Optimal Rejuvenation for Tolerating Soft Failures

András Pfening, Sachin Garg, Antonio Puliafito,
Miklós Telek, Kishor S. Trivedi

Abstract

In the paper we address the problem of determining the optimal time to rejuvenate a server software when it experiences soft failures (witnessed in telecommunication systems) because of aging. In such a system, it is assumed that crash failures do not occur. However, the service rate of the software gradually decreases with time and settles to a very low value. Since the performability in this state is very low, it is necessary to "renew" the software to its peak performance level. We optimize the performance of this system for two different queueing policies under Poisson arrivals and decreasing service rate.
Keywords: Fault Tolerant Systems, Optimal Software Rejuvenation, Markov Decision Process, Optimal Stopping Problem.

1 Introduction

Preventive maintenance is considered to be one of the key strategies to increase system availability and performance. In general, preventive maintenance consists of periodically stopping the system, and restarting it after doing proper maintenance, that reduces the probability of failure and increases system performance. Some cost is unavoidable since the system has to be stopped and it is unavailable during the maintenance. The arising research problem is to find the optimal maintenance policy, the policy that minimizes a certain cost function. While preventive maintenance concepts have been usually applied to mechanical systems, they can also be effectively applied to the field of software reliability.

System failures due to imperfect software behaviour are usually more frequent than failures caused by hardware components' faults [?]. These failures result from either inherent design defects in the software or from improper usage by clients [?]. Thus fault tolerant software has become an effective alternative to virtually impossible fault-free software.

Huang et. al. have suggested a complimentary technique which is preventive in nature. It involves periodic maintenance of the software so as to prevent crash failures, they call it *Software Rejuvenation* [?]. *Garg et. al.* [?] have improved Huang's model by allowing deterministic regeneration time and provided an optimal rejuvenation policy for the studied class of systems, regarding crash failures.

But monitoring real applications showed that software "ages" when it is run, i.e. its performance slowly decreases. Memory bloating, unreleased file-locks, data corruption are the typical causes of slow degradation (*soft failure*) which could lead to a failure state as well if it is not taken care of. Software rejuvenation involves periodically stopping the system, cleaning up, and restarting it from its peak performance level. The proposed "renewal" of software not only prevents failure states but it optimizes performance by considering the slow degradation of server software.

Two systems are analyzed in the paper, they differ in the applied queuing policy. The first studied system does not allow buffer overflow (we will refer to it as *no buffer overflow case*) by stopping and rejuvenating the system when the buffer is full and a new job arrives to the system. It may be the case when the buffer is supposed to be large enough to accommodate all the arriving jobs, or when the system operator does not want to loose jobs during the system operation. The second scenario (*buffer overflow case*) allows buffer overflow during operation, however the cost caused by the lost jobs must be reflected in the overall cost function.

The rest of the paper is organized as follows. Section 2 introduces the problem statement. Section 3 and 4 discusses the two considered system models with and without buffer overflow and their analysis. A numerical example is detailed in Section 5 and the paper is concluded in Section 6.

2 Problem Statement

We address the problem of soft failures of a server software, i.e. the preventive maintenance is done to increase system performance, while crash failures are not considered. The server software serves jobs arriving to the system with slowly degrading performance. The problem is to determine the rejuvenation time interval, if the probability distribution of the interarrival times and the service rate are known. It should be performed to optimize the cost of the rejuvenation, consisting of the costs paid for the lost jobs that arrived during the rejuvenation and costs paid for the jobs that were queued waiting for service when rejuvenation started, since these jobs are lost. We also take into account the run time of the system, since the same cost paid in case of a longer run is preferred.

We assume that the customers arrive to the system according to a Poisson process, and the service time is negative exponentially distributed. The degradation of the system is reflected in the decreasing service rate. Since at the time when we decide to rejuvenate the system the number of lost jobs due to unavailable service is not known, this value is approximated by the product of the arrival rate (λ) and the rejuvenation time (T_R), λT_R.

