A CTMDP Modeling for Multi-Stage Software Aging and Rejuvenation

Nianqiu Wang
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
wang.nianqiu@sd.cs.tsukuba.ac.jp

Fumio Machida
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
machida@cs.tsukuba.ac.jp

Abstract—**The seminal work on the software rejuvenation model used a continuous-time Markov chain to analyze the effectiveness of software rejuvenation for improving system availability. The effectiveness in terms of steady-state availability is determined by a condition on the transition rates, known as the rejuvenation threshold. However, such a theoretical condition has not been studied for software aging models with more than two stages. This paper formulates the software rejuvenation decision problem using a continuous-time Markov decision process (CTMDP) and theoretically demonstrates the conditions to determine the optimal rejuvenation policy, maximizing the steady-state availability for the three-stage software aging model. Furthermore, we develop an approximated policy evaluation algorithm for the CTMDP-based software aging and rejuvenation model that allows us to evaluate the steady-state availability of the system with a given decision policy. Through a numerical study, we confirm that the results of the policy evaluation algorithm correctly fit the boundary conditions of the optimal policy for the three-stage model. Our numerical results also demonstrate the boundary conditions for models with more than four stages.**

Index Terms—**Optimal rejuvenation policy, software aging, software rejuvenation, system availability, CTMDP**

I. Introduction

Software aging is a commonly known phenomenon observed in long-running software systems. Software aging is generally caused by aging-related software faults that are overlooked during software development and testing phases. With a long-time software execution, aging-related errors can accumulate over time and eventually cause a system failure [1]. To mitigate the problem of software aging, researchers and engineers employ software rejuvenation, which proactively releases operating system resources [2]. The first software aging and rejuvenation model was developed using a Continuous-Time Markov Chain (CTMC), consisting of four states that represent 2-stage software aging (i.e., one aging state followed by a failure state). The CTMC model enables the analysis of the condition to determine the software rejuvenation that can improve the system availability by rejuvenation at an aging state. However, the software aging can progress with multi-stages that cannot be simply represented by the 2-stage software aging model. For example, Bao *et al.* [3] considered a system with resource leaks where software aging progresses in multiple stages.

To determine the optimal timing to decide the software rejuvenation in a multi-stage software aging system, existing studies used Markov decision processes (MDPs). Pfening *et al.* [4] developed MDPs for determining the optimal rejuvenation time for server-type software. The system is a multi-stage model with fixed time intervals. They developed two different policies to decide whether the system should continue in service at any stage through MDP. Okamura *et al.* [5] presented an MDP formulation for a multi-degree degradation level system with software aging and rejuvenation. The policy iteration algorithm is used to derive the optimal rejuvenation policy. Eto *et al.* [6] formulated a semi-Markov decision process for a multi-states service degradation model. A recent study also employs MDP to derive the optimum rejuvenation policy for resiliency enforcement of heterogeneous network clusters [7]. In contrast to these existing studies, our study aims to investigate the boundary conditions of optimal rejuvenation policies for maximizing steady-state availability in multi-stage software aging and rejuvenation models.

In this paper, we use a Continuous-Time Markov Decision Process (CTMDP) to formulate a multi-stage software aging system where rejuvenation can be determined in each stage. Since the rejuvenation decision is associated with aging states, the problem is formulated as a decision problem. In contrast to MDP, CTMDP can formulate the time spent in a state, which is necessary in the analysis of availability. Given a decision policy that assigns an action to each state, a CTMDP can be transformed into the CTMC. We can derive the optimal policy of the CTMDP that can maximize the system availability by comparing the steady-state availabilities computed from the CTMCs resulting from given rejuvenation policies. Through this direct comparison approach, we theoretically show the boundary conditions of optimal rejuvenation policies for the three-stage software aging and rejuvenation model. According to our parameterization of the model, the optimal rejuvenation policy can be characterized by the difference between the failure recovery rate and the rejuvenation rate. We find that a larger difference tends to encourage an earlier decision of rejuvenation, while a smaller difference discourages rejuvenation action.

The direct comparison approach is not efficient for more than three-stage software aging models. Therefore, we develop a policy evaluation algorithm for CTMDPs that approximately estimates the steady-state availability achieved by a given rejuvenation policy. To compute the approximated availability, we introduce two reward functions for CTMDP to evaluate the total uptime and total runtime. While we cannot obtain the exact steady-state availability by a finite number of iterations, we can compare different policies with the approximated availability values computed from the accumulated uptime divided by the total runtime and determine the optimal policy that maximizes the availability. Through the numerical analysis, we visualized the boundary conditions of the optimal policy that correctly fit the theoretical boundaries for the three-stage software aging model. We also show the results of the models with more than four stages and discuss the tendencies of the optimal rejuvenation policies for multi-stage software aging models.

The rest of the paper is organized as follows. Section II reviews related work. Section III introduces CTMDP and shows how the seminal two-stage rejuvenation model can be formulated by a CTMDP. Section IV explains the CTMDP formulation for the three-stage software aging model. Based on the steady-state analysis of the CTMCs, we show the theoretical boundary conditions of the optimal rejuvenation policy. In Section V, we propose the policy evaluation algorithm to numerically derive the optimal rejuvenation policy from CTMDPs. Section VI presents our numerical evaluation results. Finally, Section VII gives our conclusion.

II. Related work

Huang *et al.* [2] first formulated software aging and rejuvenation behavior in a CTMC with two operational states. They focused on comparing the downtime costs due to system failure and rejuvenation, and a rejuvenation threshold for software rejuvenation was computed. This analytical approach gives theoretical conditions for distinguishing whether to take software rejuvenation or not in the two-stage software aging model. However, such a theoretical condition for software rejuvenation in multi-stage aging cases has not been discussed.

