This implies that all the faults will be ultimately detected and removed. Since $\lim_{i \to \infty} E(M_i)$ is bounded above, the suggested imperfect debugging HGDM belongs to the finite failures category model in classification scheme by Musa and Okumoto [15].

4. Parameter Estimation

Suppose that the software system has been tested up to nth test instance t_n. Let c_i be the observed values of C_i and $x_i = c_i - c_{i-1}$ for $i = 1, 2, \cdots$. In order to evaluate the quality of the software system under testing, we need to estimate m and the parameters associated with p_i and θ_i from the data.

The parameters have been estimated the least squares (LS) method in the previous researches on the HGDM. However, specific criteria of the LS method can be further classified into two types. The first type is the minimization of

$$\sum_{i=1}^{n} [c_i - E(C_i)]^2, \tag{19}$$

which was first considered by [23]. This criterion was also employed in Hou, Kuo and Chang [7, 8] and Jacoby and Tohma [9]. A variant of this criterion is the minimization of $\sum_{i=1}^{n} [x_i - E(N_i)]^2$. The second type is the minimization of

$$\sum_{i=1}^{n} [x_i - E(N_i | C_{i-1})]^2, \tag{20}$$

which was suggested by Tohma et al. [25]. This is equivalent to the minimization of

$$\sum_{i=1}^{n} [c_i - E(C_i | C_{i-1})]^2 \tag{21}$$

At the application of t_i, c_{i-1} is already observed. Therefore the minimization of (20) or (21) is more appropriate than the minimization of (19).

We should note that $Var(C_i)$ and $Var(C_i \mid C_{i-1})$, for $i = 1, 2, \cdots$, are not constant. In this case the weighted least squares (WLS) method is to be used. Recently Park, Yoo and Lee [17] suggested the WLS method for estimating parameters of the HGDM. They also proposed the ML method. However, the WLS method and ML method are not applicable to the model suggested in Section 3. This is because $Var(C_i)$ and the joint distribution of N_i's are not available. We thus use the LS method minimizing (20) in Section 5.

5. Numerical Examples

In this section, numerical examples are given to illustrate how to apply the suggested model to real data sets. We assume that the learning factor and the fault introduction rate are increasing and decreasing logistic functions respectively. This is because the logistic curve reflects the human learning process very well. They are respectively written as

$$p_i = \frac{1}{1 + e^{-ai+b}}$$

and

$$\theta_i = \frac{1}{1 + e^{\alpha i + \beta}},$$

where $a > 0$ and $\alpha > 0$.

To check the validity of the suggested model, it is tested on two software failure data sets. The first data set is the test-and-debug data set of a software system [11]. Since the test data is reported per week, a test instance is a week of observation. It is the collection of the cumulative number of discovered faults for the 81 test instances. The cumulative number of discovered faults up to the test instance t_{81} is 461. The second data set is from Tohma et al. [21]. The number of the cumulative failures is 481 during 111 test instances. The cumulative test time is reported in days.

The LS estimates of parameters in the imperfect debugging HGDM proposed by Huo, Kuo and Chang [7] and the model suggested in this paper are given in <Table 1-2>. The parameter p in <Table 1> is the constant learning factor in the fault detection process. The estimates were obtained by using the nonlinear least squares procedure of SAS system. Based on the value of the MSE, we can say that the first data set favors the imperfect debugging HGDM of Huo, Kuo and Chang [7] while the second data set does our model. Since MSE values for the two models are not significantly different, we can conclude that the suggested model fits as well as the model by Huo, Kuo and Chang [7]. In order to show fitness of the model, LS estimates of $E(C_i)$ (denoted by LSCI) and c_i are plotted in (Fig. 1) and (Fig. 2) respectively.

〈Table 1〉 LS estimates of the imperfect debugging HGDM of Huo, Kuo and Chang

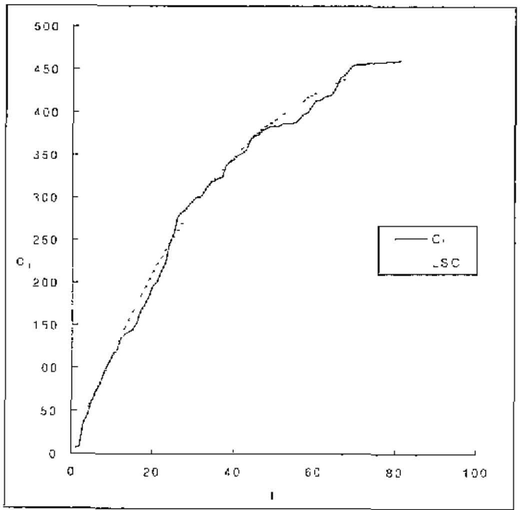
Parameter	LS estimates (first data set)	LS estimates (second data set)
m	366.5000	361.1629
p	0.0299	0.0295
α	0.2188	3.7049
β	-3.0503	-52.6521
SSE	1645.1070	3853.9422
MSE	21.3650	36.0182