Assume the following notation for the rest of the paper:

T	variable denoting the time from start to initiating rejuvenation,
T_R	time it takes to rejuvenate the system (constant),
X	random variable denoting the number of clients in the queue at time T, i.e. when rejuvenation is initiated,
Y	random variable denoting number of clients denied service when rejuvenation is in progress, i.e. in $(T, T+T_R)$,
λ	job arrival rate,
$\mu(t)$	time dependent service rate, where $\lim_{t\to\infty} \mu(t) = \mu_\infty$,
B	buffer length.

The optimization problem can be stated as:

Find T that minimizes the average cost of the run

$$\min_T \Big\{ E\left[C(X,T,Y)\right] \Big\},$$

if λ, $\mu(t)$, T_R, B are given and $C(.)$ denotes the cost function.

3 Optimal Rejuvenation without Buffer Overflow

In the first studied case if the system arrives to a state when the buffer is full, and a new job arrives, we immediately stop and rejuvenate the system, thus buffer overflow is avoided. This is the case when the system operator does not want to lose customers (jobs) during normal operation, in other words the fact that the buffer is full indicates that it is time to rejuvenate the system. The states of the system in the operational period can be described by two variables, namely the number of customers in the system, and the time spent since the last rejuvenation. In each state of the system we have to decide whether to continue service or to stop and rejuvenate the system.

3.1 Markov Decision Process Solution

In this approach time is discretized in Δ steps, and since the customers arrive to the system according to a Poisson process, and the service time in a state follows negative exponential distribution, we have a Markov decision process (MDP), more specifically an optimal stopping problem. Our goal is to find the *optimal stationary policy* f, which determines the action in each state, dependent only on the current state, i.e. to rejuvenate the system or to continue service. The policy is optimal in a sense that it minimizes the expected cost of the process.

The nature of cost functions can be summarized as follows:

$$\begin{array}{lcl} C(i,j,stop) & \geq & 0, \quad 0 \leq i \leq B, 0 \leq j, \\ C(i,j,continue) & = & 0, \quad 0 \leq i < B, 0 \leq j, \end{array}$$

where i denotes the number of customers in the system, and j is the integer number of Δ time units denoting the time spent since the last rejuvenation. At the moment of the decision Y is not explicitly known, therefore we can use the expectation of it, $E[Y] = \lambda T_R$, since Poisson arrivals are assumed. We require all the costs to be nonnegative.

We define the probability $P_{i,j,k,l}(a)$ as the probability of going from state (i,j) to state (k,l) when action a is chosen. In our case the transition probabilities are defined as follows:

$$\begin{array}{lllll} (i) & P_{\cdot,\cdot,stop,stop}(stop) & = & 1, & \\ (ii) & P_{0,j,1,j+1}(continue) & = & \lambda\Delta + o(\Delta) & j \geq 0, \\ (iii) & P_{0,j,0,j+1}(continue) & = & 1 - \lambda\Delta + o(\Delta) & j \geq 0, \\ (iv) & P_{i,j,i+1,j+1}(continue) & = & \lambda\Delta + o(\Delta) & 1 \leq i < B,\ j \geq 0, \\ (v) & P_{i,j,i-1,j+1}(continue) & = & \mu(j)\Delta + o(\Delta) & 1 \leq i < B,\ j \geq 0, \\ (vi) & P_{i,j,i,j+1}(continue) & = & 1 - (\lambda + \mu(j))\Delta + o(\Delta) & 1 \leq i < B,\ j \geq 0, \end{array}$$

where the state $(stop, stop)$ is where the process is finished. All the other transition probabilities are irrelevant. (i) describes the case when system rejuvenation is decided. When we decide to continue service, $(ii) - (iii)$ describe the situation when the buffer is empty. In this case either a new job arrives to the system, or nothing happens during the current time slot. $(iv) - (vi)$ stand for the cases when the buffer is not empty, then in addition to the previous case a job can leave the system since its service has finished $((v))$.

For any policy π we define the expected cost if the process was started in state (i,j):

$$V_\pi(i,j) = E_\pi \left[\sum_{w=0}^{\infty} C(i_w, j_w, a_w) \mid i_0 = i, j_0 = j \right], \quad 0 \leq i \leq B, 0 \leq j,$$

where (i_w, j_w) denotes the process state in $t = w\Delta$, and a_w is the action taken in $t = w\Delta$ according to the policy π.