After the seminal work, many researchers exploited various types of Markovian models to analyze the software rejuvenation systems. Dohi *et al.* [8] proposed a more generalized model using a semi-Markov process (SMP), which allows general distributions for the state transition times. Dohi *et al.* [9] then developed a non-parametric algorithm to determine the optimal rejuvenation policy. Garg *et al.* [10] introduced a time-based rejuvenation method using a Markov Regenerative Stochastic Petri Net (MRSPN) model. This model captures the effect of time-based software rejuvenation on the availability of the system. They also discussed the optimal rejuvenation interval, which minimizes the system unavailability. Machida *et al.* [11] applied SMP to model system behavior and analyzed the system availability and job completion time. They proposed a virtual machine (VM)-based software life-extension method that needs to be triggered before software rejuvenation, allowing the system to shorten the job completion time and maximize the system availability. Watanabe *et al.* [12] developed CTMCs to analyze the behaviors of drone systems subject to software aging. They focus on real-time image processing on a drone, which requires a long time operation. A method that combines task offloading and software rejuvenating is presented to improve the system availability and throughput.

Measurement-based studies focus on measuring system metrics related to the software aging problem in long-running systems. Vaidyanathan *et al.* [13] proposed a measurement-based method to estimate the trend and exhaustion rates of a system. They focused on collecting workload and resource usage data, which are considered to be the main source of system degradation. Pietrantuono *et al.* [14] investigated software aging in modern object detection algorithms through long-running experiments. By tracking software aging metrics, the experimental results reveal that the object detection application exhibits software aging phenomena regardless of the different algorithms and datasets used. Watanabe *et al.* [15] studied the software aging phenomenon in real-time object detection systems for resource-limited applications, such as IoT devices. Free memory and memory swap usage were examined to determine software aging symptoms. The experiment results also show the expected failure time due to software aging. In this study, we assume that software aging progresses in multi-stages.

CTMDP is a variant of MDP that incorporates the influence of the transition times between the states. CTMDP has been used for many computing and network applications. Zhang *et al.* [16] proposed a CTMDP-based model against multi-stage cyber attacks. Multi-stage here indicates the different phases of a cyber attack. There is a high degree of variability between the stages, which is different from the problem we are studying. Buchholz *et al.* [17] developed an algorithm to compute optimal policy for finite-horizon CTMDP model. They experimentally simulated a queueing system and algorithmically calculated the optimal policy for switching the system on or off. Hou *et al.* [18] proposed a CTMDP-based resource offloading scheme for vehicle networking. A value iteration method is used to find the optimal offloading policy. In contrast to these existing studies, we attempt to apply CTMDP to model multi-stage software aging and rejuvenation.

III. continuous-Time Markov Decision Process

A. CTMDP Definition

CTMDP is an extension of MDP. MDPs are a commonly used method to formulate decision-making problems. Consider a Markov chain $\{X_n; n \in Z^+\}$, which holds the Markov property $P\{X_{n+1} = x_{n+1} | X_0 = x_0, \ldots, X_n = x_n\} = P\{X_{n+1} = x_{n+1} | X_n = x_n\}$. A Markov chain becomes an MDP when the transition probabilities can be affected by an action [19]. The MDP describes a scenario in which an agent must choose an action from an action set in every state of the model. The objective is to find a policy which can maximize the reward. The definition of reward may depend on the specific study or application.

In a CTMDP, the problem changes into continuous space. The transition time between the current state and the next state is assumed to be exponentially distributed. Given a policy ρ, which is a mapping from states to actions, the Markov property in CTMDP is defined as $P^{\rho}\left(X\left(t+u\right)=s|\{X\left(w\right);0\leq w\leq t\}\right)=P^{\rho}(X\left(t+u\right)=s|X(t))$. $A(s)$ represents the action space, which contains a set of actions that can be selected in state s. Denote $\tau_{s,a}$ as the time to the next state when action a is chosen in state s. $\tau_{s,a}$ follows exponential distribution $H\left(x|s,a\right)$ with parameter $\mu(s,a)$. It can be defined as

$$H\left(x|s,a\right)=\ 1-e^{-\mu(s,a)x},\ x\geq 0. \tag{1}$$

$\mu\left(s,a\right)$ is a constant equal to the sum of the rates to transit from state s.

$$\mu\left(s,a\right)=\sum_{s'\in S}q(s,a,s'), \tag{2}$$

where $q(s,a,s')$ represents the transition rate from state s to state s' with action a.

The optimal policy of a CTMDP can be derived from the analysis of the value function. The value function $V_{\alpha}^{\rho}\left(s\right)$ represents the expected cumulative discounted reward obtained through the process starting from state s under policy ρ, which can be formulated with the value of the previous states $V_{\alpha}^{\rho}\left(s'\right)$ [20]

$$V_{\alpha}^{\rho}\left(s\right)= \\ r\left(s,\rho(s)\right)+\int_{0}^{\infty}e^{-\alpha t}dH\left(t|s,a\right)\sum_{s'\in S}p(s'|s,\rho(s))V_{\alpha}^{\rho}\left(s'\right), \tag{3}$$

where $r\left(s,\rho(s)\right)$ is the instantaneous reward for action $\rho(s)$, $\alpha\in[0,1]$ is a discount factor to achieve convergence of the value function over an infinite horizon, and $p(s'|s,\rho(s))$ represents the transition probability from state s to state s'. The optimal policy ρ_{opt} is given by maximizing the value function $V_{a}^{\rho}(s)$ over all possible policies ρ. Mathematically, it can be expressed as

$$\rho_{opt}(s)=\arg\max_{\rho}V_{\alpha}^{\rho}\left(s\right). \tag{4}$$

B. Two-stage model formulation

The first software rejuvenation model presented by Huang *et al.* [2] can be formulated as a two-stage aging model of CTMDP as shown in Figure 1. State 0 represents the robust state, while state 1 is a failure probable state. States F and R denote the failure and rejuvenation states, respectively. We assume that the aging rate is β. The failure rate is λ. The software rejuvenation transitions are activated only when rejuvenation action is determined. The rejuvenation trigger rate is denoted as δ. The failure-recovery rate and the rejuvenation rate are denoted as μ_f and μ_r, respectively. A decision can only be made in state 1 which is denoted as a square in Figure 1. The system can issue either one of the following two actions

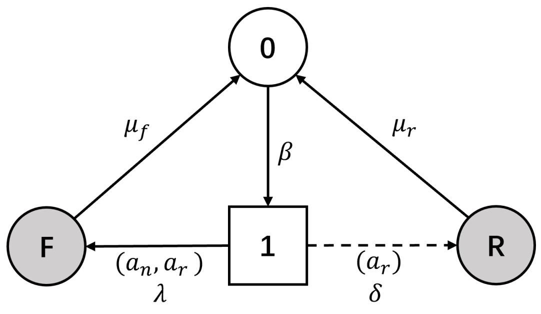


Fig. 1: A two-stage aging model by CTMDP

TABLE I: Reward for the uptime and the total runtime in the two-stage aging model.

