数理解析研究所講究録 1306 巻 2003 年 152-161

Estimating Discrete-Time Periodic Software Rejuvenation Schedules under Cost Effectiveness Criterion

岩本 一樹 †, 土肥 正 †, 海生 直人 ‡
K. Iwamoto†, T. Dohi† and N. Kaio‡

† 広島大学大学院工学研究科情報工学専攻
‡ 広島修道大学経済科学部経営情報学科

†Graduate School of Engineering, Hiroshima University, Japan
‡Faculty of Economic Sciences, Hiroshima Shudo University, Japan

1. INTRODUCTION

Software faults should ideally have been removed during the debugging phase. Even if software may have been thoroughly tested, it still may have some design faults that are yet to be revealed. Such faults are called *bohrbugs* and may exist even in mature software such as commercial operating systems. Also, even mature software can be expected to have what are known as *heisenbugs* [11]. These are bugs in the software that are revealed only during specific collusions of events. For example, a sequence of operations may leave the software in a state that results in an error on an operation executed next. Simply retrying a failed operation, or if the application process has crashed, restarting the process might resolve the problem. Another type of fault observed in software systems is due to the phenomenon of resource exhaustion. Operating system resources such as swap space and free memory available are progressively depleted due to defects in software such as memory leaks and incomplete cleanup of resources after use. These faults may exist in the operating system, middleware and the application software.

In fact, when software application executes continuously for long periods of time, some of the faults cause software to age due to the error conditions that accrue with time and/or load. *Software aging* will affect the performance of the application and eventually cause it to fail [1, 4, 8, 16]. Software aging has also been observed in widely-used software like Internet Explorer, Netscape and xrn as well as commercial operating systems and middleware. A complementary approach to handle software aging and its related transient software failures, called *software rejuvenation*, are becoming popular [12]. Software rejuvenation is a preventive and proactive solution that is particularly useful for counteracting the phenomenon of software aging. It involves stopping the running software occasionally, cleaning its internal state and restarting it. Cleaning the internal state of a software might involve garbage collection, flushing operating system kernel tables, reinitializing internal data structures, and hardware reboot.

Huang *et al.* [12] report the software aging phenomenon in real telecommunications billing application where over time the application experiences a crash or a hang failure, and propose to perform rejuvenation occasionally or periodically. More specifically, they consider the degradation as a two step process. From the clean state the software system jumps into a degraded state from which two actions are possible: rejuvenation with return to the clean state or transition to the complete failure state. They model the four-state process as a continuous-time Markov chain and derive the steady-state availability and the expected cost per unit time in the steady state. Avritzer and Weyuker [2] discuss aging in a telecommunication switching software where the effect manifests as gradual performance degradation. Garg *et al.* [9] introduce the idea of periodic rejuvenation (deterministic interval between successive rejuvenations) into the Huang *et al.* model [12] and represent the stochastic behavior by using a Markov regenerative stochastic Petri net. Dohi *et al.* [5] extend the original Huang *et al.* model to semi-Markov models and develop a non-parametric algorithm to estimate the optimal software rejuvenation schedule from the complete sample of failure time.

As another examples, it is interesting to consider both effects of aging as crash/hang failure, referred to as hard failures, and of aging as soft failures that can lead to performance degradation. Pfening *et al.* [18] model the performance degradation process by the gradual decrease of the service rate and formulate the determination problem of the optimal software rejuvenation schedule and formulate the determination problem of the optimal software rejuvenation schedule by a Markov decision process. Garg *et al.* [10] consider the transaction based software systems, which involve arrival and queueing of jobs, and analyze both effects of aging; hard failures that result in an unavailability and soft failures that result in performance degradation. Bobbio *et al.* [3] present a fine grained software rejuvenation model with the degradation process consisting of a sequence of additive random shocks. Liu *et al.* [14] model a cable modem termination system with rejuvenation by stochastic Petri nets. Park and Kim [17] carry out the availability analysis for active/standby cluster systems with rejuvenation.