Let

$$V(i,j) = \inf_{\pi} V_\pi(i,j), \quad 0 \le i \le B,\, 0 \le j.$$

The policy π^* is *optimal* if

$$V_{\pi^*}(i,j) = V(i,j), \quad \forall\, i,j:\ 0 \le i \le B,\, 0 \le j.$$

If f is a stationary policy which chooses action according to

$$f(i,j) = \arg\min_a \left\{ C(i,j,a) + \sum_{k=0}^{B-1} \sum_{l=0}^{\infty} P_{i,j,k,l}(a) V(k,l) \right\}, \quad 0 \le i \le B,\, 0 \le j, \tag{1}$$

then

$$V_f(i,j) = V(i,j), \quad 0 \le i \le B,\, 0 \le j,$$

hence f is optimal [?] ($\arg\min_a \{F(a)\}$ denotes the value of a where $F(a)$ is minimal).

Thus we have formulated the problem as a Markov Decision Process, for which stationary optimal policy exists, and it is determined by Equation 1 [?].

Substituting the transition probabilities we can write Equation 1 into a simpler form:

$$f(i,j) = \arg\min_a \left\{ C(i,j,a) + \sum_{k=0}^{B-1} P_{i,j,k,j+1}(a) V(k,l) \right\}, \quad 0 \le i \le B,\, 0 \le j.$$

The next step is to derive $V(i,j)$, the minimal expected cost in state (i,j) for all the states. We will define a series of expected cost functions, $\{V_n(i,j)\}$, or look-ahead-n cost functions, that are decreasing with n for all the states (i,j), and is an upper bound to the minimal cost function, V. We will also show, that the cost function C is an upper bound for the difference of the optimal and the look-ahead-n minimal cost functions, therefore in cases when the cost function tends to zero with time, the look-ahead cost function series V_n converges to the minimal cost function V. Bounds are given to the speed of the convergence. The proof of the above statements will follow the approach of the proof of Theorem 6.13 in [?].

Let

$$V_0(i,j) = C(i,j,stop) \quad 0 \le i \le B,\, 0 \le j,$$

and for $n > 0$,

$$V_n(i,j) = \min \left\{ C(i,j,stop), \sum_{k=0}^{B-1} P_{i,j,k,j+1}(continue) V_{n-1}(k,l) \right\} \quad 0 \le i \le B,\, 0 \le j. \tag{2}$$

If we start in state (i,j), $V_n(i,j)$ is the minimal expected cost if the process can go at most n stages before stopping. The expected cost cannot increase if we are allowed to go ahead, thus

$$V_n(i,j) \geq V_{n+1}(i,j) \geq V(i,j) \quad 0 \leq i \leq B,\, 0 \leq j. \tag{3}$$

The process is said to be *stable*, if $\lim_{n\to\infty} V_n(i,j) = V(i,j) \quad 0 \leq i \leq B,\, 0 \leq j.$

Let us also define $C_{max}(j) = \max_i \{C(i,j,stop)\}, \quad 0 \leq j.$

Theorem 1 *The difference of the minimal expected cost function and the look-ahead-n cost function satisfies the next inequality:*

$$V_n(i,j) - V(i,j) \leq C_{max}(n+j) \quad 0 \leq i \leq B,\, 0 \leq j. \tag{4}$$

Proof: Let f be an optimal policy, and let T denote the random time at which f stops. Also, let f_n be the policy which chooses the same actions as f at times $0, 1, \ldots, n-1$, but which stops at time n (if it had not previously done so). Then,

$$V(i,j) = V_f(i,j) = E_f\,[X \mid T \leq n]\, P\{T \leq n\} + E_f\,[X \mid T > n]\, P\{T > n\},$$

$$V_n(i,j) \leq V_{f_n}(i,j) = E_f\,[X \mid T \leq n]\, P\{T \leq n\} + E_{f_n}\,[X \mid T > n]\, P\{T > n\}.$$

where X denotes the total cost incurred and everything is understood to be conditional on $i_0 = i, j_0 = j$. Thus,

$$V_n(i,j) - V(i,j) \leq (E_{f_n}\,[X \mid T > n] - E_f\,[X \mid T > n])P\{T > n\}$$

$$\leq E_{f_n}\,[X \mid T > n]\,,$$

since $E_f\,[X \mid T > n] \geq 0$, for all the costs are nonnegative, and $P\{T > n\} \leq 1$.
If f_n stops after n stages, then

$$E_{f_n}\,[X \mid T > n] \leq C_{max}(n+j).$$

If f_n stops after $k < n$ stages, it happens, because doing the remaining $n-k$ steps would be more expensive, i.e.