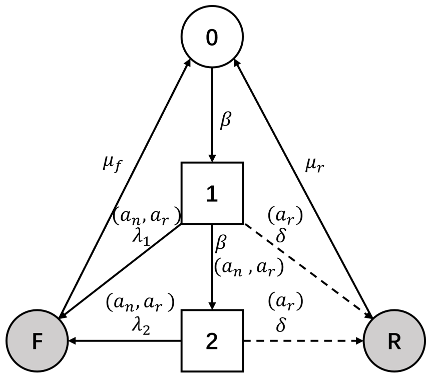
s	a	$r_u(s,a)$	$r_t(s,a)$
0	a_n	$\frac{1}{\beta}$	$\frac{1}{\beta}$
1	a_n	$\frac{1}{\lambda}$	$\frac{1}{\lambda}$
1	a_r	$\frac{1}{\delta+\lambda}$	$\frac{1}{\delta+\lambda}$
F	a_n	0	$\frac{1}{\mu_f}$
R	a_n	0	$\frac{1}{\mu_r}$

- a_n is No action.
- a_r is Rejuvenation.

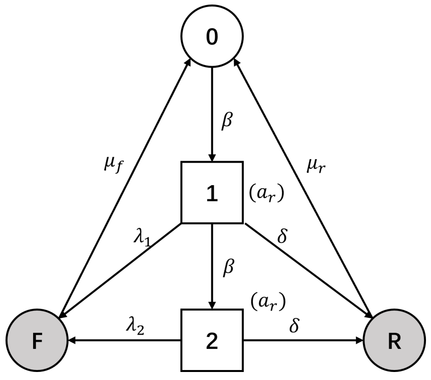
The policy is defined as the mapping from state 1 to either a_n or a_r. In this model, there are two feasible policies, which are $\rho_1=(a_r)$ and $\rho_2=(a_n)$. When policy ρ_1 is chosen, only state transition to state F is allowed at state 1. On the other hand, if policy ρ_2 is chosen, both state transitions to state F and state R are possible from state 1. As a result, policies ρ_1, ρ_2 can result in two different CTMCs corresponding to the two probabilistic state transition models with and without software rejuvenation shown in Huang *et al.* [2]. Through the analysis of two CTMCs, the seminal work derives a rejuvenation threshold. They defined the average cost per unit c_r for taking rejuvenation and c_f for failure. By taking the derivative with respect to δ of the cost expression for software rejuvenation, they derive condition g which establishes the relationship between c_r and c_f.

$$g=\frac{\lambda(\beta+\mu_r)}{\lambda(\mu_f+\beta)+\beta\mu_f}. \tag{5}$$

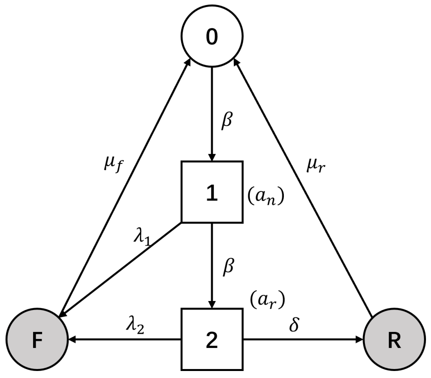
Condition g is defined as the rejuvenation threshold. When $c_f > gc_r$, it is better to perform software rejuvenation immediately in state 1. On the other hand, when $c_f < gc_r$, there is no benefit to performing rejuvenation because the cost of software rejuvenation increases and the overall runtime also increases. However, for a system with more than two-stage software aging, the condition of effective rejuvenation could be different.



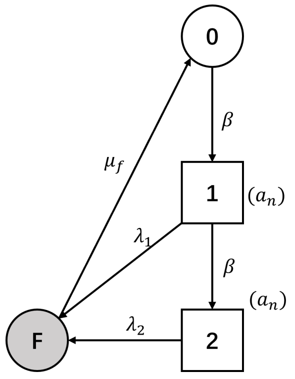
(a) A three-stage aging model by CTMDP



(b) CTMC for policy ρ_1



(c) CTMC for policy ρ_2



(d) CTMC for policy ρ_3

Fig. 2: A CTMDP for three-stage aging model and the CTMCs

IV. Three-stage aging model

A. Definition

With the definitions in the previous section, we can naturally extend the model into three stages. In this three-stage model, we add a new state that represents another stage of software aging, denoted as state 2 in Figure 2(a). The system can decide a rejuvenation action in state 1 and state 2, thus the states are represented as squares. We assume that the failure rate in a later stage is higher than that in the current stage. The failure rate from state 1 to F is λ_1, while we set the failure rate from state 2 to F to $\lambda_2 > \lambda_1$. In this model, a decision policy is defined as a mapping from state 1 and state 2 to either one of a_n or a_r. There are four possible mappings in theory. We assume if rejuvenation action is determined in state 1, a_n cannot be selected in state 2. Therefore, we assume that a_r is consistently chosen once a_r is selected in an earlier stage of aging. As a result, there are three possible rejuvenation policies in this model. Each policy is represented as a pair of actions in state 1 and state 2, such as $\rho_1 = (a_r, a_r)$, $\rho_2 = (a_n, a_r)$ and $\rho_3 = (a_n, a_n)$. Figures 2(b), (c), and (d) show the CTMCs corresponding to the chosen policy ρ_1, ρ_2, and ρ_3, respectively.