This paper treats the similar periodic software rejuvenation model to Garg *et al.* [9] under the different operation circumstance. That is, we model the stochastic behavior of telecommunication billing applications by using a discrete-time Markov regenerative process, and determine the optimal periodic software rejuvenation schedule in discrete-time setting. Recently, Dohi *et al.* [6, 7] reconsider the semi-Markov models [5] in discrete time and characterize the optimal non-periodic software rejuvenation schedules minimizing and maximizing the long-run average cost and the steady-state system availability, respectively. Also, they develop non-parametric algorithms to estimate the optimal software rejuvenation schedules, based upon the discrete total time on test (DTTT) concept. Iwamoto *et al.* [13] introduce the cost effectiveness as a criterion of optimality and obtain the optimal non-periodic software rejuvenation policy in discrete time. Here, we derive the optimal periodic software rejuvenation schedule which maximizes the cost effectiveness in discrete-time setting, and provide a statistical estimation method based on the similar DTTT concept.

2. MODEL DESCRIPTION

2.1 Notation and Assumption

Z: time interval from highly robust state to failure probable state (discrete random variable)

$F_0(n)$, $f_0(n)$, μ_0 (> 0): cdf, pmf and mean of Z, where $n = 0, 1, 2, \cdots$.

X: failure time from failure probable state (discrete random variable)

$F_f(n)$, $f_f(n)$, μ_f (> 0): cdf, pmf and mean of X

' $*$ ': discrete convolution operator, *i.e.* $F_0 * F_f(n) = \sum_{j=0}^{n} F_0(n-j) f_f(j) = \sum_{j=0}^{n} F_f(n-j) f_0(j)$

$\overline{\psi}(\cdot)$: survivor function $(= 1 - \psi(\cdot))$

$r_{0f}(n)$: $= f_0 * f_f(n) / \overline{F_0 * F_f}(n-1)$

Y: recovery time from failure state (discrete random variable)

$F_a(n)$, $f_a(n)$, μ_a (> 0): cdf, pmf and mean of Y

N: rejuvenation time from failure probable state (discrete random variable)

$F(n)$, n_0 (≥ 0): cdf and mean of N

R: system overhead incurred by software rejuvenation (discrete random variable)

$F_c(n)$, $f_c(n)$, μ_c (> 0): cdf, pmf and mean of R

c_s: recovery cost from system failure per unit time

c_p: rejuvenation cost per unit time

Assumption: $c_s\mu_a > c_p\mu_c$.

2.2 Model 1

Suppose that the software system is started for operation at time $n = 0$ and is in the highly robust state (normal operation state). Let Z be the random time to reach the failure-probable state from the highly robust state. Let $\Pr\{Z \le n\} = F_0(n)$, $(n = 0, 1, 2, \cdots)$. Just after the state becomes the failure-probable state, a system failure may occur with a positive probability. Let X be the time to failure from the failure-probable state having the probability distribution function $\Pr\{X \le n\} = F_f(n)$. If the failure occurs before triggering a software rejuvenation, then the recovery operation is started immediately. The time to complete the recovery operation Y is also the positive random variable having the probability distribution function $\Pr\{Y \le n\} = F_a(n)$. Without any loss of generality, it is assumed that after completing repair the system becomes as good as new.

On the other hand, rejuvenation is performed at a random time interval measured from the start (or restart) of the software in the robust state. The probability distribution function of the time to invoke the software rejuvenation, say N, and the probability distribution function of the time to complete software rejuvenation are represented by $F(n)$ and $F_c(n)$, respectively. Suppose that the time to rejuvenate the software is a constant n_0, *i.e.* the software rejuvenation is performed periodically. Then, the probabilitu distribution function $F(n)$ has to be replaced by

$$F(n) = U(n - n_0) = \begin{cases} 1: & (n \ge n_0) \\ 0: & (n < n_0), \end{cases} \tag{1}$$

where $U(\cdot)$ is the unit step function. We call n_0 (≥ 0) *the software rejuvenative schedule* in this paper. After completing the software rejuvenation, the software system becomes as good as new, and the software age is initiated at the beginning of the next highly robust state. We call the above model *Model 1* in this paper. Figure 1 illustrates the configuration of Model 1.

2.3 Model 2

Next, consider the other model. When the recovery operation is completed after the system failure, it is assumed in Model 1 that the system is renewed and the state is moved to the highly robust state. However, since restarting the system after recovery operation may require some cleanup and resuming the process execution at the checkpointed state, a software rejuvenation after completing recovery operation will be needed to renew the system [6, 7]. In this case, the correponding stochastic model should be distinguished from Model 1. We call this model in which the software rejuvenation is performed just after the completion of recovery operation as well as at the prespecified time after the robust state is entered, Model 2. Figure 2 depicts the configuration of Model 2.

