Analysis of a two-level software rejuvenation policy

Wei Xie a,*, Yiguang Hong b, Kishor Trivedi a

a Department of Electrical and Computer Engineering, Center for Advanced Computing and Communications (CACC), Duke University, Durham, NC 27708-0294, USA
b Institute of Systems Science, Chinese Academy of Sciences, Beijing 100080, China

Received 28 November 2003; accepted 21 February 2004

Abstract

A two-level rejuvenation policy for software systems with degradation process is studied. Both full restarts and partial restarts are considered in this rejuvenation strategy. A semi-Markov process model is constructed, and based on its closed-form solution we obtain the system availability as a bivariate function. Then, the rejuvenation policy is analyzed to maximize the system availability. Several different scenarios of software rejuvenation strategy are demonstrated by numerical examples.

Keywords: Software aging; Software rejuvenation; Semi-Markov process; System availability

1. Introduction

With the explosive growth in Internet technology and the emergence of a number of new and advanced online applications, assured quality of service and availability of e-business platforms has become a critical issue. The study of software reliability of the Web servers and application servers is needed because the consequences of software failure often cause enormous economic and reputational losses [1]. Software failures lead to more outages than hardware failures [2–4]. Furthermore, majority of these software failures occur due to error conditions that accumulate with time and use because of a variety of reasons, including the complexity of the software systems, the distributed nature and the large size of client groups. A class of software faults has been found to cause a steady accrual of error conditions in the internal state and the external operating environment of the executing software. This well-recognized phenomenon is known as software aging. Many researchers have reported software aging in different environments, which sometimes manifests itself as gradual performance degradation [2], and observed that the aging exists in both operating systems (OSs) and applications (e.g. Refs. [5,6]). Various causes of software aging include memory leaks, memory fragmentation, memory bloating, missing scheduling deadlines, broken pointers, poor register use, build-up of numerical round-off errors, and broken network connections or database connections.

Proactive detection or maintenance mechanisms in design and operation have acquired importance as techniques to improve Web server software reliability. One such technique is called software rejuvenation, first proposed in Ref. [5], which can be regarded as a preventive maintenance policy to prevent or postpone software failures. Rejuvenation usually needs to bring the system down for a short period of time to cleanup its internal state or its environment and restart the software. Over the past few years, many researchers have contributed to rejuvenation policy design. For example, the metric ‘estimated time to exhaustion’ was proposed for fault detection and software aging estimation in Ref. [7]. A time-based rejuvenation model using the semi-Markov approach and corresponding non-parametric statistical algorithm for the online estimation of software resource exhaustion were discussed in Ref. [8]. A fine-grained model to analyze rejuvenation policy has appeared (e.g. see Ref. [9]). Concurrently, the software industry has recognized the existence of software aging and the importance of software rejuvenation. In many professional-grade software such as the Apache server [10] and a number of commercial application software

* Corresponding author. Address: 143 Belaire Drive, Matawan, NJ 07747, USA.
E-mail addresses: wxie@ee.duke.edu (W. Xie); hong@ee.duke.edu (Y. Hong); kst@ee.duke.edu (K. Trivedi).

0951-8320/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
configured in production environment, simple rejuvenation schemes have been implemented in various ways to alleviate their aging problems.

To the best of our knowledge, most research effort in software rejuvenation found in the literature only considers single-level rejuvenation, usually only full restart actions, in spite of the fact that de facto multi-level rejuvenation policies have been used in practice (e.g. see Refs. [11,12]). In the crash-only software proposed by Candea and Fox [13], in which crash-recovery is cheap, microrebooting suspect components before they fail are discussed. Theoretical analysis of multi-level rejuvenation is more challenging than that of single-level rejuvenation. Based on certain deterministic assumptions, Hong et al. [14] provided a simple analysis approach for two-level rejuvenation policies implemented in Apache Webserver. Vaidyanathan et al. [15] discussed ‘minimal maintenance’ and ‘major maintenance’ while studying inspection-based preventive maintenance in operational software systems.

In this paper, we resort to a more complex but more realistic paradigm of semi-Markov processes (SMPs) to study the two-level rejuvenation policy, which consists of partial restarts and full restarts. By solving the model in closed-form, we obtain the steady-state system availability as a function of rejuvenation interval and partial restart probability. To maximize the system availability, all the possible scenarios and the corresponding rejuvenation strategies are analyzed. Specifically, for two popular time to failure distributions, we show that a multi-level rejuvenation policy is not necessarily better than a single-level rejuvenation policy in terms of optimizing overall availability. This result is consistent with that derived in Ref. [14] using a different method. Our model may be viewed as a generalization of the work in Ref. [8], where a semi-Markov chain is solved for a full-rejuvenation-only software rejuvenation policy.