〈Table 2〉 LS estimates of the binomial sensitivity factor HGDM with for imperfect debugging environment

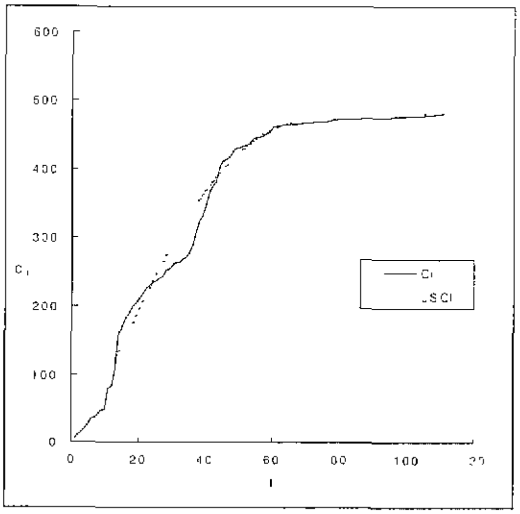
Parameter	LS estimates (first data set)	LS estimates (second data set)
m	370.5225	369.2843
a	0.0044	0.0263
b	3.5073	3.8831
α	0.1760	3.7049
β	-2.0365	-52.6521
SSE	1644.9345	3654.6646
MSE	21.6439	34.4780

6. Conclusions

In this paper, we proposed a generalized model based on the binomial sensitivity factor HGDM by relaxing the assumption that the detected faults in a program can be perfectly removed. We also consider the situation where there is learning implicitly both in the fault detection and removal process. Estimation problem was studied and its practical application has been illustrated empirically. The suggested model provides reasonable fit to real data sets. Future research will be directed to the development of the continuous-time HGDM for imperfect debugging environment.



(Fig. 1) Plots of c_i and LS estimates of $E(C_i)$ (for the first data set)



(Fig. 2) Plots of c_i and LS estimates of $E(C_i)$ (for the second data set)

References

[1] S. R. Dalal and A. A. McIntosh. "When to Stop Testing for Large Software Systems with Changing Code," IEEE Trans. Software Eng., Vol.20, No.4, pp.318-323, 1994.

[2] A. L. Goel, "Software Reliability Models : Assump-

tions, Limitations, and Applicability," IEEE Trans. Software Eng., Vol.SE-11, pp.1411-1423, 1985.

[3] A. L. Goel and K. Okumoto, "An Analysis of Recurrent Software Failures in A Real-time Control System," in Proc. ACM Annu. Tech. Conf., ACM. Washington. DC. pp.496-500, 1978.

[4] 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.

[5] 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.

[6] 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

[7] 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.

[8] 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.

[9] 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

[10] 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

[11] K. Kanoun, M. R. Bastos Martini and J. Moreira de Souza, "A Method for Software Reliability Analysis and Prediction. Application to the Tropoco-R Switching System," Research report from LAAS-CNRS, France, 1989.

[12] P. K. Kapur and S. Younes, "Modelling An Imperfect Debugging Phenomenon in Software Reliability," Microelectron. Reliab., Vol.36, No.5, pp. 645-650, 1996

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

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

[15] J. D. Musa and K. Okumoto. "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," Proc. 7th Int. Conf. Software Eng., pp.230-238, 1984.

[16] J. Y. Park, S. H. Kim and J. H. Park, "Development of the Continuous-Time HGDM with Binomial Sensitivity Factor," Transactions of Korea Information Processing Society, Vol.6, No.12, pp.3490-3499.

[17] 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

[18] 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.2343-2349.

[19] C. V. Ramamoorthy and F. B. Bastani, "Software Reliability-Status and Perspectives," IEEE Trans. Software Eng., Vol.SE-8, pp.354-371, 1982.

[20] J. G. Shanthikumar, "Software Reliability Models

A Review," Microelectron. Reliab., 5 Vol.23, pp.903-943, 1983.

[21] 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

[22] 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.

[23] 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

[24] 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

[25] 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.

[26] G. Xia, P. Zeephongsekul and S. Kumar, "Optimal Software Release Policies with Learning Factor for Imperfect Debugging," Microelectronics & Reliability, Vol.33, pp.81-86, 1993.

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 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

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