Analysis of Software Cost Models with Rejuvenation *

Tadashi Dohi
Dept. of Industrial and Systems Engineering
Hiroshima University
Higashi-Hiroshima 739-8527, Japan
dohi@gal.sys.hiroshima-u.ac.jp

Katerina Goševa-Popstojanova † Kishor S. Trivedi
Dept. of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0294, USA
{katerina, kst}@ee.duke.edu

Abstract

Software rejuvenation is a preventive maintenance technique that has been extensively studied in the recent literature. In this paper, we extend the classical result by Huang, Kintala, Kolettis and Fulton (1995), and in addition propose a modified stochastic model to generate the software rejuvenation schedule. More precisely, the software rejuvenation models are formulated via the semi-Markov process, and the optimal software rejuvenation schedule which minimizes the expected costs per unit time in the steady-state are derived analytically for respective cases. Further, we develop non-parametric algorithms to estimate the optimal software rejuvenation schedules, provided that the statistical complete (unsensored) sample data of failure time is given. In numerical examples, we compare two models in terms of economic justification, and examine asymptotic properties for the statistical estimation algorithms.

1. Introduction

Demands on software reliability and availability have increased tremendously due to the nature of present day applications. They impose stringent requirements in terms of cumulative downtime and failure free operation of software, since in many cases, the consequences of software failure can lead to huge economic losses or risk to human life. However, these requirements are very difficult to design and guarantee, particularly in applications of nontrivial complexity.

When software application executes continuously for long periods of time, it ages 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. Huang *et al.* [1] report this phenomenon in telecommunications billing applications where over time the application experiences a crash or a hang failure. Avritzer and Weyuker [2] discuss aging in telecommunication switching software where the effect manifests as gradual performance degradation. Software aging has also been observed in widely-used software like Netscape and xrn. Perhaps the most vivid example of aging in safety critical systems is the Patriot's software [3], where the accumulated errors led to a failure that resulted in loss of human life.

Resource leaking and other problems causing software to age are due to the software faults whose fixing is not always possible because, for example, application developer may not have the access to the source code. Furthermore, it is almost impossible to fully test and verify if a piece of software is fault-free. Testing software becomes harder if it is complex, and further more testing cycle times are often reduced due to smaller release time requirements. Common experience suggests that most software failures are transient in nature [4]. Since transient failures will disappear if the operation is retried later in slightly different context, it is difficult to characterize their root origin. Therefore, the residual faults have to be tolerated in the operational phase. Usual strategies to deal with failures in operational phase are reactive in nature; they consist of action taken after failure.

Recently, a complementary approach to handle transient software failures, called *software rejuvenation*, was proposed [1]. Software rejuvenation is a preventive and proactive (as opposite to being reactive) 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, *etc.* An extreme, but well known example of rejuvenation is a hardware reboot. Apart from being used in an ad-hoc manner by almost all computer

*The first author is supported in part by Telecommunication Advancement Foundation, Tokyo, Japan. This material is based upon the support by the Department of Defense, Alcatel, Telcordia and Aprisma Management Technologies, via a CACC core project, NC, USA.

†On leave of absence from the Department of Computer Science, Faculty of Electrical Engineering, Skopje, Macedonia

users, rejuvenation has been used in high assurance systems, including continuously available telecommunication systems [1, 2] and high consequence systems [5]. Although the fault in the software still remains, performing rejuvenation periodically removes or minimizes potential error conditions due to that fault, thus preventing failures that might have unacceptable consequences.

Rejuvenation has the same motivation and advantages/disadvantages as preventive maintenance policies in hardware systems. Any rejuvenation typically involves an overhead, but, on the other hand, prevents more severe failures to occur. The application will of course be unavailable during rejuvenation, but since this is a scheduled downtime the cost is expected to be much lower than the cost of an unscheduled downtime caused by failure. Hence, an important issue is to determine the optimal schedule to perform software rejuvenation in terms of availability and cost.

In this paper, we extend the classical result by Huang *et al.* [1], and in addition propose a modified stochastic model to determine the optimal software rejuvenation schedule. More precisely, the software rejuvenation models are formulated via the semi-Markov process, and the optimal software rejuvenation schedules which minimize the expected costs per unit time in the steady-state are derived analytically for respective cases. Further, since the failure time distribution can not be easily estimated from a few data samples, we develop non-parametric statistical algorithms to estimate the optimal software rejuvenation schedules, provided that the statistical complete (unsensored) sample data of failure times is given. These can be useful in determining optimal rejuvenation schedule in the early position of the operational phase. In numerical examples, we examine asymptotic properties of the statistical estimation algorithms.

2. Related work

In recent years, considerable attention has been devoted to the phenomenon of software aging. For extensive surveys, the reader is referred to [6]. The studies of aging-related failures are based on two approaches: measurement-based and modeling-based. The measurement-based approach is concentrated on the detection and validation of the existence of software aging and estimating its effects on system resources [7], [8]. The modeling-based approach is aimed at evaluating the effectiveness of software rejuvenation and determining the optimal schedules to perform rejuvenation. Next, we present the brief overview of the previous work related to modeling-based approaches.

A great deal of research effort on modeling software aging and rejuvenation considers continuously running software systems. Huang *et al.* [1] used continuous time Markov chain to model software rejuvenation. They considered the two-step failure model where the application goes from the initial robust (clean) state to a failure probable (degraded) state from which two actions are possible: rejuvenation or transition to failure state. Both rejuvenation and recovery from failure return the software system to the robust state. Garg *et al.* [9] introduced into the model the idea of periodic rejuvenation. To deal with deterministic interval between successive rejuvenations the system behavior was represented through a Markov regenerative stochastic Petri net model. The subsequent work [10] involves arrival and queueing of jobs in the system and computes load and time dependent rejuvenation policy. The above models consider the effect of aging as crash/hang failure, referred to as hard failures, which result in unavailability of the software. However, due to the aging the software system can exhibit soft failures, that is, performance degradation. In [11] the performance degradation is modeled by the gradual decrease of the service rate. Both effects of aging, hard failures that result in an unavailability and soft failures that result in performance degradation, are considered in the model of transaction based software system presented in [12]. This model was recently generalized in [13] by considering multiple servers.

The fine grained software rejuvenation model presented in [14] takes a different approach to characterize the effect of software aging. It assumes that the degradation process consists of a sequence of additive random shocks; the system is considered out of service as soon as the appropriate parameter reaches an assigned threshold level.