3. COST EFFECTIVENESS ANALYSIS

Define the time length from the beginning of the system operation to the completion of the preventive or corrective maintenance as one cycle, and suppose that the same cycle

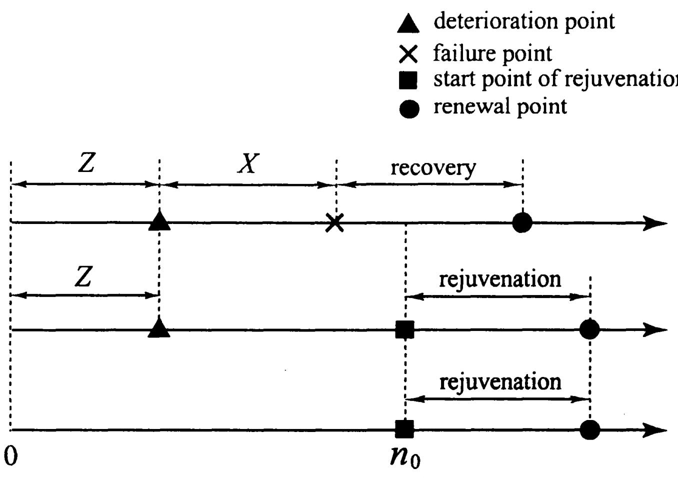

Figure 1: Configuration of Model 1.

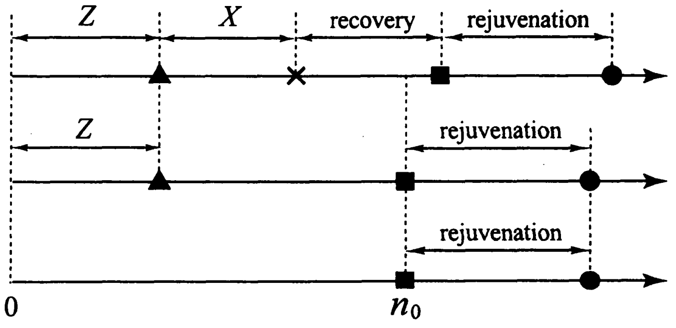

Figure 2: Configuration of Model 2.

is repeated again and again over an infinite time horizon. Then, the mean operative time during one cycle in Model 1 is given by

$$S_1(n_0) = \sum_{n=0}^{n_0-1} \overline{F_f * F_0}(n). \tag{2}$$

The total expected cost for one cycle is obtained as

$$V_1(n_0) = c_s\mu_a F_f * F_0(n_0) + c_p\mu_c \overline{F}_f * F_0(n_0). \tag{3}$$

Then the cost effectiveness for Model 1 is defined as the mean operative time per unit expected cost [13] as

$$E_1(n_0) = S_1(n_0)/V_1(n_0) \tag{4}$$

and the problem is to seek the optimal software rejuvenation schedule n_0^* maximizing it.

Taking the difference of $E_1(n_0)$ with respect to n_0, define the following function [13, 15]:

$$\begin{aligned} q_1(n_0) &= \frac{V_1(n_0)V_1(n_0+1)[E_1(n_0+1) - E_1(n_0)]}{\overline{F_0 * F_f}(n_0)} \\ &= V_1(n_0) - (c_s\mu_a - c_p\mu_c)S_1(n_0)r_{0f}(n_0+1), \end{aligned} \tag{5}$$

where $f_f * f_0(n)$ is the pmf of the probability distribution $F_f * F_0(n)$.

The following result gives the optimal software rejuvenation schedule for Model 1.

Theorem 1: For Model 1, (1) suppose that the probability distribution $F_0 * F_f(n)$ is strictly IFR (increasing failure rate) under the assumption $c_s\mu_a > c_p\mu_c$.