$$E_{f_n}\,[X \mid T > n] \leq C_{max}(n+j). \quad \square$$

Summarizing, we can define an optimal policy f based on the minimal cost function V. V is not known, but can be approximated by the look-ahead cost function series V_n. We will refer to this approximation procedure as *MDP algorithm* in the sequel. If the cost function that gives the cost of stopping in a state converges to zero with time, then the approximation is stable, and an upper bound is given by Theorem 1 to the speed of the convergence of the cost function series V_n.

However, we do not know what n is sufficiently large to get the optimal decision. In other words, when V_n is close enough to V to result in the same policy, i.e. $f_n = f$. If the conditions of the following theorem hold for a state when the look-ahead policy is calculated for a certain depth, then we know that the look-ahead-n policy's decision is the same as the optimal policy would make. The following theorem discusses this criteria.

Theorem 2 *1. If $\exists n_0 : f_{n_0}(i,j) = continue$ then $\forall n \geq n_0 : f_n(i,j) = continue$ and $f(i,j) = continue$, i.e. the optimal policy will also decide to continue service in this state.*

2. If $\exists n_0 : f_{n_0}(i,j) = stop$ and

$$C(i,j,stop) < \sum_{k=0}^{B-1} P_{i,j,k,j+1}(continue)(V_{n_0}(k,l) - C_{max}(n_0 + l)) \tag{5}$$

then $\forall n \geq n_0 : f_n(i,j) = stop$ and $f(i,j) = stop$, i.e. the optimal policy will choose action stop in state (i,j).

Proof:

1. Since f_n chooses action according to

$$f_n(i,j) = \arg\min_a \left\{ C(i,j,a) + \sum_{k=0}^{B-1} P_{i,j,k,j+1}(a) V_n(k,l) \right\}, \quad 0 \leq i \leq B,\ 0 \leq j,$$

and $C(\cdot,\cdot,continue) = 0$, and the process is finished after a *stop* decision, $f_{n_0}(i,j) = continue$ means that

$$C(i,j,stop) \geq \sum_{k=0}^{B-1} P_{i,j,k,j+1}(continue) V_{n_0}(k,l).$$

Since the function series V_n is monotone decreasing (3)

$$\sum_{k=0}^{B-1} P_{i,j,k,j+1}(continue) V_{n_0}(k,l) \geq \sum_{k=0}^{B-1} P_{i,j,k,j+1}(continue) V_{n_0+1}(k,l),$$

and as a consequence of it

$$f_{n_0+1}(i,j) = continue.$$

From Equation (3) $V_n(i,j) \geq V(i,j)$, therefore the optimal policy will also continue the service.

2. $f_{n_0}(i,j) = stop$ means that

$$C(i,j,stop) < \sum_{k=0}^{B-1} P_{i,j,k,j+1}(continue) V_{n_0}(k,l).$$

From Theorem 1

$$V_{n_0}(i,j) - C_{max}(n_0 + j) \leq V(i,j).$$

Then simply substituting it to the rhs of (5), we obtain

$$C(i,j,stop) < \sum_{k=0}^{B-1} P_{i,j,k,j+1}(continue) V(k,l),$$

i.e. the optimal policy will stop in state (i,j).

On the other hand if we suppose that $\exists\, n \geq n_0$ such that $f_n(i,j) = continue$ then by Theorem 2/1 $f(i,j) = continue$ would follow, and that is a contradiction. □

Theorem 2 can be used to speed up the MDP algorithm, and also as a condition to finish the analysis. The transition probabilities of the underlying Markov process have a special structure, namely nonzero probabilities lead only to the next time step. The state space of a system is illustrated in Figure 2, the small squares refer to the states; the horizontal axis shows the number of time steps, while the vertical axis the buffer content. If we follow a sample path of the process, in each time step we move to the next column of squares. So if for all the states represented by the first k columns of Figure 2 the conditions of Theorem 2 hold, for larger n these columns can be ignored, since the optimal decision is known for them, and they have no effect on the rest of the states. Similarly, if the conditions hold for all the states of interest, the optimal solution is found.