The rest of the paper is organized as follows. In Section 2, the two-level software rejuvenation policy is introduced. Then, in Sections 3 and 4, a SMP model for the software rejuvenation policy is constructed and closed-form solution is obtained. Maximizing availability by selecting rejuvenation policy parameters is also elaborated. In Section 5, numerical results are presented and several examples are provided to illustrate different optimal rejuvenation scenarios for different system parameters. Section 6 presents the concluding remarks.

2. Software rejuvenation policy

Without any rejuvenation, both the OS software and the application software running on top of it degrade in performance with time due to the exhaustion of system resources such as free physical memory, and eventually crash. In other words, with no proactive rejuvenation action applied to the software system, the quality of service keeps decreasing until an unexpected outage occurs, which is very undesirable to high availability (HA) systems and fatal to mission-critical applications.

It is well known that memory leaks, unreleased object references, faulty pointer handling, dead database connections in connection pools, and round-off errors are some typical sources of software aging during software execution. Our major interest in this paper is with a comprehensive theoretical analysis of aging processes and rejuvenation policy. We do not focus on any particular aging metric and we will use free memory level as an example system attribute for convenience.

To make theoretical analysis simple and clear, we focus on the software rejuvenation with its actions offered at two levels similar to that discussed in Ref. [11]:

- **Level-1** is a service-level rejuvenation, where the application software is written in such a way that a stoppage of its service will save any necessary data and the following restart of the service will resume the operation in a more usable state. Level-1 rejuvenation is also called partial restart and we will use these two terms interchangeably in this paper. The impact of other applications running on the same OS is assumed to be minimum. For instance, restarting an application server can restore its broken database connections and some other resources back to a healthy state, and the Webserver, database gateways and server management utilities running on the same OS are not affected.

- **Level-2** is a box-level rejuvenation, where the OS and all services on the same machine have to be stopped and the rejuvenation action restarts the OS to recover all its free memory. Level-2 rejuvenation is also called full restart. This is a thorough rejuvenation, and it affects all applications (e.g. Webserver instances, application server instances, logging services, management services, CORBA servers) on this machine and naturally takes more time to complete.

If some resources are exhausted before the system is rejuvenated, a system crash occurs. Recovery from a system crash typically takes much longer because (1) it is unexpected and it takes time for the production support personnel to receive the alarm, come to the scene, and identify the problem, (2) lost data and interrupted transactions need to be properly handled, and (3) a full restart is needed.

Thus, there are three possible maintenance actions: 'partial restart' (or 'level-1 rejuvenation'), 'full restart' (or 'level-2 rejuvenation'), and 'reboot from crash'; the first two are proactive while the third one is reactive.

The SMP model for this problem is shown in the transition diagram of Fig. 1. Circles represent states and directed arcs represent transitions. The firing times of all the transitions are generally distributed, and their cumulative distribution functions (cdfs) are shown.
In the model by Huang et al. [5], a software instance has two available states in its lifecycle: the robust and failure-prone states. Also, Gurler and Kaya [16] classified the lifetime of each component in their multi-component system into four states. In this section, we study the semi-Markov model presented in Section 2 for the software system with two-level rejuvenation actions in detail.

We take the so-called two-stage method to solve the aforementioned semi-Markov model, which can be fully described by its kernel matrix $K(t)$ as follows [18]

$$
K(t) = \begin{pmatrix}
0 & k_{01}(t) & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & k_{12}(t) & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & k_{23}(t) & 0 & 0 & k_{26}(t) \\
0 & 0 & 0 & 0 & k_{34}(t) & k_{35}(t) & 0 \\
0 & k_{43}(t) & 0 & 0 & 0 & 0 & 0 \\
k_{50}(t) & 0 & 0 & 0 & 0 & 0 & 0 \\
k_{60}(t) & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