By the analysis of the corresponding CTMCs, the achievable steady-state availability can be computed by the sum of the steady-state probabilities that the system is in either one of the operating states (i.e., states 0, 1, 2). Let $\boldsymbol{\pi}$ =$(\pi_0, \pi_1, \pi_2, \pi_f, \pi_r)$ be the steady-state probability vector of the CTMC. The steady-state probability $\boldsymbol{\pi}$ is given by the solution of the system of equations $\boldsymbol{\pi Q} = 0$, and $\boldsymbol{\pi e}^T = 1$ where $\boldsymbol{Q}$ is the infinitesimal generator matrix of the CTMC and $\boldsymbol{e}^T$ is n-dimensional vector which all the element is 1.

The infinitesimal generator matrices $\boldsymbol{Q}_j$, $j \in \{1, 2, 3\}$ are given by

$$\boldsymbol{Q}_1 = \begin{bmatrix} -\beta & \beta & 0 & 0 & 0 \\ 0 & -(\beta+\delta+\lambda_1) & \beta & \lambda_1 & \delta \\ 0 & 0 & -(\delta+\lambda_2) & \lambda_2 & \delta \\ \mu_f & 0 & 0 & -\mu_f & 0 \\ \mu_r & 0 & 0 & 0 & -\mu_r \end{bmatrix}, \tag{6}$$

$$\boldsymbol{Q}_2 = \begin{bmatrix} -\beta & \beta & 0 & 0 & 0 \\ 0 & -(\beta+\lambda_1) & \beta & \lambda_1 & 0 \\ 0 & 0 & -(\delta+\lambda_2) & \lambda_2 & \delta \\ \mu_f & 0 & 0 & -\mu_f & 0 \\ \mu_r & 0 & 0 & 0 & -\mu_r \end{bmatrix}, \tag{7}$$

$$\boldsymbol{Q}_3 = \begin{bmatrix} -\beta & \beta & 0 & 0 \\ 0 & -(\beta+\lambda_1) & \beta & \lambda_1 \\ 0 & 0 & -(\delta+\lambda_2) & \lambda_2 \\ \mu_f & 0 & 0 & -\mu_f \end{bmatrix}. \tag{8}$$

The steady-state probabilities $\boldsymbol{\pi}$ for the corresponding CTMCs can be derived by the linear system of equations. Then, the steady-state availabilities A_i for three CTMCs are derived as follows.

$$
\begin{aligned}
A_1 &= \frac{X_1}{X_1+Y_1},\\
X_1 &= 1+\frac{\beta(\beta+\delta+\lambda_2)}{(\beta+\delta+\lambda_1)(\delta+\lambda_2)},\\
Y_1 &= \frac{\beta((\delta+\lambda_2)(\delta\mu_f+\lambda_1\mu_r)+\beta(\delta\mu_f+\lambda_2\mu_r))}{(\beta+\delta+\lambda_1)\,(\delta+\lambda_2)\,\mu_f\mu_r}.
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
A_2 &= \frac{X_2}{X_2+Y_2},\\
X_2 &= 1+\frac{\beta(\beta+\delta+\lambda_2)}{(\beta+\lambda_1)(\delta+\lambda_2)},\\
Y_2 &= \frac{\beta(\lambda_1\,(\delta+\lambda_2)\,\mu_r+\beta(\delta\mu_f+\lambda_2\mu_r))}{(\beta+\lambda_1)\,(\delta+\lambda_2)\,\mu_f\mu_r}.
\end{aligned} \tag{10}
$$

$$
A_3 = \frac{(\beta^2+2\beta\lambda_2+\lambda_1\lambda_2)\mu_f}{\lambda_1\lambda_2\mu_f+\beta^2(\lambda_2+\mu_f)+\beta\lambda_2(\lambda_1+2\mu_f)}. \tag{11}
$$

B. Analysis

We can compare the steady-state availabilities achieved by different rejuvenation policies from expressions (9)(10)(11). The optimal policy for the CTMDP can be analytically given by the following proposition.

Proposition 1. Given parameters $\beta, \delta, \lambda_1, \lambda_2, \mu_f, \mu_r \in \mathcal{R}^+$, where $\lambda_1 < \lambda_2$ and $\mu_f < \mu_r$. The optimal policy ρ_{opt} is uniquely determined by the difference of rejuvenation and failure-recovery rates $\Delta\mu = \mu_r - \mu_f$ with the boundary conditions g_1 and g_2.

$$
\rho_{opt} = \begin{cases} \rho_1 & \Delta\mu \geq g_1, \\ \rho_2 & g_1 > \Delta\mu \geq g_2, \\ \rho_3 & \Delta\mu < g_2. \end{cases} \tag{12}
$$

$$
g_1 = \frac{(\beta^2+\beta\delta+\beta\lambda_2)\mu_f}{\delta\lambda_1+\beta\lambda_2+\lambda_1\lambda_2}, \tag{13}
$$

$$
g_2 = \frac{(\beta^2+\beta\lambda_1)\mu_f}{-\beta\lambda_1+2\beta\lambda_2+\lambda_1\lambda_2}. \tag{14}
$$

Proof. A system with ρ_1 achieves a higher availability than a system with ρ_2 when $A_1 - A_2 \geq 0$. Therefore, taking the difference between A_1 and A_2, we have

$$
\begin{aligned}
&A_1 - A_2 = \\
&-(\beta\delta\,(\delta+\lambda_2)\,\mu_f\mu_r(\beta^2\mu_f+\lambda_1(\delta+\lambda_2) \\
&(\mu_f-\mu_r)+\beta(\delta\mu_f+2\lambda_2\mu_f-\lambda_2\mu_r))/(\lambda_1(\delta+\lambda_2)\mu_f\mu_r \\
&+\beta(\delta+\lambda_2)(\lambda_1+2\mu_f)\mu_r+\beta^2(\delta\mu_f+(\lambda_2+\mu_f)\mu_r)) \\
&((\delta+\lambda_1)(\delta+\lambda_2)\mu_f\mu_r+\beta^2(\delta\mu_f+(\lambda_2+\mu_f)\mu_r) \\
&+\beta(\delta+\lambda_2)(\delta\mu_f+(\lambda_1+2\mu_f)\mu_r))).
\end{aligned} \tag{15}
$$