(i) If $q_1(\infty) < 0$, then there exist (at least one, at most two) optimal software rejuvenation schedules n_0^* $(0 < n_0^* < \infty)$ satisfying $q_1(n_0^* - 1) > 0$ and $q_1(n_0^*) \le 0$. Then, the maximum cost effectiveness is given by

$$\underline{E}_1(n_0^*) \le E_1(n_0^*) < \overline{E}_1(n_0^*), \tag{6}$$

where

$$\underline{E}_1(n_0^*) = \frac{1}{(c_s\mu_a - c_p\mu_c)r(n_0^* + 1)}, \tag{7}$$

$$\overline{E}_1(n_0^*) = \frac{1}{(c_s\mu_a - c_p\mu_c)r(n_0^*)}. \tag{8}$$

(ii) If $q(\infty) \ge 0$, then the optimal software rejuvenation schedule becomes $n_0^* \to \infty$, *i.e.* it is optimal not to carry out the software rejuvenation. Then, the maximum cost effectiveness is given by

$$E_1(\infty) = \frac{\mu_f + \mu_0}{c_s\mu_a}. \tag{9}$$

(2) Suppose that the probability distribution $F_0 * F_f(n)$ is DFR (decreasing failure rate) under the assumption $c_s\mu_a > c_p\mu_c$. Then, the cost effectiveness $E_1(n_0)$ is a convex function of n_0, and the optimal software rejuvenation schedule is $n_0^* = 0$ or $n_0^* \to \infty$.

Next, consider Model 2. The mean length of operative time for one cycle and the total expected cost during one cycle are given by

$$S_2(n_0) = \sum_{n=0}^{n_0-1} \overline{F_f * F_0}(n) \tag{10}$$

and

$$V_2(n_0) = c_s\mu_a F_f * F_0(n_0) + c_p\mu_c, \tag{11}$$

respectively. Then, the cost effectiveness for Model 2 is formulated as

$$E_2(n_0) = S_2(n_0)/V_2(n_0). \tag{12}$$

In a fashion similar to Model 1, define the following function:

$$q_2(n_0) = V_2(n_0) - c_s\mu_a r_{0f}(n_0 + 1)S_2(n_0). \tag{13}$$

Theorem 2: For Model 2, (1) suppose that the probability distribution $F_0 * F_f(n)$ is strictly IFR.

(i) If $q_2(\infty) < 0$, then there exist (at least one, at most two) optimal software rejuvenation schedules n_0^* $(0 < n_0^* < \infty)$ satisfying $q_2(n_0^* - 1) > 0$ and $q_2(n_0^*) \le 0$. Then, the maximum cost effetiveness is given by

$$\underline{E}_2(n_0^*) \le E_2(n_0^*) < \overline{E}_2(n_0^*), \tag{14}$$

where

$$\underline{E}_2(n_0^*) = \frac{1}{c_s\mu_a r(n_0^* + 1)}, \tag{15}$$

$$\overline{E}_2(n_0^*) = \frac{1}{c_s\mu_a r(n_0^*)}. \tag{16}$$

(ii) If $q_2(\infty) \geq 0$, then the optimal software rejuvenation schedule becomes $n_0^* \to \infty$, and the maximum cost effectiveness is given by

$$E_2(\infty) = \frac{\mu_f + \mu_0}{c_p\mu_c + c_s\mu_a}. \tag{17}$$

(2) Suppose that the probability distribution $F_0{*}F_f(n)$ is DFR. Then, the cost effectiveness $E_2(n_0)$ is a convex function of n_0, and the optimal software rejuvenation schedule is $n_0^* = 0$ or $n_0^* \to \infty$.

4. ESTIMATION ALGORITHMS

4.1 Graphical Method

For the discrete cdf $F_f * F_0(n)$, define the scaled DTTT transform [6, 7]:

$$\phi(p) = \sum_{n=0}^{(F_f*F_0)^{-1}(p)} \frac{\overline{F_f * F_0}(n)}{\mu_f + \mu_0}, \tag{18}$$

where

$$(F_f * F_0)^{-1}(p) = \min\{n : \ F_f * F_0(n) > p\} - 1, \tag{19}$$

if the inverse function exists. Then it is evident that

$$\mu_f + \mu_0 = \sum_{n=0}^{\infty} \overline{F_f * F_0}(n). \tag{20}$$

After a few algebraic manipulations, we can obtain the following result:

Theorem 3: For Model i $(i = 1, 2)$, obtaining the optimal software rejevenation schedule $n_0{}^*$ maximizing the cost effectiveness $E_i(n_0)$ is equivalent to obtaining p^* $(0 \leq p^* \leq 1)$ such as

$$\max_{0 \leq p \leq 1} \frac{\phi(p)}{p + \beta_i}, \tag{21}$$

where $\beta_1 = c_p\mu_c/(c_s\mu_a - c_p\mu_c)$ and $\beta_2 = c_p\mu_c/c_s\mu_a$.