in which

$$
k_{ij}(t) = P[Y_1 = j, T_i = t | Y_0 = i], \quad i, j \in \Omega,
$$

3. Semi-Markov model analysis

For convenience, the seven SMP states are numbered sequentially as shown in Fig. 1. The state space can be denoted as $\Omega = \{0, 1, 2, 3, 4, 5, 6\}$. To cover real life scenarios, we consider general distributions, i.e., both exponential and non-exponential distributions, of the transitions between different states. When the software in question is in state U (or state M), the SMP will enter state M (or state L) when the transition with a general distribution function $F_1(t)$ (or $F_3(t)$) occurs. When it is in state L, it may enter state D with a general distribution function $F_1(t)$ (inspection begins), or fail with a distribution function $F_4(t)$ (enters state B). From state D, it may enter state P with a general distribution function $F_6(t)$ (for partial restarts, or move to state R with a general distribution function $F_{15}(t)$ for full restarts. In practice, the sojourn time in state D is very short in comparison with the sojourn times in other states. If the decision is made instantly, $\delta = 0$ and transitions from state D to state P and from D to R may be viewed as instantaneous transitions with branching probabilities, and the analysis of this type of SMPs can be found in Ref. [17]. The distribution function for the duration of full restarts is $F_{15}(t)$ (in state R), while the function for that of partial restarts is $F_{15}(t)$ (in state P). When the software is in state B, the system is being repaired after a crash. The crash repair time has a distribution function of $F_{15}(t)$. Since each state is a regenerative state, the underlying stochastic process is a SMP [18]. The model proposed in Ref. [8], where only full restart actions were considered, is a special case of Fig. 1.

**State U (Up).** This is the highly efficient and highly robust software execution phase. Both a full restart and a reboot after a crash bring the system back to this state. The system works perfectly in this state. It is available in this state.

**State M (Medium-efficient).** This is the medium-efficient software execution phase. Resource degradations start to occur in the OS software but they are not a threat yet. Partial restarts will take the system back to this state. The system still works well in this state. This is also an available state.

**State L (Low-efficient).** In this state, the software is running at a low-efficient execution phase or alert phase. The OS and some application software are in the failure-prone state but it is still available. A rejuvenation action is needed or a system crash may happen.

**State D (Decision).** In this state, the system is taken offline for rejuvenation, where it is determined which level of rejuvenation is appropriate according to current system conditions. Usually the decision is made very quickly, i.e., the sojourn time in this state is very short. The system is unavailable to users.

**State P (Partial rejuvenation).** In this state, the level-1 restart is in progress and the software in question is unavailable to its clients.

**State R (Full rejuvenation).** In this state, the level-2 restart is in progress, and the system is unavailable.
where \([\{T_n, S_n\}, n \geq 0\] is the underlying Markov renewal sequence of random variables\(^1\). In other words, \(k_0(t)\) is the probability that if the SMP has just entered state \(i\), the next transition occurs within time \(t\) and the next state is state \(j\). Therefore, the non-zero elements of \(K(t)\) could be derived as follows:

\[
k_{01}(t) = F_1(t), \quad k_{12}(t) = F_2(t),
\]

\[
k_{23}(t) = \int_0^t F_3(x) dF_1(x), \quad k_{26}(t) = \int_0^t F_1(x) dF_3(x),
\]

\[
K_{34}(t) = \int_0^t F_4(x) dF_2(x), \quad k_{35}(t) = \int_0^t F_2(x) dF_3(x),
\]

\[
k_{41}(t) = F_1(t), \quad k_{50}(t) = F_5(t), \quad k_{60}(t) = F_6(t),
\]

where \(F_i(t) = 1 - F_i(t)\) is the complementary distribution function for any transition \(i \in \{1, 2, 3, 4, 5, 6\}\).

In practice, the time to trigger rejuvenation is typically a fixed duration \([8]\), i.e. its cdf has the form of \(F_1(t) = u(t - \tau)\), where \(u(t)\) is the unit step function, and \(r\) is the time to trigger rejuvenation.

We are ready to study the steady-state availability when the kernel is given. Following the two-stage method of SMP analysis \([18]\), let \(K(\infty) = \lim_{t \to \infty} K(t)\) be the one-step transition probability matrix of the embedded Markov chain (EMC) of the SMP

\[
K(\infty) = \begin{pmatrix}
0 & p_{01} & 0 & 0 & 0 & 0 \\
0 & 0 & p_{12} & 0 & 0 & 0 \\
0 & 0 & 0 & p_{23} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
p_{50} & 0 & 0 & 0 & p_{54} & p_{55} \\
p_{60} & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

where \(p \triangleq k_3(\infty)\), and \(p_{26} = F_5(r)\).

Solving the EMC steady-state equations \(v = v \cdot K(\infty)\) and \(ve^T = 1\), where row vectors \(v = (v_0, v_1, \ldots, v_6)\) and \(e = (1, 1, 1, \ldots, 1)\), we obtain the steady-state probabilities of the EMC for the SMP as

\[
v = \frac{1}{D(r, p)} \begin{pmatrix}
1 - pF_1(r) \\
1 - pF_1(r) \\
F_3(r) \\
0 \\
0 \\
0 \\
0 \\
\end{pmatrix},
\]

in which \(D(r, p) = 5 - pF_3(r) + pF_3(r)\).