$A_1 - A_2 \geq 0$ holds when the numerator is positive, which means

$$
\begin{aligned}
&(\beta\delta\,(\delta+\lambda_2)\,\mu_f\mu_r(\beta^2\mu_f+\lambda_1(\delta+\lambda_2)(\mu_f-\mu_r) \\
&+\beta(\delta\mu_f+2\lambda_2\mu_f-\lambda_2\mu_r))\geq 0.
\end{aligned} \tag{16}
$$

From the equation above, we can drive the condition of $\Delta\mu$ for satisfying $A_1 - A_2 \geq 0$ as follows

$$
\Delta\mu \geq \frac{(\beta^2+\beta\delta+\beta\lambda_2)\mu_f}{\delta\lambda_1+\beta\lambda_2+\lambda_1\lambda_2} = g_1. \tag{17}
$$

Next, a system with ρ_2 achieves a higher availability than a system with ρ_3 when $A_2 - A_3 \geq 0$. Therefore, taking difference between A_2 and A_3, we have

$$
\begin{aligned}
&A_2 - A_3 = \\
&-(\beta\delta\,(\delta+\lambda_2)\,\mu_f\mu_r \\
&(\beta^2\mu_f+\lambda_1(\delta+\lambda_2)(\mu_f-\mu_r)+ \\
&\beta(\delta\mu_f+2\lambda_2\mu_f-\lambda_2\mu_r))/(\lambda_1(\delta+\lambda_2)\mu_f\mu_r+ \\
&\beta(\delta+\lambda_2)(\lambda_1+2\mu_f)\mu_r+\beta^2(\delta\mu_f+(\lambda 2+\mu_f)\mu_r)) \\
&((\delta+\lambda_1)(\delta+\lambda 2)\mu_f\mu_r+\beta^2(\delta\mu_f+(\lambda_2+\mu_f)\mu_r) \\
&+\beta(\delta+\lambda_2)(\delta\mu_f+(\lambda 1+2\mu_f)\mu_r))).
\end{aligned} \tag{18}
$$

$A_2 - A_3 \geq 0$ holds when the numerator is positive, which means

$$
\Delta\mu \geq \frac{(\beta^2+\beta\lambda_1)\mu_f}{-\beta\lambda_1+2\beta\lambda_2+\lambda_1\lambda_2} = g_2. \tag{19}
$$

Finally, comparing g_1 and g_2, we have

$$
\begin{aligned}
&g_1 - g_2 = \\
&\frac{\mu_f\beta(\lambda_2-\lambda_1)(\beta^2+2\beta(\delta+\lambda_2)+\lambda_1(\delta+\lambda_2))}{(\delta\lambda_1+(\beta+\lambda_1)\,\lambda_2)((2\beta+\lambda_1)\,\lambda_2)-\beta\lambda_1)}.
\end{aligned} \tag{20}
$$

Under the condition $\lambda_1 < \lambda_2$, the numerator and denominator are both positive. Therefore, two boundary conditions satisfy $g_1 > g_2$. $\square$

The proposition states that the optimal rejuvenation policy that maximizes the steady-state availability is uniquely determined by the difference between rejuvenation and failure-recovery rates with the boundary conditions g_1 and g_2. We can also show the following corollaries about the boundary conditions without proof.

Corollary 1. The boundary conditions g_1 and g_2 are monotonically increasing with respect to β.

Corollary 2. The boundary condition g_2 is independent of δ.

From Proposition 1 and Corollary 1, the boundaries of the optimal policies can be visualized on $\beta - \Delta\mu$ coordinate for given λ_1, λ_2, δ, and μ_f. Figure 3 shows an example of boundary conditions g_1 and g_2 in the $\beta - \Delta\mu$ coordinate where other parameters are set as shown in Table III. Given the value of β, the optimal rejuvenation policy is uniquely determined by $\Delta\mu$. The upper area above the g_1 condition implies that ρ_1 is the optimal policy if a $\beta - \Delta\mu$ coordinate falls in this area. Similarly, if the coordinates fall in the middle area between g_1 and g_2, ρ_2 is the optimal policy. If the coordinates fall in the lower area under g_2, ρ_3 is the optimal policy.

V. Policy evaluation algorithm

Although the conditions for the optimal rejuvenation policy can be obtained theoretically for the three-stage aging model, such a symbolic solution is formidable when the number of stages increases. In order to analyze the optimal rejuvenation policies for multi-stage software aging and rejuvenation models in general, we develop a numerical policy evaluation algorithm. The algorithm derives the optimal policy by comparing the approximated steady-state availabilities achieved by feasible policies. We introduce two rewards for each pair of state and action in the CTMDP, which are the expected uptime $r_u(s,a)$ and the expected total runtime $r_t(s,a)$. The availability can be approximated by the ratio of $r_u(s,a)$ to $r_t(s,a)$. The rewards for the two-stage and three-stage aging models can be defined as Tables I and II, respectively. For example, the expected sojourn time in state 1 when choosing a_r in the two-stage model is $\frac{1}{\delta+\lambda}$, which is equal to the uptime reward and the total runtime reward. Since F and R are down states, the uptime reward is 0, while the total runtime reward is equal to the expected time to transit to state 0.

Algorithm 1 Policy evaluation for n-stage aging model

```
Input: $n$; $\beta, \delta, \mu_f, \mu_r, \lambda_1, \lambda_2, ..., \lambda_{n-1}$
 1: for $\rho \in \rho_1, ..., \rho_n$ do
 2:     for $s \in S$ do
 3:         $U_\rho(s) \leftarrow 0$
 4:         $T_\rho(s) \leftarrow 0$                    ▷ Initialize state value to 0
 5:     end for
 6:     repeat
 7:         for $s \in S$ do
 8:             $\Delta \leftarrow 0$
 9:             $u \leftarrow r_u(s,\rho(s)) + \sum_{s' \in S} p(s'|s,\rho(s)) U_\rho(s')$
10:             $t \leftarrow r_t(s,\rho(s)) + \sum_{s' \in S} p(s'|s,\rho(s)) T_\rho(s')$
11:             if $t \neq 0 \wedge T_\rho(s) \neq 0$ then
12:                 $\Delta \leftarrow \max(\Delta, \left|\frac{u}{t} - \frac{U_\rho(s)}{T_\rho(s)}\right|)$
13:             end if
14:             $U_\rho(s) \leftarrow u$
15:             $T_\rho(s) \leftarrow t$    ▷ Pass the value for the next loop
16:         end for
17:     until $\Delta < \sigma$
18: end for
19: $\rho_{opt} \leftarrow \arg\max \frac{U_\rho(0)}{T_\rho(0)}$
Output: $\rho_{opt}$
```