Theorem 3 is the dual of Theorem 1 and Theorem 2. From this result, it is seen that the optimal software rejuvenation schedule $n_0{}^* = (F_f * F_0)^{-1}(p^*)$ is determined by calculating the optimal point $p^*(0 \leq p^* \leq 1)$ maximizing the tangent slope from the point $(-\beta_i, 0)$ $(i = 1, 2)$ to the curve $(p, \phi(p)) \in [0,1] \times [0,1]$ in the two-dimensional plane.

4.2 Statistical Non-parametric Method

Next, suppose that the optimal software rejuvenation schedule has to be estimated from k ordered complete observations: $0 = x_0 \leq x_1 \leq x_2 \leq \cdots \leq x_k$ of the times from a discrete cdf $F_0 * F_f(n)$, which is unknown. Then, the empirical distribution for this sample, is given by

$$F_{fk}(n) = \begin{cases} i/k & \text{for} \quad x_i \leq n < x_{i+1}, \\ 1 & \text{for} \quad x_k \leq n. \end{cases} \tag{22}$$

Then the numerical counterpart of the scaled DTTT transform, called the *scaled DTTT statistics*, based on this sample, is defined by

$$\phi_{ik} \ = \ \psi_i/\psi_k, \quad i = 0, 1, 2, \cdots, k, \tag{23}$$

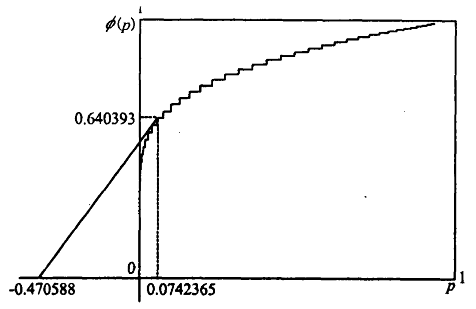

Figure 3: Determination of the optimal software rejuvenation schedule for Model 1.

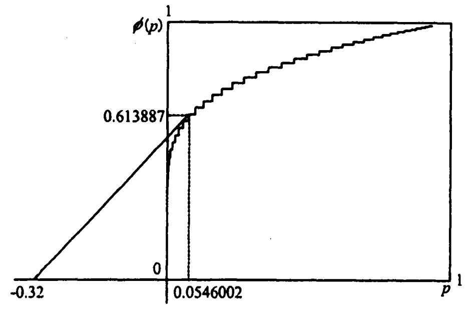

Figure 4: Determination of the optimal software rejuvenation schedule for Model 2.

where

$$\psi_i \;=\; \sum_{j=1}^{i}(k-j+1)(x_j - x_{j-1}), \quad i = 1, 2, \cdots, k;\ \psi_0 = 0. \tag{24}$$

The resulting step function by plotting the points $(i/k, \phi_{ik})$ $(i = 0, 1, 2, \cdots, k)$ is called *the scaled DTTT plot.*

The following theorem gives statistically non-parametric estimation algorithms for the optimal software rejuvenation schedules.

Theorem 4: Suppose that the optimal software rejuvenation schedule has to be estimated from k ordered complete sample $0 = x_0 \le x_1 \le x_2 \le \cdots \le x_k$ of the times from a discrete cdf $F_f{*}F_0(n)$, which is unknown. Then, a non-parametric estimator of the optimal software rejuvenation schedule $\hat{n}_0^*$ which maximizes $E_i(n_0)$ $(i = 1, 2)$ is given by x_{j^*}, where

$$j^* = \left\{ j \mid \max_{0 \le j \le n} \frac{\phi_{jk}}{j/k + \beta_i} \right\}, \quad i = 1, 2. \tag{25}$$

5. NUMERICAL EXAMPLES

5.1 Illustrative Examples

We present some examples to determine the optimal software rejuvenation schedule which maximizes the cost effectiveness. Suppose that the time X obeys the negative binomial distribution with pmf:

$$f_f(n) = \binom{n-1}{r-1} q^r (1-q)^{n-r}, \quad n = 1, 2, 3, \cdots, \tag{26}$$

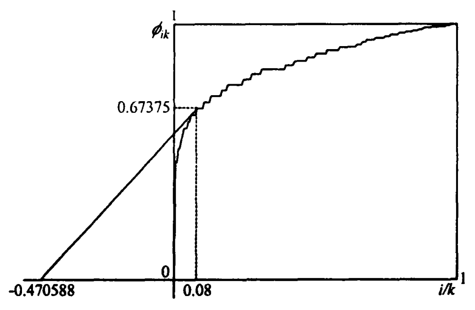

Figure 5: Estimation of the optimal software rejuvenation schedule for Model 1.