According to Ref. \([18]\), the steady-state probability of each SMP state is

\[
\pi_i = \frac{v_i}{v^T X}, \quad i \in \Omega,
\]

where \(\chi_i\) is the expected sojourn time of state \(i\). It is straightforward to obtain the expected sojourn times in state 0 and state 1 as \(\chi_0 = \int_0^\infty F_1(t) dt\), \(\chi_1 = \int_0^\infty F_2(t) dt\), respectively, while that in state 2 is \(\chi_2 = \int_0^\infty F_1(t) F_3(t) dt = \int_0^\infty F_3(t) dt\). The expected sojourn times in states 3, 4, 5, and 6 are given as \(\chi_3 = \int_0^\infty F_4(t) F_5(t) dt\), \(\chi_4 = \int_0^\infty F_5(t) dt\), \(\chi_5 = \int_0^\infty F_6(t) dt\), \(\chi_6 = \int_0^\infty F_6(t) dt\). Note that the total mean times to complete the level-1 and level-2 rejuvenations are \(\chi_4 = \chi_3 + \chi_4\) and \(\chi_5 = \chi_3 + \chi_5\), respectively. Define \(t_i \triangleq \chi_i\) for \(i = 0, 1, 2, 3, 6\).

For the rest of this paper, we make the following reasonable assumption

\[
t_4 > t_5 > t_6,
\]

which indicates that the mean time to reboot from crash is strictly larger than the mean time to complete the level-2 (full restart) rejuvenation, which in turn is strictly larger than the mean time to complete the level-1 (partial restart) rejuvenation.

4. Availability optimization

In our model, states 0, 1, and 2 are the only available states as described in Section 2, hence the steady-state system availability as a function of parameters \(r\) and \(p\) can be written as (from Eqs. (2) and (3), and \(t_i, i \in \Omega\))

\[
A(r, p) = \pi_0 + \pi_1 + \pi_2 = \frac{S(r, p)}{S(r, p) + V(r, p)},
\]

where

\[
\begin{align*}
S(r, p) & \triangleq (1 - p + pF_3(r)) t_0 + t_1 + \int_0^r \tilde{F}_3(t) dt, \\
V(r, p) & \triangleq (p - pF_3(r)) t_5 + (1 - p - F_3(r)) + pF_3(r) t_5 + F_3(r) t_6.
\end{align*}
\]

Note that the first-order partial derivatives of \(S(r, p)\) and \(V(r, p)\) are

\[
\frac{\partial S(r, p)}{\partial r} = pt_0 F_3(r) + \tilde{F}_3(r) \triangleq S',
\]

\[
\frac{\partial V(r, p)}{\partial r} = ((t_5 - t_4)p + t_6 - t_5) F_3(r) \triangleq V'.
\]
optimal values of \( r \), three categories of widely used general distributions: Dohi’s model \([8]\), i.e. there is no level-1 rejuvenation for any \( r \) will be used in the sequel.

We are interested in selecting the time to trigger rejuvenation \( r \) and level-1 rejuvenation probability \( p \) to maximize the system availability \( A(r, p) \).

A special case is \( r = \infty \), or there is no rejuvenation at all. Under this condition no matter what value \( p \) takes, the availability is the same

\[
A(\infty) \triangleq A(\infty, p) = \frac{t_0 + t_1 + \int_0^{\infty} F_3(t) \, dt}{t_0 + t_1 + \int_0^{\infty} F_3(t) \, dt + t_6},
\]

for any \( 0 \leq p \leq 1 \). The numerator of \( A(\infty) \) is the sum of the mean sojourn times of states U, M, and L, and the denominator is the sum of the mean sojourn times of states U, M, L, and B.

Denote the instantaneous failure rate (also known as hazard rate) \([19]\) of the distribution \( F_3(t) \) as \( h(r) = F_3'(r)/F_3(r) \) with \( F_3'(r) = \frac{d F_3(r)}{d r} \). In this paper we consider three categories of widely used general distributions: \( F_3(t) \) is an increasing failure rate (IFR, i.e. \( h(r) > 0 \), for any \( r \geq 0 \)) distribution; \( F_3(t) \) is a decreasing failure rate (DFR, i.e. \( h(r) < 0 \), for any \( r \geq 0 \)) distribution; \( F_3(t) \) is a constant failure rate (CFR, i.e. \( h(r) = 0 \), for any \( r \geq 0 \)). Note that the exponential distribution is the only distribution with a CFR \([20]\). IFR distributions are often used to describe wearout failures and DFR distributions are used to capture reliability growth.