The calculations defined in Theorem 2 are relatively simple, the magnitude of operations remains the same, i.e. $O(nBT + n^2B)$ if B is the length of the buffer, T is time range that is studied, and n is the depth of the analysis (look-ahead-n is used).

3.2 Simple Cost Function

Let the cost function be

$$C(b,t,stop) = \frac{b+\lambda T_R}{t+T_R}.$$

Since $b \le B$, $\lim_{t\to\infty} C_{max}(t) = 0$, the MDP solution will work according to Theorem 1. However, for this cost function the optimal decision can be derived explicitly for a large range of the states, and also an upper limit for the depth of the analysis, i.e. a limit n_U, will be derived, such that if $n \ge n_U$ then $f_n \equiv f$.

Theorem 3 *1. If $b \ge (\lambda - \mu(t))t - \mu(t)T_R$ holds for $1 \le b \le B$, then $f(b,t) = continue$.*

2. $\forall b, 0 \le b \le B : f(b,0) = continue$.

Proof: The condition for continuing the service is

$$C(b,t,stop) \ge \sum_{k=0}^{B-1} P_{b,t,k,t+\delta}(continue)V(k,t+\delta),$$

where δ denotes an arbitrarily small time interval. (The time is not discretized now.)

Since $V(k,t+\delta) \le C(k,t+\delta,stop)$, if

$$C(b,t,stop) \ge \sum_{k=0}^{B-1} P_{b,t,k,t+\delta}(continue)C(k,t+\delta,stop)$$

holds, then the service should be continued.

Substituting the cost function, we have the next inequalities:

- if $1 \le b \le B$

$$\frac{b+\lambda T_R}{t+T_R} \ge \lambda\delta\frac{b+1+\lambda T_R}{t+T_R+\delta} + \mu(t)\delta\frac{b-1+\lambda T_R}{t+T_R+\delta} + (1-(\lambda+\mu(t))\delta)\frac{b+\lambda T_R}{t+T_R+\delta},$$

- if $b = 0$

$$\frac{\lambda T_R}{t+T_R} \ge \lambda\delta\frac{1+\lambda T_R}{t+T_R+\delta} + (1-\lambda\delta)\frac{\lambda T_R}{t+T_R+\delta},$$

from which:

- if $1 \le b \le B$

$$b \ge (\lambda-\mu(t))t - \mu(t)T_R, \tag{6}$$

- if $b = 0$

$$0 \geq \lambda \delta t, \tag{7}$$

$\square$

According to Theorem 3 in case of a nonempty buffer, we have a simple rule to decide about the continuation of the service: if (6) holds, we should continue service. However it doesn't mean that if (6) does not hold, we should stop, since the initial approximation of $V(b,t)$ by $C(b,t,stop)$.

For the case of an empty buffer, we did not get a general simple rule, (7) holds only for $t = 0$, i.e. in $t = 0$ we should continue service. In the rest of the cases of an empty buffer the MDP algorithm can help.

In case of nonempty buffer, if $\lambda \leq \mu(t)$, in other words the service intensity is not less than the arrival rate, the service should be continued independently of the number of jobs in the system.

Another interesting result is that if the buffer continues more jobs than a certain limit at time t, the service should be continued - the more jobs are in the buffer, the more the need is to continue the service.

Theorem 4 *If $\exists t_{limit}$ such that in t_{limit} the system will be stopped anyway, then if $B \leq (\lambda - \mu(t))t - \mu(t)T_R$ then $f(b,t) = stop$ for $\forall b : 0 \leq b \leq B$.*

Proof: Suppose that $f(b, t+\delta) = stop\ \forall b, 0 \leq b \leq B$. The condition for stopping the service in t is

$$C(b,t,stop) \leq \sum_{k=0}^{B-1} P_{b,t,k,t+\delta}(continue) V(k, t+\delta).$$

Since $V(k,t+\delta) = C(k, t+\delta, stop)$, if

$$C(b,t,stop) \leq \sum_{k=0}^{B-1} P_{b,t,k,t+\delta}(continue) C(k, t+\delta, stop)$$

holds, then the service should be stopped.