The policy evaluation algorithm is shown in Algorithm 1. For an n-stage software aging and rejuvenation model, there are n corresponding policies. The number of stages n and the feasible policies $\rho_1, ..., \rho_n$ are given as inputs. For each policy ρ, two arrays $U_\rho(s)$ and $T_\rho(s)$ are initialized with zero values. These arrays are used to accumulate the total uptime and runtime, respectively, during repeat iteration. In the repeat iteration starting from line 6, two temporary variables, u and t, are computed as the values of the given state s by expression (3) in lines 9 and 10. Note that we set the discount factor $\alpha = 0$ in our algorithm to make a stable numerical computation. The

TABLE II: Reward for the uptime and the total runtime in the three-stage aging model.

s	a	$r_u(s,a)$	$r_t(s,a)$
0	a_n	$\frac{1}{\beta}$	$\frac{1}{\beta}$
1	a_n	$\frac{1}{\lambda_1+\beta}$	$\frac{1}{\lambda_1+\beta}$
1	a_r	$\frac{1}{\lambda_1+\beta+\delta}$	$\frac{1}{\lambda_1+\beta+\delta}$
2	a_n	$\frac{1}{\lambda_2}$	$\frac{1}{\lambda_2}$
2	a_r	$\frac{1}{\lambda_2+\delta}$	$\frac{1}{\lambda_2+\delta}$
F	a_n	0	$\frac{1}{\mu_f}$
R	a_n	0	$\frac{1}{\mu_r}$