Generally, for a multi-variable function, its Hessian matrix is needed to determine the local extrema and saddle points. However, owing to the specifics in our case, we take a simpler approach to achieve the same goal. We will enumerate all \( (r, p) \) pairs that are at boundaries and/or the first order partial derivatives are zeros, and find the pair, which gives the maximum availability. In what follows, we need to consider the combinations of \( r = 0 \) (i.e. immediate rejuvenation), \( r = \infty \) (i.e. no rejuvenation), \( 0 < r < \infty \), and \( p = 0 \) (i.e. only level-2 rejuvenation), \( p = 1 \) (i.e. only level-1 rejuvenation), and \( 0 < p < 1 \). We denote the optimal values of \( p \) and \( r \) as \( p^* \) and \( r^* \).

Case 1 \((p = 0)\). In this case, our model degrades to Dohi’s model \([8]\), i.e. there is no level-1 rejuvenation (partial restart) at all. To obtain the optimal time to trigger rejuvenation \( r^* \), we solve the equation \( A'(r, 0) = 0 \), which is

\[
\frac{dA(r, p)}{dr} \bigg|_{p=0} = S'(S + V) - S(S' + V') \bigg|_{p=0} = 0
\]

Because \( S'(r, 0) > 0 \) for \( r < \infty \), we have \( (VS' - V'S)|_{p=0} = 0 \) which is equivalent to

\[
q_0(r) \triangleq V(r, 0) - (t_6 - t_5)h(r)S(r, 0) = 0.
\]

Note that the first order derivative of \( q_0(r) \) is \( q_0'(r) = -(t_6 - t_5)h(r) \).

- \( h(r) \) is an IFR (i.e. \( h'(r) > 0 \)); \( q_0'(r) < 0 \) for \( 0 \leq r \leq \infty \) by the assumption in Eq. (4). If \( q_0(0) > 0 \) and \( q_0(\infty) < 0 \), there is one and only one \( r_0^* \) such that \( q_0(r_0^*) = 0 \). Moreover, because \( q_0(r) < 0, A'(r_0^*, 0) = q_0'^*(S + V)^2/S'|_{p=0=r_0^*} < 0 \) (i.e. \( A(r, 0) \) is convex with respect to \( r \) at \( r = r_0^* \)). The corresponding availability is

\[
A(r_0^*, 0) = \frac{1}{(t_6 - t_5)h(r_0^*) + 1}.
\]

If \( q_0(0) \leq 0 \), the optimal value is at \( r_0^* = 0 \), with corresponding availability

\[
A(0, 0) = \frac{t_0 + t_1}{t_0 + t_1 + t_6}.
\]

It is obvious that the numerator of \( A(0, 0) \) is the mean sojourn time of state U,M, and L (0 in this case). The denominator is the sum of the mean sojourn times of states U, M, L, and B.

Case 2 \((p = 1)\). In this case, there is no full restart action. To obtain the optimal value of \( r \), we solve the equation \( A'(r, 1) = 0 \). Similar to Case 1, we can consider its equivalent form:

\[
q_1(r) \triangleq V(r, 1) - \frac{(t_6 - t_5)h(r)}{t_0h(r) + 1}S(r, 1) = 0.
\]

The first order derivative of \( q_1(r) \) is

\[
q_1'(r) = -\frac{(t_6 - t_5)h'(r)}{(1 + t_0h(r))^2}.
\]

- \( h(r) \) is an IFR: by the assumption in Eq. (4), \( q_1'(r) < 0 \) for \( 0 \leq r \leq \infty \). Therefore, if \( q_1(0) > 0 \) and \( q_1(\infty) < 0 \), there exists one and only one \( r_1^* \) so that \( q_1(r_1^*) = 0 \). Moreover, because \( q_1'(r) < 0, A'(r_1^*, 1) = q_1'^*(S + V)^2/S'|_{p=1=r_1^*} < 0 \) (i.e. \( A(r, 1) \) is convex at \( r = r_1^* \)). The corresponding
availability is
\[ A(r^1_1, 1) = \frac{t_0h(r^1_1) + 1}{(t_0 - t_4)h(r^1_1) + t_0h(r^1_1) + 1}. \]  
(18)

If \( q_1(0) \leq 0 \), the optimal is at \( r^1_1 = 0 \) with its availability
\[ A(0, 1) = \frac{t_1}{t_1 + t_4}. \]  
(19)

It is obvious that the numerator of \( A(0, 1) \) is the mean sojourn time of state \( M \) and \( L \) (0 in this case), and the denominator is the sum of the mean sojourn times of all states. If \( q_1(\infty) \geq 0 \), the optimal is at \( r^2_1 = \infty \) with its availability \( A(\infty) \) in Eq. (12).