Substituting the cost function, we have

- $1 \leq b \leq B$

$$\frac{b+\lambda T_R}{t+T_R} \leq \lambda\delta \frac{b+1+\lambda T_R}{t+T_R+\delta} + \mu(t)\delta \frac{b-1+\lambda T_R}{t+T_R+\delta} + (1-(\lambda+\mu(t))\delta)\frac{b+\lambda T_R}{t+T_R+\delta}$$

- $b = 0$

$$\frac{\lambda T_R}{t + T_R} \leq \lambda\delta \frac{1 + \lambda T_R}{t + T_R + \delta} + (1 - \lambda\delta) \frac{\lambda T_R}{t + T_R + \delta}$$

Simplifying the results we have:

- $1 \leq b \leq B$

$$b \leq (\lambda - \mu(t))t - \mu(t) T_R \tag{8}$$

- $b = 0$

$$0 \leq \lambda\delta t \tag{9}$$

Since $b \leq B$ and (9) holds for all $t \geq 0$ the theorem is proven. □

The assumption that the system will be stopped once is quite reasonable, e.g. because of hardware maintenance.

Since $\mu(t)$ is decreasing such that $\lambda > \mu(t)$ for large t, the condition of the statement will be satisfied as time progresses.

An upper limit has been derived for the time to stop the system. Together with the result of Theorem 3 it may be enough to define a policy in practical cases, since we know the optimal decision for $t \geq \frac{B+\mu(t)T_R}{\lambda-\mu(t)}$ and for $t \leq \frac{b+\mu(t)T_R}{\lambda-\mu(t)}$, where b is the buffer content at time t. The region where we have no explicit answer for the question of optimal decision is

$$\frac{b + \mu(t)T_R}{\lambda - \mu(t)} \leq t \leq \frac{B + \mu(t)T_R}{\lambda - \mu(t)}.$$

If this region is narrow enough then there is no need to run the MDP algorithm.

If we want to know the optimal policy in the region where Theorem 3 and Theorem 4 do not help, we have to run the MDP algorithm. However, we know that if $n \geq n_U = \frac{t_{limit}}{\Delta}$ then $f_n \equiv f$, since the optimal decision in $t \geq t_{limit}$ is known, i.e. Theorem 4 reduces the problem to be a finite time problem. The assumption that the system will be stopped at a time t_{limit} does not implies finite time analysis since its value is assumed not to be known.

4 Buffer Overflow Case

In this system model we assume that when the buffer is full, and a new job arrives to the system, the job is lost, but the system does not have to stop and rejuvenate, however is allowed to do so. For the analysis we have to introduce another variable to describe the actual system state, since we have to remember the number of lost jobs.

4.1 MDP Solution

The optimization problem is slightly modified by introducing a new random variable, L, describing the number of lost jobs at the time T when rejuvenation is decided:
find T that minimizes the average cost of the run

$$\min_T \Big\{ E\left[C(X,T,Y,L)\right] \Big\},$$

if λ, $\mu(t)$, T_R and B are given.

The cost function structure is defined as follows:

$$\begin{array}{lcll} C(i,j,k,stop) & \geq & 0, & 0 \leq i \leq B,\ 0 \leq j, 0 \leq k \leq j, \\ C(i,j,k,continue) & = & 0, & 0 \leq i < B,\ 0 \leq j, 0 \leq k \leq j, \end{array}$$

where i and j are defined as in Section 3, while k denotes the number of lost jobs until time $t = j\Delta$. The same approximation is used for Y. We require all the costs to be nonnegative.