TABLE III: Parameters values for numerical experiments

Variables	Values [1/hour]
δ	$1/2$
λ_1	$1/12$
λ_2	$1/10$
μ_f	1

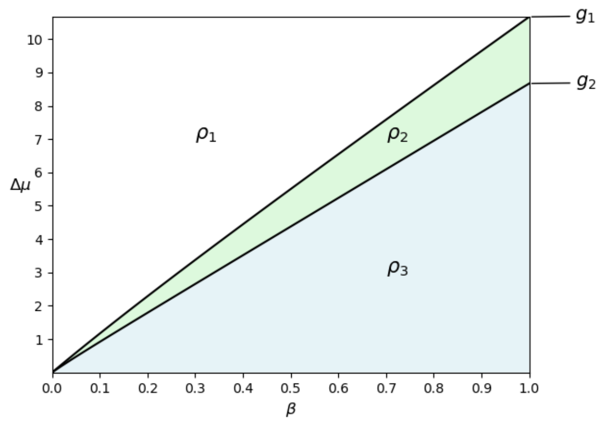


Fig. 3: Boundary conditions of the optimal policy

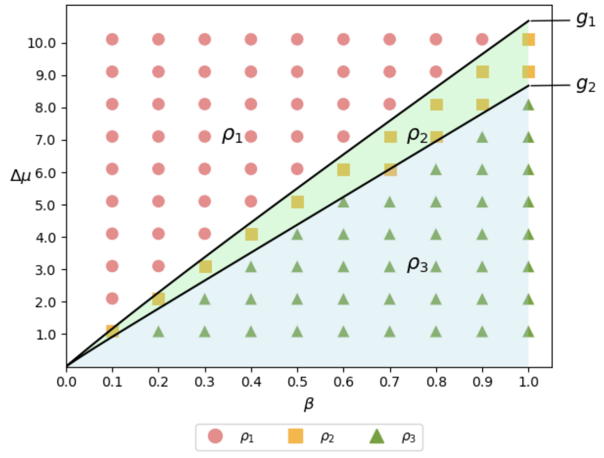


Fig. 4: Computed policy for three-stage aging model

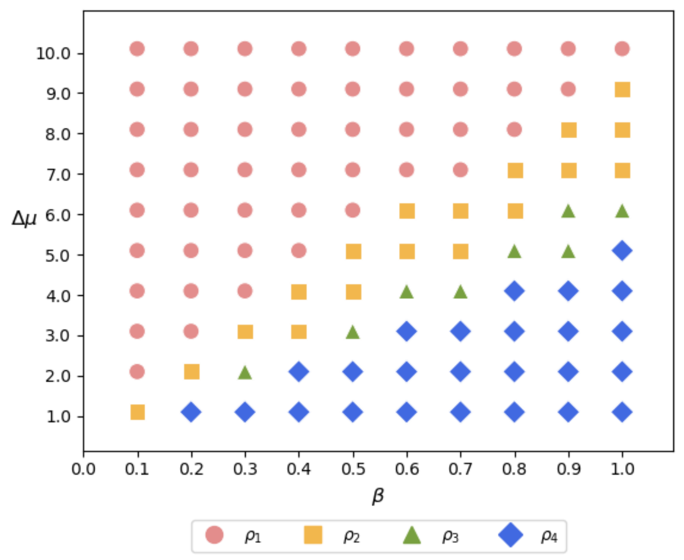


Fig. 5: Computed policy for four-stage aging model

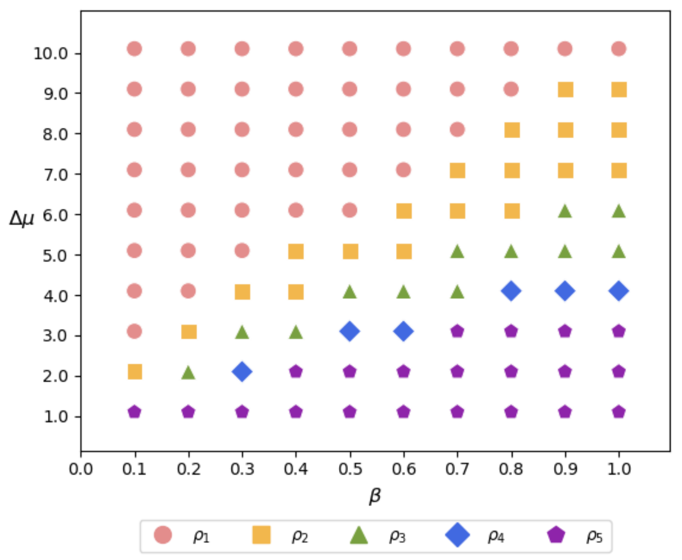


Fig. 6: Computed policy for five-stage aging model

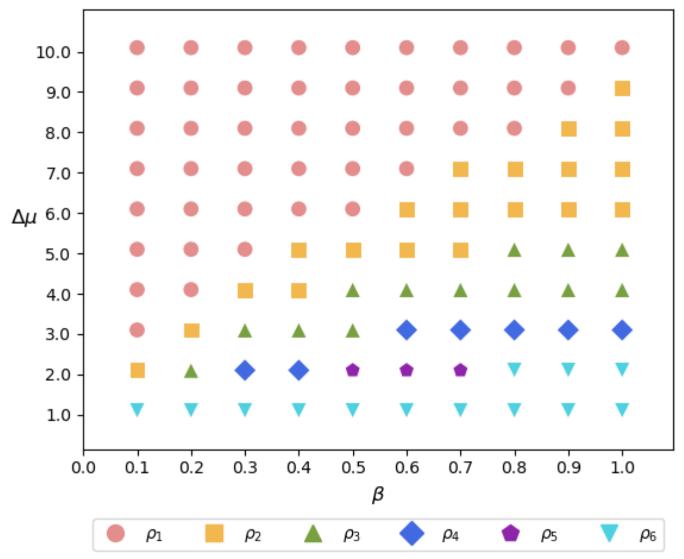


Fig. 7: Computed policy for six-stage aging model

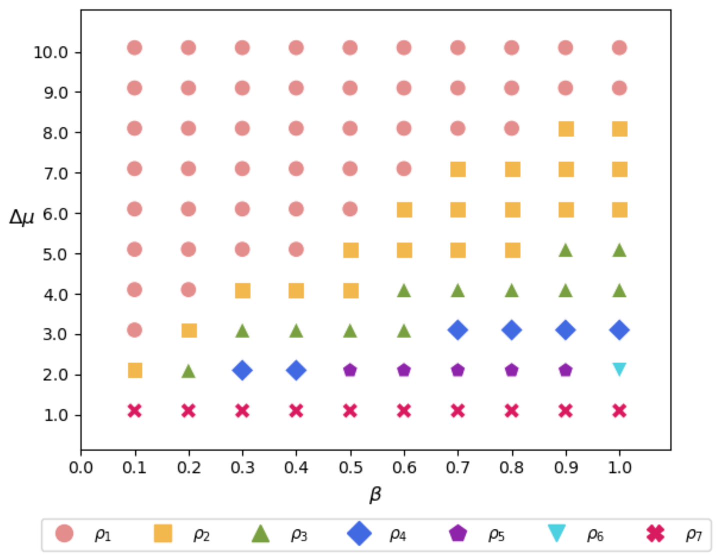


Fig. 8: Computed policy for seven-stage aging model

proportion of u to t is then compared to the proportion of $U_\rho(s)$ to $T_\rho(s)$, and the difference is recorded as Δ at line 12. Then, $U_\rho(s)$ and $T_\rho(s)$ are updated with the current values u and t, respectively, at lines 14 and 15. After computing the values of all the states, in line 17, the algorithm judges whether the value of Δ is less than σ, which is a small number to judge the convergence of values. The same calculation is performed for all policies. Finally, after obtaining the values of all the states, we calculate the ratio of $U_\rho(s)$ and $T_\rho(s)$ in state 0, which represents the approximated system availability. This value is used as a criterion to compare the effectiveness of different policies. At line 19, the policy ρ that maximizes the value of $U_\rho(0)/T_\rho(0)$ is chosen as the best policy.

VI. Numerical study

A. *Three-stage model*

In this section, we apply the numerical policy evaluation algorithm for the CTMDP and validate the results with the theoretical conditions shown in Section IV. We use the parameters in Table III and vary the values of β within $\{0.1, ..., 1\}$ and $\Delta\mu$ within $\{1.0, ..., 10.0\}$. Note that these parameter values satisfy the precondition of Proposition 1. The computed optimal policies at different combinations of β and $\Delta\mu$ are plotted in Figure 4, where each optimal algorithm is distinguished by shape and color. For the comparison, the theoretical boundary conditions (g_1 and g_2) are also depicted on the plot.

As can be seen, the optimum policies evaluated by the algorithm are clearly distinguished by the theoretical boundary conditions, indicating that for the three-stage aging model, the optimal policy derived by the algorithm correctly matches the theoretical results.

B. More than four-stage models

By applying the policy evaluation algorithm, we can analyze the boundary conditions for more than four-stage aging models. We show the results obtained for $n = \{4, 5, 6, 7\}$ in Figure 5, 6, 7, and 8, respectively. The transition rate between two consecutive states is set to β. The failure rate $\lambda_1, \lambda_2, \ldots$ increase monotonically which represents $\lambda_1 < \lambda_2 < \ldots < \lambda_{n-1}$. The condition $\mu_f < \mu_r$ remains unchanged. The values of μ_f and δ are assumed unchanged.

We can see that given a value of β for a multi-stage aging model, when $\Delta\mu$ increases, software rejuvenation should be triggered as early as possible. This is reflected in the fact that changes in strategy are monotonous. The state s of the trigger of software rejuvenation is advanced one step at a time as $\Delta\mu$ increases. Similarly, given a value of $\Delta\mu$, the software rejuvenation should be delayed as much as possible as the value of β increases. It can be deduced that the same law is satisfied for models with more than seven stages.