- \( h(r) \) is a DFR: \( q_1(r) > 0 \). Even if there is a \( \tilde{r}_1 \) such that \( q_1(\tilde{r}_1) = 0 \), from \( A(\tilde{r}_1, 1) > 0 \) (i.e. \( A(r, 1) \) is concave \( r = \tilde{r}_1 \)), the optimal value of \( r \) is still either at \( r^1_1 = 0 \) or \( \infty \), and the corresponding availability will be in the form of either Eq. (19) or Eq. (12).

- \( h(r) \) is a CFR: \( q_0(r) = 0 \). \( A(r, 1) \) is a straight line and the optimal value of \( r \) to optimize the availability (when \( p = 0 \)) is found at either boundaries, i.e. either \( r^1_1 = 0 \) or \( r^1_1 = \infty \), same as the DFR case described above.

Case 3 \((0 < p < 1)\). In this case, both full restart and partial restart are possible. If \( r = \infty \), the availability equals to \( A(\infty) \) as discussed above. Therefore, here we only consider finding the optimal value of \((r, p) \in (0, \infty) \times (0, 1)\). We need to solve the equations
\[ \frac{\partial A(r, p)}{\partial r} = 0, \quad \frac{\partial A(r, p)}{\partial p} = 0, \]
which are equivalent to
\[
\begin{aligned}
q_3(r, p) &= V(r, p) - \frac{(t_5 - t_4)p + t_6 - t_5}{pt_0h(r) + 1}h(r)S(r, p) = 0, \\
q_4(r, p) &= V(r, p) - \frac{t_5 - t_4}{t_0}S(r, p) = 0,
\end{aligned}
\]
(20)
\[
\begin{aligned}
(pt_0h(r) + 1)V(r, p) - (t_6 - t_5 + (t_5 - t_4)p)h(r)S(r, p) = 0, \\
(1 - F_3(r))t_0V(r, p) - (t_5 - t_4)S(r, p) = 0.
\end{aligned}
\]
(21)

Since \( r < \infty \), \( 1 - F_3(r) \neq 0 \) (because \( h(r) \) is an IFR or a DFR or a CFR), the second equation of Eq. (21) becomes \( V(r, p) = (t_5 - t_4)S(r, p)t_0 \). Substituting this into the first equation of Eq. (21), we get
\[
\begin{aligned}
h(r^*) &= \frac{t_5 - t_4}{t_0(t_6 - t_5)} , \\
0 &= t_0t_5 + (t_4 - t_5)(t_0 + t_1) + t_0(t_6 - t_5)F(r^*) \\
+ (t_4 - t_5) r^* \int_0^{r^*} F_3(t)dt.
\end{aligned}
\]
(22)
Note that the variable \( p \) disappeared in Eq. (22). This means there are two different equations, denoted by Eq. (22), for a single variable \( r^* \). We have the following two possibilities:

- Generally, there is no common solution to these two equations. This implies that Case 3 does not apply and the optimal availability is obtained at the boundary values of \( p \) or \( r \). For \( p^* = 0 \) or 1 cases, refer to Cases 1 and 2. For \( r = 0, S = (1 - p)t_0 + t_1 \) and \( V = pt_4 + (1 - p)t_5 \). We have
\[
\begin{aligned}
A(0, p) &= \frac{t_0 + t_4}{t_0 + t_1 + t_5 - (t_0 + t_5 - t_4)p} , \\
\frac{\partial A(0, p)}{\partial p} &= \frac{t_1(t_5 - t_4) - t_0t_4}{(S(0,p) + V(0,p))^2}.
\end{aligned}
\]
(23)
(24)
The numerator on the right hand side of Eq. (24) is a constant and independent of \( p \), which means that the optimal value of \( p \) can be found at \( p^* = 0 \) or 1 whether or not expression (24) equals 0. Whether \( p^* = 0 \) or 1 directly depends on the values of parameters \( t_0, t_1, t_2, t_3 \) and \( t_6 \) and then the optimal availability is either \( A(0, 0) \) in Eq. (16) or \( A(0, 1) \) in Eq. (19).

- Since \( h(r) \) is either an IFR or a DFR or a CFR, the first equation of Eq. (22) has at most one solution. Therefore, if there is one \( r^* \) that happens to satisfy both equations in Eq. (22)
\[
r^* = h^{-1} \left( \frac{t_5 - t_4}{t_0(t_6 - t_5)} \right),
\]
if we denote the inverse function of \( h \) as \( h^{-1} \).
The corresponding availability will be
\[
A(r^*, p) = \frac{t_0}{t_0 + t_5 - t_4},
\]
(25)
which is independent of \( p \).