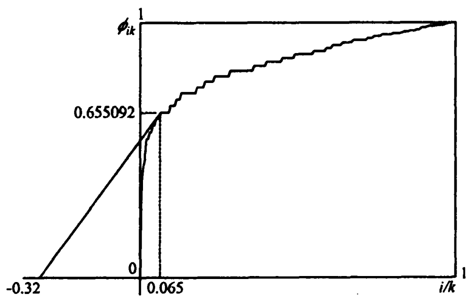

Figure 6: Estimation of the optimal software rejuvenation schedule for Model 2.

where $q \in (0,1)$ and $r = 1, 2, \cdots$ is the natural number. Also, it is assumed that Z is a geometrically distributed random variable having pmf:

$$f_0(n) = p(1-p)^n, \quad n = 0, 1, 2, \cdots. \tag{27}$$

In the rest part of this section, we assume that $(r,\ q) = (10,\ 0.3)$, $p = 0.3$, $c_s = 5.0 \times 10$ [\$/day], $c_p = 4.0 \times 10$ [\$/day], $\mu_a = 5.0$ [day] and $\mu_c = 2.0$ [day].

Figures 3 and 4 illustrate the determination of the optimal software rejuvenation schedule on the two-dimensional graph for Model 1 and Model 2, respectively. Since $p^* = 0.0742365$ has the maximum slope from $(-\beta_1, 0) = (-0.470588, 0)$ in Fig. 3, the optimal software rejuvenation schedule for Model 1 is given by $n_0^* = (F_f * F_0)^{-1}(0.0742365) = 24$. On the other hand, we obtain $n_0^* = (F_f * F_0)^{-1}(0.0546002) = 23$ in Model 2. In both cases, the maximum cost effectiveness are given by $E_1(24) = 0.246606$ and $E_2(23) = 0.233799$, respectively.

Figures 5 and 6 show the estimation results of the optimal software rejuvenation schedule for Model 1 and Model 2, respectively, where the time data are generated from the negative binomial distribution. For 200 simulation data (negative binomial distributed random number), the estimates of the optimal rejuvenation schedule and its associated cost effectiveness are $\hat{n}_0^* = x_{17} = 25$ and $E_1(\hat{n}_0^*) = 0.264209$ in Model 1. On the other hand, one estimates $\hat{n}_0^* = x_{14} = 24$ and $E_2(\hat{n}_0^*) = 0.246961$ in Model 2.

5.2 Asymptotic Behavior

Of our next interest is the investigation of the asymptotic behavior of the estimates for the optimal software rejuvenation schedule. In Figs. 7 and 8, the estimates of the maximum cost effectiveness are plotted where the horizontal line denotes the real maximum calculated based on the negative binomial distribution with same parameters. From these figures, it is observed that the estimate of the cost effectiveness fluctuates around the real maximum and that the non-parametric method proposed here can provide a good

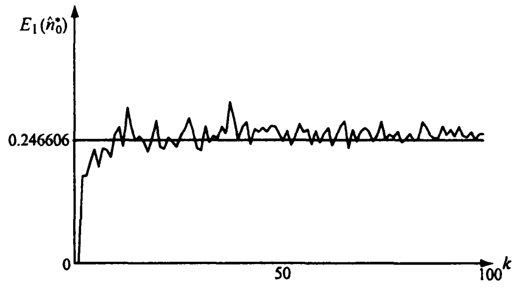

Figure 7: Asymptotic behavior of the estimates for the maximum cost effectiveness (Model 1).

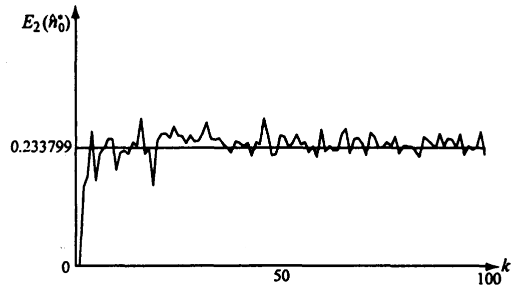

Figure 8: Asymptotic behavior of the estimates for the maximum cost effectiveness (Model 2).