As the model has more aging stages, candidate policies also increase. For example, when $n = 7$, the optimal policy can be either one of seven candidate policies. However, the policy that triggers software rejuvenation only in later stages, such as ρ_6 in Figure 8, has a lower chance of the optimum policy. This observation implies that an earlier decision of software rejuvenation potentially achieves a better availability than the rejuvenation decision in the later stages.

Considering practical usage, our numerical plots can give a hint to determine the appropriate timing to trigger software rejuvenation. Although engineers cannot access the exact failure rate values, such as $\lambda_1, \lambda_2, \ldots$, they may collect the information for $\Delta\mu$ by experiments and software aging stages via monitoring resource usages which may provide a guess of the region on the $\beta - \Delta\mu$ coordinate. The engineers can at least avoid a wrong choice of rejuvenation policy from the pattern of optimum rejuvenation policies exhibited in our plots.

VII. Conclusion

We present an approach to formulate multi-stage software aging and rejuvenation systems based on CTMDP and analyze the optimal policies maximizing the system availability. For the three-stage aging model, we derive theoretical conditions that can determine the optimal rejuvenation policy. For the software rejuvenation decision problem formulated by CTMDP, we developed an approximate policy evaluation algorithm to derive the optimal rejuvenation policy by introducing the rewards for evaluating availability. This algorithm enables the comparison of the policies without generating CTMCs. Through the numerical analysis, we show that the obtained optimal policy matches with the theoretical results in the three-stage model. We also showed the boundaries of the optimal rejuvenation policies for more than four-stage models.

References

[1] M. Grottke, R. Matias, and K. S. Trivedi, "The fundamentals of software aging," in *2008 IEEE International conference on software reliability engineering workshops (ISSRE Wksp)*. Ieee, 2008, pp. 1–6.

[2] Y. Huang, C. Kintala, N. Kolettis, and N. D. Fulton, "Software rejuvenation: Analysis, module and applications," in *Twenty-fifth international symposium on fault-tolerant computing. Digest of papers*. IEEE, 1995, pp. 381–390.

[3] Y. Bao, X. Sun, and K. S. Trivedi, "A workload-based analysis of software aging, and rejuvenation," *IEEE Transactions on Reliability*, vol. 54, no. 3, pp. 541–548, 2005.

[4] A. Pfening, S. Garg, A. Puliafito, M. Telek, and K. S. Trivedi, "Optimal software rejuvenation for tolerating soft failures," *Performance Evaluation*, vol. 27, pp. 491–506, 1996.

[5] H. Okamura and T. Dohi, "Dynamic software rejuvenation policies in a transaction-based system under markovian arrival processes," *Performance Evaluation*, vol. 70, no. 3, pp. 197–211, 2013.

[6] H. Eto and T. Dohi, "Analysis of a service degradation model with preventive rejuvenation," in *Service Availability: Third International Service Availability Symposium, ISAS 2006, Helsinki, Finland, May 15-16, 2006. Revised Selected Papers 3*. Springer, 2006, pp. 17–29.

[7] A. Avritzer, A. Janes, A. Marin, A. van Hoorn, M. Camilli, C. Trubiani, and D. S. Menasché, "Assessment of aging and rejuvenation for resiliency in heterogeneous network clusters," in *2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2023, pp. 198–205.

[8] T. Dohi, K. Goseva-Popstojanova, and K. S. Trivedi, "Statistical non-parametric algorithms to estimate the optimal software rejuvenation schedule," in *Proceedings. 2000 Pacific Rim International Symposium on Dependable Computing*. IEEE, 2000, pp. 77–84.

[9] T. Dohi, K. Goševa-Popstojanova, and K. Trivedi, "Estimating software rejuvenation schedules in high-assurance systems," *The Computer Journal*, vol. 44, no. 6, pp. 473–485, 2001.

[10] S. Garg, A. Puliafito, M. Telek, and K. S. Trivedi, "Analysis of software rejuvenation using markov regenerative stochastic petri net," in *Proceedings of Sixth International Symposium on Software Reliability Engineering. ISSRE'95*. IEEE, 1995, pp. 180–187.

[11] F. Machida, J. Xiang, K. Tadano, and Y. Maeno, "Lifetime extension of software execution subject to aging," *IEEE Transactions on Reliability*, vol. 66, no. 1, pp. 123–134, 2016.

[12] K. Watanabe and F. Machida, "Availability analysis of a drone system with proactive offloading for software life-extension," in *2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*. IEEE, 2022, pp. 1–6.

[13] K. Vaidyanathan and K. S. Trivedi, "A measurement-based model for estimation of resource exhaustion in operational software systems," in *Proceedings 10th International Symposium on Software Reliability Engineering (Cat. No. PR00443)*. IEEE, 1999, pp. 84–93.

[14] R. Pietrantuono, D. Cotroneo, E. Andrade, and F. Machida, "An empirical study on software aging of long-running object detection algorithms," in *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2022, pp. 1091–1102.

[15] K. Watanabe, F. Machida, E. Andrade, R. Pietrantuono, and D. Cotroneo, "Software aging in a real-time object detection system on an edge server," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 2023, pp. 671–678.

[16] Y. Zhang, X. Chang, J. Mišić, V. B. Mišić, and Y. Cai, "Cost-effective migration-based dynamic platform defense technique: a ctmdp approach," *Peer-to-Peer Networking and Applications*, vol. 14, pp. 1207–1217, 2021.

[17] P. Buchholz, I. Dohndorf, and D. Scheftelowitsch, "Optimal decisions for continuous time markov decision processes over finite planning horizons," *Computers & Operations Research*, vol. 77, pp. 267–278, 2017.

[18] L. Hou, K. Zheng, P. Chatzimisios, and Y. Feng, "A continuous-time markov decision process-based resource allocation scheme in vehicular cloud for mobile video services," *Computer Communications*, vol. 118, pp. 140–147, 2018.

[19] C. C. White III and D. J. White, "Markov decision processes," *European Journal of Operational Research*, vol. 39, no. 1, pp. 1–16, 1989.

[20] X. Guo, O. Hernández-Lerma, X. Guo, and O. Hernández-Lerma, *Continuous-time Markov decision processes*. Springer, 2009.