To summarize all the cases discussed above, we can find the maximum availability at either \( p = 0 \) or 1 (considering \( h(r) \) is either an IFR or a DFR or a CFR and Eq. (4) holds), which is similar to the result obtained in Ref. [14] using a different set of assumptions.

The overall maximum availability is
\[
A_{\max} = \max(A(r^*_0, 0), A(r^*_1, 1), A(0, 0), A(0, 1), A(\infty)).
\]
(26)
Ignore \( A(r^*_0, 0) \) (or \( A(r^*_1, 1) \)) in Eq. (26) if \( r^*_0 \) (or \( r^*_1 \)) does not exist.

Table 1
Default parameters unless otherwise specified

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>2</td>
<td>Shape parameter, IFR</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>0.7854 day(^{-1} )</td>
<td>Decided by ( t_2 ) and ( \alpha )</td>
</tr>
<tr>
<td>( t_0 )</td>
<td>1 day</td>
<td>Mean time in state 0</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>1 day</td>
<td>Mean time in state 1</td>
</tr>
<tr>
<td>MTTF</td>
<td>1 day</td>
<td>Mean of ( F_3(t) )</td>
</tr>
<tr>
<td>( t_6 )</td>
<td>8 h</td>
<td>Mean crash repair time</td>
</tr>
</tbody>
</table>
Our model is closely related to that in Ref. [15], which is an inspection-based preventive maintenance analysis with two levels of maintenance, based on a Markov regenerative process (MRGP) model. Exponential distributions are assumed between system deterioration states, and it concentrates on the optimal values of inspection interval.

5. Numerical results

In this section, several cases are studied numerically to illustrate the theoretical analysis given in the preceding sections.

The failure distribution $F_3(t)$ is chosen as a Weibull distribution, which is commonly used to describe all the three phases of the mortality curve (DFR, CFR, and IFR phases) [8,19]. The distribution function is given by

$$F_3(t) = 1 - e^{-\lambda t^\alpha}, \quad t \geq 0, \quad \lambda > 0, \quad \alpha > 0,$$

and the failure rate (hazard rate) by

$$h(t) = \lambda \alpha t^{\alpha-1}.$$

It is well known that if $\alpha = 1$, it is a CFR, if $\alpha < 1$, it is a DFR, and if $\alpha > 1$, it is an IFR. According to the discussions in Section 4, we can find the maximum availability at $p = 0$ or 1.

The expected sojourn time of state 2 can be re-written as

$$t_2 = \int_0^r F(r) dr = \int_0^r e^{-\lambda t^\alpha} dt = \frac{\Gamma(1/\alpha)}{\alpha \Gamma(a/\alpha)} \Gamma(\lambda r^\alpha, 1/\alpha)$$

where the incomplete and complete Gamma functions are defined as

$$I(x, \alpha) = \frac{1}{\Gamma(\alpha)} \int_0^x t^{\alpha-1} e^{-t} dt,$$

$$\Gamma(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt.$$

In the following, we will show three examples to demonstrate three different cases: (1) $p^* = 0$, (2) $p^* = 1$, and (3) $p^*$ is any $p$ for different sets of parameters. Note that these three examples should not be confused with the three cases discussed in Section 4, and we do not intend to repeat the theoretical analysis procedures shown there numerically step-by-step. The default values used in these numerical examples are shown in Table 1, unless otherwise specified. All the parameters we choose are for demonstration purposes and they do not come from real life systems.

(1) Example 1. $A_{\text{max}} = A(r_p^0, 0)$. The global maximum availability is found when $p^* = 0$ given these parameters:

$$t_4 = 4 \text{ h}, \quad t_5 = 5 \text{ h}.$$  

Fig. 2 shows the respective changes of the optimal rejuvenation time $r_p$ and the corresponding system availability $A$ versus the mean of failure distribution $F_3(t)$.

$$F(\alpha) = \int_0^\infty t^{\alpha-1} e^{-t} dt.$$  

(28)

FIG. 2. Effects of the MTTF on the optimal rejuvenation time.