Several studies considered software with a finite mission time. The work in [15] analyzes the effects of checkpointing and rejuvenation used together on the expected completion time of a software program. The use of preventive on-board maintenance that includes periodic software and system rejuvenation is proposed and analyzed in [5].

The cost-based models considered in this paper have similar but somewhat generalized mathematical structure to that in Huang *et al.* [1]. However, the approaches taken to estimate the optimal software rejuvenation schedules are quite different. Note that in the above literature, the failure time distribution needs to be specified to derive the optimal rejuvenation schedule and to calculate several reliability measures. This seems to be restrictive, since the determination of the theoretical distribution from the real data is rather troublesome, and needs both the goodness-of-fit test and the parameter estimation based on several candidate distribution functions. Although in the existing modeling-based approach, the failure time distribution is fixed, for instance, the Weibull distribution, this has not yet been validated for software aging. By contrast, our approach does not depend on the kind of distribution function and can provide non-parametric estimators of the optimal software rejuvenation schedules which minimize the expected costs per

unit time in the steady-state. Thus we provide a very powerful approach for the application of the rejuvenation to a real system operation.

3. Model description

3.1. Basic model

First, we introduce the basic software rejuvenation model proposed by Huang *et. al* [1]. Although they formulated it as a simple continuous-time Markov chain, we extend their result in the more general mathematical framework. In particular, we regard the software rejuvenation models as continuous-time semi-Markov processes. Define the following four states:

State 0: highly robust state (normal operation state)

State 1: failure probable state

State 2: failure state

State 3: software rejuvenation state.

Suppose that all the states mentioned above are regeneration points. More specifically, let Z be the random time interval when the highly robust state changes to the failure probable state, having the common distribution function $\Pr\{Z \leq t\} = F_0(t)$ with finite mean μ_0 (> 0). Just after the state becomes the failure probable state, a system failure may occur with a positive probability. Without loss of generality, we assume that the random variable Z is observable during the system operation [1, 9]. The transition diagram for Model 1 is depicted in Fig. 1.

Define the failure time X (from State 1) and the repair time Y, having the distribution functions $\Pr\{X \leq t\} = F_f(t)$ and $\Pr\{Y \leq t\} = F_a(t)$ with finite means λ_f (> 0) and μ_a (> 0), respectively. If the system failure occurs before triggering a software rejuvenation, then the repair is started immediately at that time and is completed after the random time Y elapses. Otherwise, the software rejuvenation is started. Note that in the basic model referred to as Model 1 the software rejuvenation cycle is measured from the time instant just after the system enters State 1 from State 0. Denote the distribution functions of the time to software rejuvenation and its system overhead by $F_r(t)$ and $F_c(t)$ (with mean μ_c (> 0)), respectively. After completing the repair or the rejuvenation, the software system becomes as good as new, and the software age is initiated at the beginning of the next highly robust state. Consequently, we define the time interval from the beginning of the system operation to the next one as one cycle, and the same cycle is repeated again and again. If we consider the time to software rejuvenation as a constant t_0, then it follows that

$$F_r(t) = U(t - t_0) = \begin{cases} 1 & \text{if } t \geq t_0 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

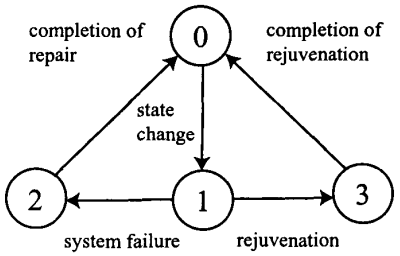

Figure 1. State transition diagram of Model 1

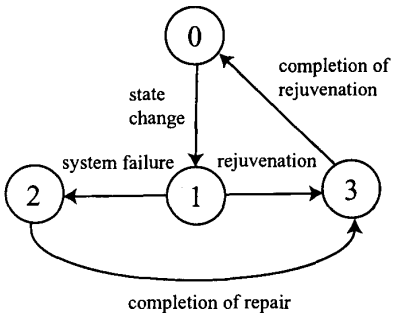

Figure 2. State transition diagram of Model 2

We call t_0 (≥ 0) as *the software rejuvenation schedule* in this paper and $U(\cdot)$ is the unit step function. Hence, the underlying stochastic process is the semi-Markov process with four regeneration states. Note that under the assumption that the sojourn times in all states are exponentially distributed, Model 1 is reduced to the one in Huang *et al.* [1].

3.2. Modified model

The next model is a modification of the basic model. When the repair is completed after the system failure, is the software system really renewed? Probably, the answer is "NO" in many cases. If one distinguishes a software repair and the software rejuvenation, an additional rejuvenation might be needed after the software repair. For example, restarting the system after repair might require some cleanup and resuming the process execution at the checkpointed state. Such an additional rejuvenation policy has not been considered in the literature in spite of the practical need. Figure 2 is the transition diagram for Model 2. In this model, the software rejuvenation is performed just after the completion of repair as well as at the constant time t_0 after the failure probable state is entered, *i.e.*, $\min\{Z + X + Y, Z + t_0\}$.

4. Cost analysis

For each semi-Markov process described in Section 3, define the transition probability $Q_{ij}(t)$ $(i,j = 0,\cdots,3, i \neq j)$. Also, we define the Laplace Stieltjes transform (LST) of the transition probability by $q_{ij}(s) = \int_0^\infty \exp\{-st\}dQ_{ij}(t)$. For Model 1, it is straightforward to obtain

$$q_{01}(s) = \int_0^\infty \exp\{-st\}dF_0(t), \quad (2)$$

$$q_{12}(s) = \int_0^\infty \exp\{-st\}\overline{F}_r(t)dF_f(t), \quad (3)$$

$$q_{13}(s) = \int_0^\infty \exp\{-st\}\overline{F}_f(t)dF_r(t), \quad (4)$$

$$q_{20}(s) = \int_0^\infty \exp\{-st\}dF_a(t), \quad (5)$$

$$q_{30}(s) = \int_0^\infty \exp\{-st\}dF_c(t), \quad (6)$$

where in general $\overline{\psi}(\cdot) = 1 - \psi(\cdot)$. Define the recurrent time distribution from State 0 to 0 again by $H_{00}(t)$. Then the LST of the recurrent time distribution is

$$h_{00}(s) = \int_0^\infty \exp\{-st\}dH_{00}(t) = q_{01}(s)q_{12}(s)q_{20}(s) + q_{01}(s)q_{13}(s)q_{30}(s). \quad (7)$$