Acknowledgments: This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B), Grant No. 13480109 (2001), and the Research Program 2002 under the Institute for Advanced Studies of the Hiroshima Shudo University, Hiroshima, Japan.

REFERENCES

[1] Adams, E. (1984), Optimizing preventive service of the software products, *IBM Journal of Research & Development*, **28**, 2–14.

[2] Avritzer, A. and Weyuker, E. J. (1997), Monitoring smoothly degrading systems for increased dependability, *Empirical Software Engineering*, **2**, 59–77.

[3] Bobbio, A., Sereno, M. and Anglano, C. (2001), Fine grained software degradation models for optimal rejuvenation policies, *Performance Evaluation*, **46**, 45–62.

[4] Castelli, V., Harper, R. E., Heidelberger, P., Hunter, S. W., Trivedi, K. S., Vaidyanathan, K. V. and Zeggert, W. P. (2001), Proactive management of software aging, *IBM Journal of Research & Development*, **45**, 311–332.

[5] Dohi, T., Goševa-Popstojanova, K. and Trivedi, K. S. (2001), Estimating software rejuvenation schedule in high assurance systems, *The Computer Journal*, **44** (6), 473–485.

[6] Dohi, T., Iwamoto, K., Okamura, H. and Kaio, N. (2002), Discrete-time cost analysis for a telecommunication billing application with rejuvenation, *Proc. Second Euro-Japanese Workshop on Stochastic Risk Modelling for Fianance, Insurance, Production and Reliability*, 181–190.

[7] Dohi, T., Iwamoto, K., Okamura, H. and Kaio, N. (2002), Discrete availability models to rejuvenate a telecommunication billing application, *Proc. 7th IEEE Int'l Symposium on High Assurance Systems Engineering*, 159-166.

[8] Dohi, T., Goševa-Popstojanova, K., Vaidyanathan, K., Trivedi, K. S. and Osaki, S. (2003), Software rejuvenation – modeling and applications, *Springer Reliability Engineering Handbook* (H. Pham, ed.), Springer-Verlag, in press.

[9] Garg, S., Telek, M., Puliafito, A. and Trivedi, K. S. (1995), Analysis of software rejuvenation using Markov regenerative stochastic Petri net, *Proc. 6th Int'l Symposium on Software Reliability Engineering*, 24–27.

[10] Garg, S., Pfening, S., Puliafito, A., Telek, M. and Trivedi, K. S. (1998), Analysis of preventive maintenance in transactions based software systems, *IEEE Transactions on Computers*, **47**, 96–107.

[11] Gray, J. (1986), Why do computers stop and what can be done about it?, *Proc. 5th Int'l Symposium on Reliability in Distributed Software and Database Systems*, 3–12.

[12] Huang, Y., Kintala, C., Kolettin, N. and Funton, N. D. (1995), Software rejuvenation: analysis, module and applications, *Proc. 25th Int'l Symposium on Fault Tolerant Computing*, 381-390.

[13] Iwamoto, K., Dohi, T., Okamura, H. and Kaio, N. (2003), Estimation of discrete-time software rejuvenation schedule based on the cost effectiveness, *Transactions of IEICE (A)*, in press.

[14] Liu, Y., Ma, Y., Han, J. J., Levendel, H. and Trivedi, K. S. (2002), Modeling and analysis of software rejuvenation in cable modem termination system, *Proc. 13th Int'l Symp. on Software Reliability Engineering*, 159–170.

[15] Nakagawa, T. (1984), A summary of discrete replacement policies, *European Journal of Operational Research*, **17** (3), 382–392.

[16] Parnas, D. L. (1994), Software aging, *Proc. 16th Int'l Conf. on Software Engineering*, 279–287.

[17] Park, K. and Kim, S. (2002), Availability analysis and improvement of active/standby cluster systems using software rejuvenation, *Journal of Systems and Software*, **61**, 121–128.

[18] Pfening, S., Garg, S., Puliafito, A., Telek, M. and Trivedi, K. S. (1996), Optimal rejuvenation for toleranting soft failure, *Performance Evaluation*, **27/28**, 491–506.