<table>
<thead>
<tr>
<th>Title</th>
<th>Estimating Discrete-Time Periodic Software Rejuvenation Schedules under Cost Effectiveness Criterion (Mathematics of Decision-making under uncertainty)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Iwamoto, Kazuki; Dohi, Tadashi; Kaio, Naoto</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (2003), 1306: 152-161</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2003-02</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/42817">http://hdl.handle.net/2433/42817</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>

京都大学学術情報リポジトリ
Kyoto University Research Information Repository
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 collisions 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 example, 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)

\[ F_{0}(n), f_{0}(n), \mu_{0}(>0): \text{cdf, pmf and mean of } Z, \text{where } n = 0, 1, 2, \ldots \]

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

\[ F_{f}(n), f_{f}(n), \mu_{f}(>0): \text{cdf, pmf and mean of } X \]

\[ * *: \text{discrete convolution operator, i.e. } F_{0} * F_{f}(n) = \sum_{j=0}^{n} F_{0}(n-j)f_{f}(j) = \sum_{j=0}^{n} F_{f}(n-j)f_{0}(j) \]

\[ \overline{\psi}(\cdot): \text{survivor function} \ (= 1 - \psi(\cdot)) \]

\[ r_{f}(n): = f_{0} * f_{f}(n)/F_{0} * F_{f}(n-1) \]

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

\[ F_{a}(n), f_{a}(n), \mu_{a}(>0): \text{cdf, pmf and mean of } Y \]

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

\[ F(n), n_{0}(\geq 0): \text{cdf and mean of } N \]
$R$: system overhead incurred by software rejuvenation (discrete random variable)

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

c$_s$: recovery cost from system failure per unit time

c$_p$: rejuvenation cost per unit time

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

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 $\Pr\{Z \leq n\} = F_0(n)$, $(n = 0, 1, 2, \cdots)$. 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 $\Pr\{X \leq n\} = F_f(n)$. 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 $\Pr\{Y \leq n\} = F_a(n)$. 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 probability distribution function $F(n)$ has to be replaced by

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

where $U(\cdot)$ is the unit step function. We call $n_0$ ($\geq 0$) 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 corresponding 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

$$S_1(n_0) = \sum_{n=0}^{n_0-1} \bar{F}_f * \bar{F}_0(n).$$  (2)

The total expected cost for one cycle is obtained as

$$V_1(n_0) = c_s \mu_a F_f \ast F_0(n_0) + c_p \mu_c \bar{F} \ast F_0.$$  (3)

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

$$E_1(n_0) = S_1(n_0)/V_1(n_0)$$  (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]:

$$q_1(n_0) = \frac{V_1(n_0) V_1(n_0 + 1) [E_1(n_0 + 1) - E_1(n_0)]}{\bar{F}_0 \ast \bar{F}_f(n_0)} = V_1(n_0) - (c_s \mu_a - c_p \mu_c) S_1(n_0) r_{0f}(n_0 + 1),$$  (5)

where \(f_f \ast f_0(n)\) is the pmf of the probability distribution \(F_f \ast F_0(n)\).
The following result gives the optimal software rejuvenation schedule for Model 1.

**Theorem 1**: For Model 1, (1) suppose that the probability distribution $F_0 \ast F_f(n)$ 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 $q_1(n_0^*) \leq 0$. Then, the maximum cost effectiveness is given by

$$E_1(n_0^*) \leq E_1(n_0^*) < E_1(n_0^*),$$

where

$$E_1(n_0^*) = \frac{1}{(c_s \mu_a - c_p \mu_c) r(n_0^* + 1)},$$

and

$$E_1(n_0^*) = \frac{1}{(c_s \mu_a - c_p \mu_c) r(n_0^*)}.$$  

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

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

(2) Suppose that the probability distribution $F_0 \ast F_f(n)$ 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 $n_0^* \rightarrow \infty$.

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

$$S_2(n_0) = \sum_{n=0}^{n_0-1} \overline{F_f} \ast F_0(n),$$

and

$$V_2(n_0) = c_s \mu_a \overline{F_f} \ast F_0(n_0) + c_p \mu_c,$$

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

$$E_2(n_0) = S_2(n_0)/V_2(n_0).$$

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

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

**Theorem 2**: For Model 2, (1) suppose that the probability distribution $F_0 \ast F_f(n)$ 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 $q_2(n_0^*) \leq 0$. Then, the maximum cost effectiveness is given by

$$E_2(n_0^*) \leq E_2(n_0^*) < E_2(n_0^*),$$

where

$$E_2(n_0^*) = \frac{1}{c_s \mu_a r(n_0^* + 1)},$$

and

$$E_2(n_0^*) = \frac{1}{c_s \mu_a r(n_0^*)}.$$
(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
\[
E_2(\infty) = \frac{\mu_f + \mu_0}{c_p\mu_c + c_s\mu_a}.
\] (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 \( n_0^* = \min \{0, \infty\} \).

4. ESTIMATION ALGORITHMS

4.1 Graphical Method

For the discrete cdf \( F_f*F_0(n) \), define the scaled DTTT transform \([6, 7]\):
\[
\phi(p) = \sum_{n=0}^{(F_f*F_0)^{-1}(p)} \frac{F_f*F_0(n)}{\mu_f + \mu_0},
\] (18)
where
\[
(F_f*F_0)^{-1}(p) = \min \{n : F_f*F_0(n) > p\} - 1,
\] (19)
if the inverse function exists. Then it is evident that
\[
\mu_f + \mu_0 = \sum_{n=0}^{\infty} \frac{F_f*F_0(n)}{\mu_f + \mu_0}.
\] (20)

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

**Theorem 3:** For Model \( i \) (\( i = 1, 2 \)), obtaining the optimal software rejuvenation schedule \( n_0^* \) maximizing the cost effectiveness \( E_i(n_0) \) is equivalent to obtaining \( p^* \) (0 \( \leq p^* \leq 1 \)) such as
\[
\max_{0 \leq p \leq 1} \frac{\phi(p)}{p + \beta_i},
\] (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 \( (p, \phi(p)) \) \([0, 1] \times [0, 1]\) 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
\[
F_{fk}(n) = \begin{cases} i/k & \text{for } x_i \leq n < x_{i+1}, \\ 1 & \text{for } x_k \leq n. \end{cases}
\] (22)

Then the numerical counterpart of the scaled DTTT transform, called the **scaled DTTT statistics**, based on this sample, is defined by
\[
\psi_{ik} = \frac{\psi_i/\psi_k}{}, \quad i = 0, 1, 2, \cdots, k,
\] (23)
where

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

The resulting step function by plotting the points \((i/k, \phi_{ik})\) \((i = 0, 1, 2, \ldots, k)\) 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 \(0 = x_0 \leq x_1 \leq x_2 \leq \cdots \leq x_k\) of the times from a discrete cdf \(F_{i*}F_0(n)\), which is unknown. Then, a non-parametric estimator of the optimal software rejuvenation schedule \(n_0^*\) which maximizes \(E_i(n_0)\) \((i = 1, 2)\) is given by \(x_j^*\), where

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

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:

$$f_{j}(n) = \binom{n - 1}{r - 1} q^r(1 - q)^{n-r}, \quad n = 1, 2, 3, \ldots, \quad (26)$$
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:

$$f_0(n) = p(1-p)^n, \quad n = 0, 1, 2, \cdots$$

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

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