Of our concern is the derivation of the transient probability to stay in State j $(j = 0,\cdots,3)$ at arbitrary time t (> 0), provided that the initial state at time $t = 0$ is 0. Define the transient probability from State 0 to j $(j = 0,\cdots,3)$ and its LST by $P_{0j}(t)$ and $p_{0j}(s) = \int_0^\infty \exp\{-st\}dP_{0j}(t)$, respectively. After some manipulations, we have

$$p_{00}(s) = \overline{q}_{01}(s)/\overline{h}_{00}(s), \quad (8)$$

$$p_{01}(s) = q_{01}(s)\Big(\overline{q}_{12}(s) - q_{13}(s)\Big)/\overline{h}_{00}(s), \quad (9)$$

$$p_{02}(s) = q_{01}(s)q_{12}(s)\overline{q}_{20}(s)/\overline{h}_{00}(s), \quad (10)$$

$$p_{03}(s) = q_{01}(s)q_{13}(s)\overline{q}_{30}(s)/\overline{h}_{00}(s). \quad (11)$$

In a fashion similar to Model 1, define the LST of the transition probability for Model 2 by

$$q_{01}(s) = \int_0^\infty \exp\{-st\}dF_0(t), \quad (12)$$

$$q_{12}(s) = \int_0^\infty \exp\{-st\}\overline{F}_r(t)dF_f(t), \quad (13)$$

$$q_{13}(s) = \int_0^\infty \exp\{-st\}\overline{F}_c(t)dF_r(t), \quad (14)$$

$$q_{23}(s) = \int_0^\infty \exp\{-st\}dF_a(t), \quad (15)$$

$$q_{30}(s) = \int_0^\infty \exp\{-st\}dF_c(t). \quad (16)$$

Also, the LST of the recurrent time distribution is

$$h_{00}(s) = q_{01}(s)q_{12}(s)q_{23}(s)q_{30}(s) + q_{01}(s) \times q_{13}(s)q_{30}(s). \quad (17)$$

Then, we have

$$p_{00}(s) = \overline{q}_{01}(s)/\overline{h}_{00}(s), \quad (18)$$

$$p_{01}(s) = q_{01}(s)\Big(\overline{q}_{12}(s) - q_{13}(s)\Big)/\overline{h}_{00}(s), \quad (19)$$

$$p_{02}(s) = q_{01}(s)q_{12}(s)\overline{q}_{23}(s)/\overline{h}_{00}(s), \quad (20)$$

$$p_{03}(s) = \Big\{q_{01}(s)q_{12}(s)q_{23}(s)\overline{q}_{30}(s) + q_{01}(s)q_{13}(s) \times \overline{q}_{30}(s)\Big\}/\overline{h}_{00}(s). \quad (21)$$

Define the following cost components:

c_s (> 0): repair cost per unit time

c_p (> 0): rejuvenation cost per unit time.

Further, we make the following assumptions:

(A-1) $\mu_a > \mu_c$

(A-2) $c_s > c_p$.

The assumption (A-1) means that the mean time to repair is strictly larger than the mean time to complete software rejuvenation. Also, the assumption (A-2) implies that the cost of repair is larger than the cost of software rejuvenation. These assumptions are quite reasonable and intuitive.

Then, the expected cost per unit time in the steady-state for Model 1 becomes

$$\begin{aligned} C_1(t_0) &= \lim_{t\to\infty} \frac{\mathrm{E}[\text{total cost during } (0,t]]}{t} \\ &= \lim_{t\to\infty} \Big\{c_s P_{02}(t) + c_p P_{03}(t)\Big\} \\ &= \lim_{s\to 0} -\Big\{c_s p_{02}(s) + c_p p_{03}(s)\Big\} \\ &= \frac{c_s\mu_a F_f(t_0) + c_p\mu_c\overline{F}_f(t_0)}{\mu_0 + \mu_a F_f(t_0) + \mu_c\overline{F}_f(t_0) + \int_0^{t_0}\overline{F}_f(t)dt} \\ &= V_1(t_0)/S_1(t_0). \end{aligned} \quad (22)$$

Also, the expected cost per unit time in the steady-state for Model 2 is

$$\begin{aligned} C_2(t_0) &= \frac{c_s\mu_a F_f(t_0) + c_p\mu_c}{\mu_0 + \mu_c + \mu_a F_f(t_0) + \int_0^{t_0}\overline{F}_f(t)dt} \\ &= V_2(t_0)/S_2(t_0). \end{aligned} \quad (23)$$

The problem is to derive the optimal software rejuvenation schedule t_0^* which minimizes the expected cost per unit time in the steady-state $C_i(t_0)$ $(i = 1,2)$. The following results give the optimal software rejuvenation schedule for Model i $(i = 1,2)$.

Theorem 1: For Model 1, (1) suppose that the failure time distribution is strictly IFR (increasing failure rate) under the assumptions (A-1) and (A-2). Define the following non-linear function:

$$q_1(t_0) = (c_s\mu_a - c_p\mu_c)r_f(t_0)S_1(t_0) - \Big\{(\mu_a - \mu_c)r_f(t_0) + 1\Big\}V_1(t_0), \quad (24)$$

where $r_f(t) = (dF_f(t)/dt)/\overline{F}_f(t)$ is the failure rate.

(i) If $q_1(0) < 0$ and $q_1(\infty) > 0$, then there exists a finite and unique optimal software rejuvenation schedule t_0^* $(0 < t_0^* < \infty)$ satisfying $q_1(t_0^*) = 0$, and the minimum expected cost is

$$C_1(t_0^*) = \frac{(c_s\mu_a - c_p\mu_c)r_f(t_0^*)}{(\mu_a - \mu_c)r_f(t_0^*) + 1}. \quad (25)$$

(ii) If $q_1(0) \geq 0$, then the optimal software rejuvenation schedule is $t_0^* = 0$, *i.e.* it is optimal to start the software rejuvenation just after entering the failure probable state, and the minimum expected cost is $C_1(0) = c_p\mu_c/(\mu_0 + \mu_c)$.

(iii) If $q_1(\infty) \leq 0$, then the optimal software rejuvenation schedule is $t_0^* \to \infty$, *i.e.* it is optimal not to carry out the software rejuvenation, and the minimum expected cost is $C_1(\infty) = (c_s\mu_a)/(\mu_0 + \mu_a + \lambda_f)$.