FIG. 3. 3D plot of availability: $p = 0$ is optimal.
conventionally called mean time to failure or MTTF) for $p = 0$ and 1, represented by solid and dashed lines, respectively. Solving Eq. (14) (Eq. (17)), we may obtain the optimal time to rejuvenation $r^*_p(r^*_1)$, and then from Eq. (15) (Eq. (18)), we may get the corresponding system availability $A(r^*_0, 0)$ ($A(r^*_1, 1)$). With the set of parameters given in Eq. (29), we can find that the maximum availability will be achieved at $p = 0$ with $r^*_0 \approx 0.5$ day, or doing a full restart every 12 h, for an MTTF of 1 day. The corresponding maximum availability is approximately 0.915. This result can be confirmed by Fig. 3, the 3D diagram of $A$ versus $r$ and $p$. The availability diagram in Fig. 2 is the special case in Fig. 3 when $p = 0$ and 1. It clearly shows that the ridge of the availability surface is going up monotonically while $p$ decreases and the maxima of $A$ is reached at $p = 0$, and $A$ does not change with $p$ when $r \to \infty$.

(2) Example 2. $A_{\text{max}} = A(r^*_1, 1)$. The global maximal availability is found when $p^* = 1$ given these parameters:

$$t_4 = 1 \text{~h}, \quad t_5 = 6 \text{~h}.$$  \hfill (30)

Fig. 4 shows the respective changes of the optimal rejuvenation interval $r^*$ and the corresponding system availability $A$ versus the mean of failure distribution $F_3(t)$ for $p = 0$ and 1. In this example, only $t_4$ and $t_5$ are different and we obtain the totally different results, as shown in Fig. 5. The ridge of the availability surface is going up monotonically while $p$ increases and the maximum availability is reached at $p = 1$ with $r^*_1 \approx 0.8$ day, or doing a full restart roughly every 20 h for an MTTF of 1 day.

(3) Example 3. $A_{\text{max}} = A(r^*_0, 0) = A(r^*_1, 1)$. We take

$$t_4 = 2 \text{~h}, \quad t_5 = 3.77 \text{~h}.$$  \hfill (31)

Fig. 6 shows the respective changes of the optimal rejuvenation interval $r^*$ and the system availability $A$ versus the mean of failure distribution $F_3(t)$ in two cases: $p = 0$ and 1. Fig. 7 shows that the maximum availability will be achieved at any $p$ with a optimal value $r^* \approx 0.5$ day. Note that, because it is mathematically formidable to prove two different equations have a common solution, and the numerical results in Fig. 7 just illustrates that for some particular sets of parameters, the maximum availability for any $p$ are the same given a small tolerance. The ridge of the availability surface is rather flat for different values of $p$.

These examples clearly showed that the two-level rejuvenation strategy is highly dependent on the system parameters, and the optimal rejuvenation interval $r^*$ and level-1 rejuvenation probability $p^*$ could be dramatically different for different scenarios. The Website administrators and operators should gather their system information and make the two-level rejuvenation policy accordingly.

6. Conclusions

Software aging is recognized by both the academic and industry communities as a major obstacle for achieving HA systems. Many researchers have attempted to systematically study different rejuvenation policies. To date, most of them concentrated on single-level rejuvenation, which is...
relatively simple to analyze but unrealistic. This paper has addressed a two-level software rejuvenation policy for aging in software systems.

SMP is employed in the study and corresponding theoretical availability analysis is provided in closed-form. To determine the optimal rejuvenation policy that maximizes the system availability, all possible scenarios are elaborated. Specifically, we discussed in detail for three popular categories of failure distributions: IFR, DFR, and CFR. We found that in the context defined in this paper, the maximum availability is achieved by either pure level-1 rejuvenation or pure level-2 rejuvenation. A mixed rejuvenation policy does not perform better than the pure ones. This result is consistent with Ref. [14]. Several numerical examples were presented to demonstrate different optimal rejuvenation policies with different system parameters. It was clearly shown that for some sets of system parameters, pure level-1 rejuvenation is the best, and for some sets of parameters, pure level-2 rejuvenation is the best, and for some sets of parameters, it does not matter whether a mixed rejuvenation policy or a pure rejuvenation policy is used. Although, in reality, the failure rates may not be strictly IFR or DFR or CFR during their entire life cycles and partial restart actions may be more complex than what we have assumed, the results obtained in this paper are still helpful for system administrators and operators to understand the two-level rejuvenation and customize their own optimal rejuvenation policies.

Acknowledgements

This research was supported in part by DARPA and US Army Research Office under Award No. C-DAAD19 01-1-0646, and in part by the Air Force Office of Scientific Research under MURI Grant no. F49620-00-1-0327, and in part by NSF 60221301 and 60334040 of China. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and does not necessarily reflect the view of the sponsoring agencies.

References

Symposium on Software Reliability Engineering, Paderborn, Germany, 1998.