(2) Suppose that the failure time distribution is DFR (decreasing failure rate) under the assumptions (A-1) and (A-2). Then, the expected cost function $C_1(t_0)$ is a concave function of t_0, and the optimal software rejuvenation schedule is $t_0^* = 0$ or $t_0^* \to \infty$.

Proof: Differentiating $C_1(t_0)$ with respect to t_0 and setting it equal to 0 implies $q_1(t_0) = 0$. Further differentiation yields

$$\frac{dq_1(t_0)}{dt_0} = \frac{dr_f(t_0)}{dt_0}\Big[(c_s\mu_a - c_p\mu_c)S_1(t_0) - (\mu_a - \mu_c)V_1(t_0)\Big]. \quad (26)$$

If the term in the bracket of the right hand side in Eq.(26) is negative, then $c_s + \mu_c(c_s - c_p)/(\mu_a - \mu_c) \leq C_1(t_0)$ for all $t_0 \in [0, \infty)$. Since $\mu_c(c_s - c_p)/(\mu_a - \mu_c) > 0$ from the assumptions (A-1) and (A-2), this contradicts the probability law even if the repair occurs in the steady-state with probability one. Hence, it follows from the reduction argument that $c_s + \mu_c(c_s - c_p)/(\mu_a - \mu_c) > C_1(t_0)$.

If $r_f(t_0)$ is strictly increasing, then the function $q_1(t_0)$ is strictly increasing and the expected cost $C_1(t_0)$ is strictly convex in t_0. Further, if $q_1(0) < 0$ and $q_1(\infty) > 0$, then there exists a unique optimal software rejuvenation schedule t_0^* $(0 < t_0^* < \infty)$ satisfying $q_1(t_0^*) = 0$. If $q_1(0) \geq 0$ or $q_1(\infty) \leq 0$, then the expected cost is monotonically decreasing or increasing in t_0, and the optimal policy becomes $t_0^* = 0$ or $t_0^* \to \infty$. On the other hand, if $r_f(t_0)$ is a decreasing function of t_0, then $C_1(t_0)$ is a concave function of t_0, and the optimal software rejuvenation schedule is $t_0^* = 0$ or $t_0^* \to \infty$. The proof is thus completed.

It is easy to check that the result above implies that in Huang *et. al* [1], although they used the system downtime and its associated cost as criteria of optimality. As is clear from Theorem 1, when the failure time obeys the exponential distribution, the optimal software rejuvenation schedule becomes $t_0^* = 0$ or $t_0^* \to \infty$. It means that the rejuvenation should be performed as soon as the software enters the failure probable state $(t_0 = 0)$ or should not be performed at all $(t_0 \to \infty)$. Therefore, the determination of the optimal rejuvenation schedule based on the the expected cost is never motivated in such a situation. Since for a software system which ages it is more realistic to assume that failure time distribution is strictly IFR, our general setting is plausible and the result satisfies our intuition.

Similarly, consider Model 2. The result is given without proof for brevity.

Theorem 2: For Model 2, (1) suppose that the failure time distribution is strictly IFR. Define the following non-linear function:

$$q_2(t_0) = c_s\mu_a r_f(t_0)S_2(t_0) - \Big\{\mu_a r_f(t_0) + 1\Big\}V_2(t_0). \quad (27)$$

(i) If $q_2(0) < 0$ and $q_2(\infty) > 0$, then there exists a finite and unique optimal software rejuvenation schedule t_0^* $(0 < t_0^* < \infty)$ satisfying $q_2(t_0^*) = 0$, and the minimum expected cost is

$$C_2(t_0^*) = \frac{c_s\mu_a r_f(t_0^*)}{\mu_a r_f(t_0^*) + 1}. \quad (28)$$

(ii) If $q_2(0) \geq 0$, then the optimal software rejuvenation schedule is $t_0^* = 0$, and the minimum expected cost is $C_2(0) = c_p\mu_c/(\mu_0 + \mu_c)$.

(iii) If $q_2(\infty) \leq 0$, then the optimal software rejuvenation schedule is $t_0^* \to \infty$, and the minimum expected cost is $C_2(\infty) = (c_s\mu_a + c_p\lambda_c)/(\mu_0 + \mu_a + \mu_c + \lambda_f)$.

(2) Suppose that the failure time distribution is DFR. Then, the expected cost $C_2(t_0)$ is a concave function of t_0, and the optimal software rejuvenation schedule is $t_0^* = 0$ or $t_0^* \to \infty$.

Notice that the assumptions (A-1) and (A-2) are not needed for Model 2. It can be seen from the fact $c_s > C_2(t_0)$ for

all $t_0 \in [0, \infty)$ that the expected cost is convex (concave) in t_0 when the failure time distribution is IFR (DFR).

In this section, we derived the optimal software rejuvenation schedules which minimize the expected costs per unit time in the steady-state. It should be noted, however, that the optimal software rejuvenation schedule has to be determined from model parameters; μ_0, μ_c, μ_a, c_s and c_p and the failure time distribution $F_f(t)$. In other words, it is not easy in general to specify the failure time distribution in the software operational phase. In the following section, we develop statistical non-parametric algorithms to estimate the optimal software rejuvenation schedules, provided that the complete (unsensored) sample data of failure times is given.

5. Statistical optimization algorithms

Before developing the statistical estimation algorithms for the optimal software rejuvenation schedules, we translate the underlying problems $\min_{0 \le t_0 < \infty} C_i(t_0)$ $(i = 1, 2)$ to graphical ones. Following Barlow and Campo [16], define the scaled total time on test (TTT) transform of the failure time distribution:

$$\phi(p) = (1/\lambda_f) \int_0^{F_f^{-1}(p)} \overline{F}_f(t) dt, \tag{29}$$

where

$$F_f^{-1}(p) = \inf\{t_0; F_f(t_0) \ge p\}, \quad (0 \le p \le 1). \tag{30}$$

It is well known [16] that $F_f(t)$ is IFR (DFR) if and only if $\phi(p)$ is concave (convex) on $p \in [0, 1]$. After a few algebraic manipulations, we have the following result.

Theorem 3: Suppose that the assumptions (A-1) and (A-2) are satisfied. For Model i $(i = 1, 2)$, obtaining the optimal software rejuvenation schedule $t_0{}^*$ minimizing the expected cost per unit time in the steady-state $C_i(t_0)$ is equivalent to obtaining p^* $(0 \le p^* \le 1)$ such as

$$\max_{0 \le p \le 1} \frac{\phi(p) + \alpha_i}{p + \beta_i}, \tag{31}$$

where

$$\alpha_1 = \frac{(c_s - c_p)\mu_c(\mu_0 + \mu_a)}{\lambda_f(c_s\mu_a - c_p\mu_c)}, \tag{32}$$

$$\beta_1 = \frac{c_p\mu_c}{c_s\mu_a - c_p\mu_c}, \tag{33}$$

$$\alpha_2 = \frac{(c_s - c_p)\mu_c + c_s\mu_0}{c_s\lambda_f}, \tag{34}$$

$$\beta_2 = \frac{c_p\mu_c}{c_s\mu_a}. \tag{35}$$

Theorem 3 is the dual of Theorem 1 and Theorem 2. From this result, it is seen that the optimal software rejuvenation schedule $t_0{}^* = F_f^{-1}(p^*)$ is determined by calculating the optimal point $p^*(0 \le p^* \le 1)$ maximizing the tangent slope from the point $(-\beta_i, -\alpha_i) \in (-\infty, 0) \times (-\infty, 0)$ to the curve $(p, \phi(p)) \in [0, 1] \times [0, 1]$. This idea is essentially due to the statistical checkpointing algorithms in [17].

Next, suppose that the optimal software rejuvenation schedule has to be estimated from an ordered complete (unsensored) observation $0 = x_0 \le x_1 \le x_2 \le \cdots \le x_n$ of the failure times from an absolutely continuous distribution F_f, which is unknown. Then the scaled TTT statistics based on this sample are defined by $\phi_{nj} = \psi_j/\psi_n$, where

$$\psi_j = \sum_{k=1}^{j} (n-k+1)(x_k - x_{k-1}), \quad (j = 1, 2, \cdots, n;\ \psi_0 = 0). \tag{36}$$

Since the empirical distribution function $F_n(x)$ corresponding to the sample data x_j $(j = 0, 1, 2, \cdots, n)$ is

$$F_n(x) = \begin{cases} j/n & \text{for} \quad x_j \le x < x_{j+1}, \\ 1 & \text{for} \quad x_n \le x, \end{cases} \tag{37}$$

the resulting polygon by plotting the points $(F_n(x), \phi_{nj})$ $(j = 0, 1, 2, \cdots, n)$ and connecting them by line segments is called the *scaled TTT plot.* In other words, the scaled TTT plot can be regarded as a numerical counter part of the scaled TTT transform.

The following result gives non-parametric statistical estimation algorithms for the optimal software rejuvenation schedules.

Theorem 4: (i) Suppose that the optimal software rejuvenation schedule has to be estimated from n ordered complete sample $0 = x_0 \le x_1 \le x_2 \le \cdots \le x_n$ of the failure times from an absolutely continuous distribution F_f, which is unknown. Then, a non-parametric estimator of the optimal software rejuvenation schedule $\hat{t}_0^*$ which minimizes $C_i(t_0)$ $(i = 1, 2)$ is given by x_{j^*}, where

$$j^* = \left\{ j \mid \max_{0 \le j \le n} \frac{\phi_{nj} + \alpha_i}{j/n + \beta_i} \right\} \tag{38}$$

and λ_f in Eqs. (32) and (34) is replaced by $\sum_{k=1}^{n} x_k/n$.
(ii) The estimator given in (i) is strongly consistent, *i.e.* x_{j^*} converges to the optimal solution $t_0{}^*$ uniformly with probability one as $n \to \infty$, if a unique optimal software rejuvenation schedule exists.

It is straightforward to prove the above result in (i) from Theorem 3. The uniform convergence property in (ii) is due to the Glivenko-Cantelli lemma [16] and the strong law of large numbers. The graphical procedure proposed here has

an educational value for better understanding of the optimization problem and it is convenient for performing sensitivity analysis of the optimal software rejuvenation policy when different values are assigned to the model parameters. The special interest is, of course, to estimate the optimal schedule without specifying the failure time distribution. Although some typical theoretical distribution functions such as the Weibull distribution and the gamma distribution are assumed in the reliability analysis, our nonparametric estimation algorithm can generate the optimal software rejuvenation schedule using the on-line knowledge about the observed failure times.

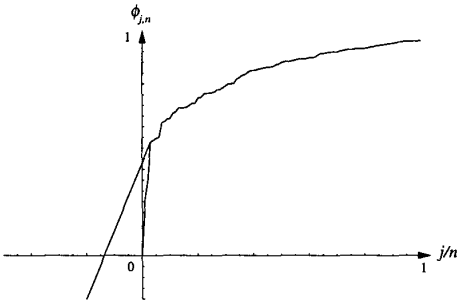

Figure 3. Estimation of the optimal software rejuvenation schedule on the two-dimensional graph (Model 1): $\theta = 9.0 \times 10^{-1}$, $\beta = 4.0$, $\mu_0 = 2.0$, $\mu_a = 4.0 \times 10^{-2}$, $\mu_c = 3.0 \times 10^{-2}$, $c_s = 2.0 \times 10^{-2}$, $c_p = 1.0 \times 10^{-2}$.

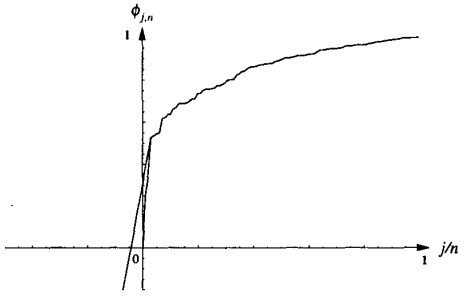

Figure 4. Estimation of the optimal software rejuvenation schedule on the two-dimensional graph (Model 2): $\theta = 9.0 \times 10^{-1}$, $\beta = 4.0$, $\mu_0 = 2.0$, $\mu_a = 4.0 \times 10^{-2}$, $\mu_c = 3.0 \times 10^{-2}$, $c_s = 2.0 \times 10^{-2}$, $c_p = 1.0 \times 10^{-2}$.

Figures 3 and 4 show the estimation results of the optimal software rejuvenation schedule for Model 1 and Model 2, respectively, where the failure time data is generated from the Weibull distribution $F_f(t) = 1 - e^{-(\frac{t}{\theta})^\beta}$ with shape parameter $\beta = 4.0$ and scale parameter $\theta = 0.9$. For 100 failure data points, we have $j^*/100 = 0.1170$ in Fig.3. The estimates of the optimal rejuvenation schedule and the minimum expected cost are $\hat{t}_0^* = 0.401221$ and $C_1(\hat{t}_0^*) = 0.000131508$, respectively. On the other hand, in Model 2, one estimates $\hat{t}_0^* = 0.30165$ and $C_2(\hat{t}_0^*) = 0.000132982$ from $j^*/100 = 0.0900$ (see Fig.4).

In the following section, we compare two models numerically in terms of cost minimization. Then, the dependence of model parameters in the respective optimal rejuvenation schedules is investigated in numerical examples. Also, we examine asymptotic properties for the statistical estimation algorithms using the simulation data.

6. Numerical examples

6.1. Performance comparison

In this section, we compare two cost models and carry out the sensitivity analysis on the model parameters. Figures 5 and 6 show the behavior of the expected costs for Model 1 and Model 2, respectively, where the failure time distribution is the Weibull distribution in Figs. 3 and 4. As the MTTF ($\lambda_f = \theta\Gamma(1 + 1/\beta)$, where $\Gamma(\cdot)$ denotes the standard gamma function) becomes larger for a fixed shape parameter, the optimal software rejuvenation schedule which minimizes the expected cost takes larger value for each case. Also, dependences of MTTF on the optimal software rejuvenation time and its associated cost value are investigated in Figs. 7 and 8, respectively. As the MTTF gets larger, the rejuvenation schedule becomes monotonically larger, but the minimum expected cost becomes smaller for both models. In particular, it is found that the rejuvenation time (expected cost) for Model 1 is always larger (smaller) than that for Model 2. Intuitively, if restarting the system after repair requires rejuvenation as in Model 2 system downtime due to a failure is higher than in the case without rejuvenation after repair (Model 1). Therefore, the rejuvenation schedule should be set at a smaller value so that the system stays in State 0 longer, and hence, the chance of a failure becomes smaller.

Table 1 presents the dependence of cost ratio c_s/c_p on the rejuvenation schedule. As the cost ratio c_s/c_p increases, the rejuvenation time monotonically decreases, but the cost value increases for both models. This monotone tendency can be also observed for other parameters. In Table 2, we examine the dependence of ratio μ_a/μ_c on the rejuvenation schedule. If the mean repair time becomes larger for fixed time to complete rejuvenation, the resulting optimal rejuvenation schedule can take rather smaller value. The final example in Table 3 presents the relationship between λ_f/μ_0 and the optimal rejuvenation schedule. As λ_f takes larger value with respect to μ_0, *i.e.* the system tends to be

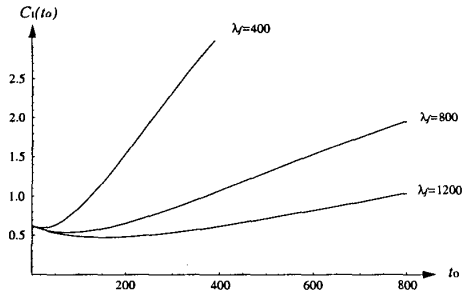

Figure 5. Behavior of expected cost in Model 1: $\mu_a = 0.6$, $\mu_c = 0.3$, $\mu_0 = 2.4 \times 10^2$, $\beta = 2.0$, $c_s = 5.0 \times 10^3$, $c_p = 5.0 \times 10^2$.

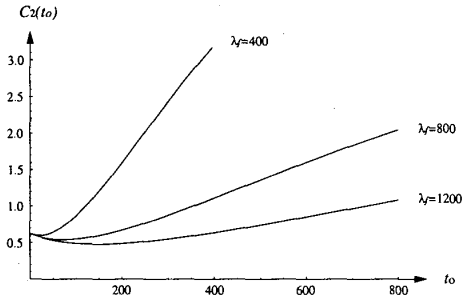

Figure 6. Behavior of expected cost in Model 2: $\mu_a = 0.6$, $\mu_c = 0.3$, $\mu_0 = 2.4 \times 10^2$, $\beta = 2.0$, $c_s = 5.0 \times 10^3$, $c_p = 5.0 \times 10^2$.

Table 1. Dependence of cost ratio c_s/c_p on the rejuvenation schedule: $\mu_a = 0.6$, $\mu_c = 0.3$, $\mu_0 = 2.4 \times 10^2$, $\beta = 2.0$, $\lambda_f = 21.6 \times 10^2$, $c_p = 5.0 \times 10^2$.

	Model 1		Model 2	
c_s/c_p	t_0^*	$C_1(t_0^*)$	t_0^*	$C_2(t_0^*)$
2	1207.10	0.1829	1013.40	0.2047
3	883.35	0.2230	788.58	0.2389
4	715.56	0.2529	657.23	0.2655
5	609.20	0.2767	568.87	0.2873
6	534.35	0.2968	504.44	0.3057
7	478.16	0.3139	454.89	0.3216
8	434.07	0.3288	415.35	0.3356
9	398.37	0.3420	382.91	0.3481
10	368.74	0.3538	355.72	0.3593
11	343.67	0.3645	332.52	0.3694

more reliable, t_0^* becomes larger and the optimistic preventive maintenance should be carried out.

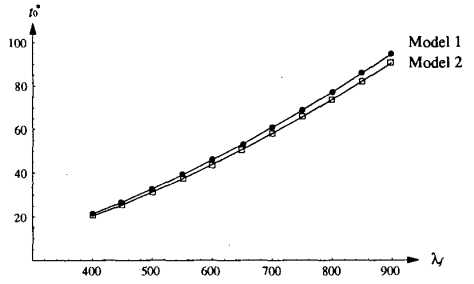

Figure 7. Dependence of MTTF on the optimal software rejuvenation time: $\mu_a = 0.6$, $\mu_c = 0.3$, $\mu_0 = 2.4 \times 10^2$, $\beta = 2.0$, $c_s = 5.0 \times 10^3$, $c_p = 5.0 \times 10^2$.

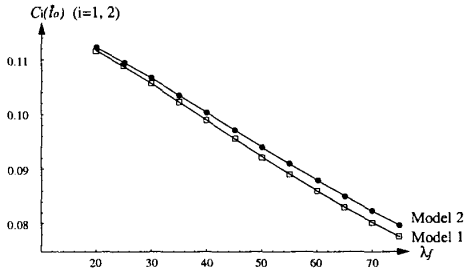

Figure 8. Dependence of MTTF in the minimum expected cost: $\mu_a = 0.6$, $\mu_c = 0.3$, $\mu_0 = 2.4 \times 10^2$, $\beta = 2.0$, $c_s = 5.0 \times 10^3$, $c_p = 5.0 \times 10^2$.

Table 2. Dependence of ratio μ_a/μ_c on the rejuvenation schedule: $\mu_c = 0.3$, $\mu_0 = 2.4 \times 10^2$, $\beta = 2.0$, $\lambda_f = 21.6 \times 10^2$, $c_s = 5.0 \times 10^3$, $c_p = 5.0 \times 10^2$.

	Model 1		Model 2	
μ_a/μ_c	t_0^*	$C_1(t_0^*)$	t_0^*	$C_2(t_0^*)$
2	368.74	0.3538	355.72	0.3593
3	272.29	0.3988	265.58	0.4023
4	218.11	0.4295	213.93	0.4321
5	182.81	0.4523	179.93	0.4543
6	157.76	0.4700	155.65	0.4716
7	138.98	0.4843	137.36	0.4855
8	124.33	0.4960	123.04	0.4970
9	112.55	0.5058	111.50	0.5067
10	102.86	0.5142	101.98	0.5150
11	94.739	0.5215	93.999	0.5221

6.2. Asymptotic behavior

Next, we examine the asymptotic properties of the estimators developed in Section 5. Of the most important problem in practical applications is the speed of convergence of the estimates for the optimal software rejuvenation sched-

Table 3. Dependence of ratio λ_f/μ_0 on the rejuvenation schedule: $\mu_a = 0.6$, $\mu_c = 0.3$, $\mu_0 = 2.4 \times 10^2$, $\beta = 2.0$, $c_s = 5.0 \times 10^3$, $c_p = 5.0 \times 10^2$.

	Model 1		Model 2	
λ_f/μ_0	t_0^*	$C_1(t_0^*)$	t_0^*	$C_2(t_0^*)$
2	30.23	0.5873	28.80	0.5889
3	63.82	0.5511	60.95	0.5540
4	105.42	0.5121	100.93	0.5160
5	152.51	0.4741	146.31	0.4788
6	203.34	0.4390	195.43	0.4408
7	256.78	0.4073	247.15	0.4126
8	312.08	0.3790	300.75	0.3844
9	368.74	0.3538	355.72	0.3593
10	426.41	0.3314	411.71	0.3368
11	484.85	0.3114	468.48	0.3167

ules. In other words, since a large number of sample failure time data points are not available in the earlier operational phase, it is significant to investigate the number of data at which one can estimate the optimal software rejuvenation schedule accurately without specifying the failure time distribution. Figures 9 and 10 illustrate the asymptotic behavior of the estimates for the optimal software rejuvenation schedule and its associated minimum expected cost, respectively, in Model 1, where the failure time data are generated from a Weibull distribution with shape parameter $\beta = 4.0$ and scale parameter $\theta = 9.0 \times 10^{-1}$.

In these figures, estimates of the optimal policy $\hat{t}_0^*$ and the corresponding minimum cost value $C_1(\hat{t}_0^*)$ are calculated in accordance with the estimation algorithm in Theorem 4, where the sample mean $\hat{\lambda}_f = \sum_{k=1}^{n} x_k/n$ changes as the failure time data is observed, where the horizontal line in the figures denotes the real optimal rejuvenation schedule and the minimum cost value ($t_0^* = 0.401221$ and $C_1(t_0^*) = 0.000131508$). From Figs. 9 and 10, it is seen that the estimate of the optimal rejuvenation schedule fluctuates until the number of observations is about 50. On the other hand, it is found that the expected cost per unit time in the steady state can be estimated accurately after the number of observations becomes about 20. These results enable us to use the non-parametric algorithm proposed here to estimate precisely the optimal software rejuvenation schedule under the incomplete knowledge of the failure time distribution. In Figs. 11 and 12, asymptotic behavior in Model 2 are presented, where the simulation data is based on the Weibull distribution with $\beta = 4.0$ and $\theta = 9.0 \times 10^{-1}$ and where the real optima are $t_0^* = 0.30165$ and $C_2(t_0^*) = 0.000132982$. The results are quite similar to Figs. 9 and 10.

7. Concluding remarks

In this paper, we have analyzed two generalized software rejuvenation models and developed a statistical non-parametric algorithms to estimate the optimal software rejuvenation schedules which minimize the expected costs per unit time in the steady-state. The resulting estimators for the optimal software rejuvenation schedules have quite nice convergence properties and are useful to apply to a real software operation without specifying the underlying failure time distribution. In fact, the measurement-based approach [7, 8] to perform the effective software rejuvenation requires much effort to measure the physical characteristics for the system. Also, the modeling-based approaches studied in the literature [10, 11, 12, 13, 14] can not explain the software aging phenomenon completely, since the underlying failure time distribution is unknown in many cases. The statistical approach developed in this paper is simple, but can guarantee the real optimal software rejuvenation schedule if the number of failure time data becomes large. Such an on-line estimation algorithm should be applied to other complex software systems, as the

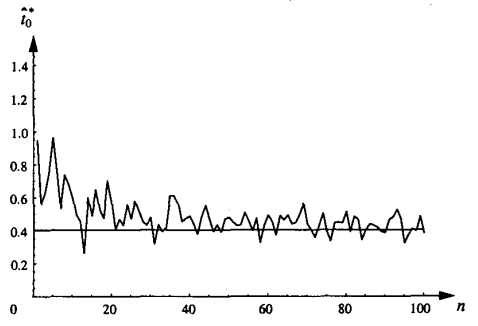

Figure 9. Asymptotic behavior of the optimal software rejuvenation schedule for Model 1: $\theta = 9.0 \times 10^{-1}, \beta = 4.0, \mu_0 = 2.0, \mu_a = 4.0 \times 10^{-2}, \mu_c = 3.0 \times 10^{-2}, c_s = 2.0 \times 10^{-2}, c_p = 1.0 \times 10^{-2}$.

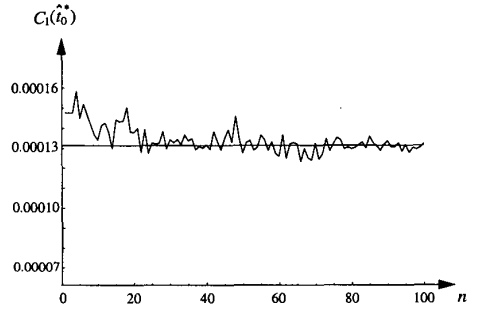

Figure 10. Asymptotic behavior of the minimum expected cost for Model 1: $\theta = 9.0 \times 10^{-1}$, $\beta = 4.0$, $\mu_0 = 2.0$, $\mu_a = 4.0 \times 10^{-2}$, $\mu_c = 3.0 \times 10^{-2}$, $c_s = 2.0 \times 10^{-2}$, $c_p = 1.0 \times 10^{-2}$.

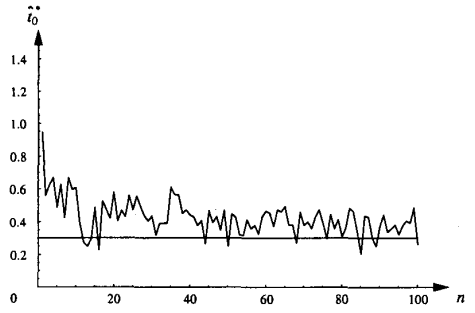

Figure 11. Asymptotic behavior of the optimal software rejuvenation schedule for Model 2: $\theta = 9.0\times10^{-1}$, $\beta = 4.0$, $\mu_0 = 2.0$, $\mu_a = 4.0\times10^{-2}$, $\mu_c = 3.0\times10^{-2}$, $c_s = 2.0\times10^{-2}$, $c_p = 1.0\times10^{-2}$.

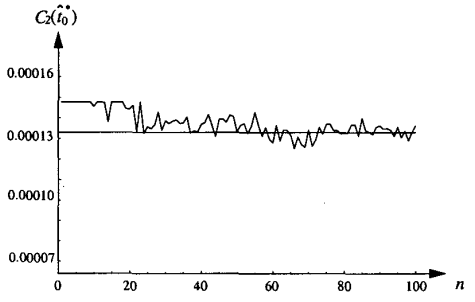

Figure 12. Asymptotic behavior of the minimum expected cost for Model 2: $\theta = 9.0\times10^{-1}$, $\beta = 4.0$, $\mu_0 = 2.0$, $\mu_a = 4.0\times10^{-2}$, $\mu_c = 3.0\times10^{-2}$, $c_s = 2.0\times10^{-2}$, $c_p = 1.0\times10^{-2}$.

transaction-based software systems. In the future, we plan to develop non-parametric estimation algorithms for other fault-tolerant software systems with rejuvenation.

References

[1] Y.Huang, C.Kintala, N.Kolettis, and N.D.Funton, "Software Rejuvenation: Analysis, Module and Applications", *Proc. 25th IEEE Int'l Symp. on Fault Tolerant Computing*, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 381-390.

[2] A.Avritzer, and E.J.Weyuker, "Monitoring Smoothly Degrading Systems for Increased Dependability", *Empirical Software Engineering*, **2**, 1997, pp. 59-77.

[3] E.Marshall, "Fatal Error: How Patriot Overlooked a Scud", *Science*, **255**, 1992, pp. 1347.

[4] J.Gray, and D.P.Siewiorek, "High-Availability Computer Systems", *IEEE Computer*, **9**, 1991, pp. 39-48.

[5] A.T.Tai, L.Alkalai, and S.N.Chau, "On-board Preventive Maintenance for Long-Life Deep-Space Missions: a Model-Based Analysis", *Proc. 3rd Annual IEEE Int'l Computer Performance & Dependability Symp.*, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 196-205.

[6] K.S.Trivedi, K.Vaidyanathan, and K.Goševa-Postojanova, "Modeling and Analysis of Software Aging and Rejuvenation", *Proc. 33rd Annual Simulation Symp.*, IEEE Computer Society Press, Los Alamitos, CA, 2000, pp. 270-279.

[7] S.Garg, A.Van Moorsel, K.Vaidyanathan, and K.S.Trivedi, "A Methodology for Detection and Estimation of Software Aging", *Proc. 9th Int'l Symp. on Software Reliability Eng.*, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 282-292.

[8] K.Vaidyanathan, and K.S.Trivedi, "A Measurement-Based Model for Estimation of Resource Exhaustion in Operational Software Systems", *Proc. 10th Int'l Symp. on Software Reliability Eng.*, IEEE Computer Society Press, Los Alamitos, CA, 1999, pp. 84-93.

[9] S.Garg, A.Puliafito, M.Telek, and K.S.Trivedi, "Analysis of Software Rejuvenation using Markov Regenerative Stochastic Petri Net", *Proc. 6th Int'l Symp. on Software Reliability Eng.*, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 24-27.

[10] S.Garg, Y.Huang, C.Kintala, and K.Trivedi, "Time and Load Based Software Rejuvenation: Policy, Evaluation and Optimality", *Proc. 1st Fault-Tolerant Symp.*, 1995, pp. 22-25.

[11] S.Pfening, S.Garg, A.Puliafito, M.Telek, and K.S.Trivedi, "Optimal Rejuvenation for Toleranting Soft Failure", *Performance Evaluation*, **27/28**, 1996, pp. 491-506.

[12] S.Garg, S.Pfening, A.Puliafito, M.Telek, and K.S.Trivedi, "Analysis of Preventive Maintenance in Transactions Based Software Systems", *IEEE Trans. Computers*, **47**, 1998, pp. 96-107.

[13] H.Okamura, A.Fujimoto, T.Dohi, S.Osaki, and K.S.Trivedi, "The Optimal Preventive Maintenance Policy for a Software System with Multi Server Station", *Proc. 6th ISSAT Int'l Conf. Reliability and Quality in Design*, 2000, pp. 275-279.

[14] A.Bobbio, and M.Sereno, "Fine Grained Software Rejuvenation Models", *Proc. 3rd IEEE Int'l Computer Performance & Dependability Symp.*, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 4-12.

[15] S.Garg, Y.Huang, C.Kintala, and K.S.Trivedi, "Minimizing Completion Time of a Program by Checkpointing and Rejuvenation", *Proc. 1996 ACM SIGMETRICS Conf.*, ACM Press, Cambridge, MA, 1996, pp. 252-261.

[16] R.E.Barlow, and R.Campo, "Total Time on Test Processes and Applications to Failure Data Analysis", *Reliability and Fault Tree Analysis* (R. E. Barlow, J. Fussell and N. D. Singpurwalla, eds.), SIAM, Philadelphia, 1975, pp. 451-481.

[17] T.Dohi, N.Kaio, and S.Osaki, "Optimal Checkpointing and Rollback Strategies with Media Failures: Statistical Estimation Algorithms", *Proc. 1999 Pacific Rim Int'l Symp. Dependable Comput.*, IEEE Computer Society Press, Los Alamitos, CA, 1999, pp. 161-168.