We define the probability $P_{i,j,k,p,q,r}(a)$ as the probability of going from state (i,j,k) to state (p,q,r) when action a is chosen. In our case the transition probabilities are defined as follows:

$$\begin{array}{llcll} (i) & P_{\cdot,\cdot,\cdot,stop,stop,stop}(stop) & = & 1, & \\ (ii) & P_{0,j,k,1,j+1,k}(continue) & = & \lambda\Delta + o(\Delta) & j \geq 0,\ 0 \leq k \leq j, \\ (iii) & P_{0,j,k,0,j+1,k}(continue) & = & 1 - \lambda\Delta + o(\Delta) & j \geq 0,\ 0 \leq k \leq j, \\ (iv) & P_{i,j,k,i+1,j+1,k}(continue) & = & \lambda\Delta + o(\Delta) & 1 \leq i < B,\ j \geq 0,\ 0 \leq k \leq j, \\ (v) & P_{i,j,k,i-1,j+1,k}(continue) & = & \mu(j)\Delta + o(\Delta) & 1 \leq i < B,\ j \geq 0,\ 0 \leq k \leq j, \\ (vi) & P_{i,j,k,i,j+1,k}(continue) & = & 1 - (\lambda + \mu(j))\Delta + o(\Delta) & 1 \leq i < B,\ j \geq 0,\ 0 \leq k \leq j, \\ (vii) & P_{B,j,k,B-1,j+1,k}(continue) & = & \mu(j)\Delta + o(\Delta) & j \geq 0,\ 0 \leq k \leq j, \\ (viii) & P_{B,j,k,B,j+1,k}(continue) & = & 1 - (\lambda + \mu(j))\Delta + o(\Delta) & j \geq 0,\ 0 \leq k \leq j, \\ (ix) & P_{B,j,k,B,j+1,k+1}(continue) & = & \lambda\Delta + o(\Delta) & j \geq 0, 0 \leq k \leq j, \end{array}$$

where the state $(stop, stop, stop)$ is where the process is finished. The above definitions $(i) - (ix)$ follow the same discipline as in Section 3, the slight difference is that we have to define probabilities for the case, when the buffer is full, and service continuation is chosen $((vii) - (ix))$.

We define the same functions and policies as in Section 3. For any policy π,

$$V_\pi(i,j,k) = E_\pi\left[\sum_{w=0}^{\infty} C(i_w, j_w, k_w, a_w) \mid i_0 = i, j_0 = j\right], \quad 0 \leq i \leq B,\ 0 \leq j,\ 0 \leq k \leq j,$$

i.e. the expected cost if the process was started in state (i,j,k). (i_w, j_w, k_w) denotes the process state in $t = w\Delta$, and a_w is the action taken in $t = w\Delta$ according to the policy π. Let

$$V(i,j,k) = \inf_\pi V_\pi(i,j,k), \quad 0 \leq i \leq B,\ 0 \leq j,\ 0 \leq k \leq j.$$

The policy π^* is optimal if

$$V_{\pi^*}(i,j,k) = V(i,j,k), \quad \text{for all } i,j,k: \ 0 \le i \le B,\, 0 \le j,\, 0 \le k \le j.$$

If f is a stationary policy which chooses action according to

$$f(i,j,k) = \arg\min_a \Big\{ C(i,j,k,a) + \sum_{p=0}^{B-1} \sum_{q=0}^{\infty} \sum_{r=0}^{q} P_{i,j,k,p,q,r}(a)\ V(p,q,r) \Big\}, \tag{10}$$

where $0 \le i \le B$, $0 \le j$, $0 \le k \le j$, then

$$V_f(i,j,k) = V(i,j,k), \quad 0 \le i \le B,\, 0 \le j,\, 0 \le k \le j,$$

hence f is optimal [**?**].

Thus we have formulated the problem as a Markov Decision Process, for which stationary optimal policy exists, and it is determined by Equation 10 [**?**].
Substituting the transition probabilities we can write Equation 10 into a simpler form:

$$f(i,j,k) = \arg\min_a \Big\{ C(i,j,k,a) + \textstyle\sum_{p=0}^{B-1} \sum_{r=0}^{j+1} P_{i,j,k,p,j+1,r}(a)\ V(p,q,r) \Big\}, \qquad 0 \le i \le B,\, 0 \le j,\, 0 \le k \le j \tag{11}$$

We carry on the same way as in Section 3. Let

$$V_0(i,j,k) = C(i,j,k,stop) \quad 0 \le i \le B,\, 0 \le j,\, 0 \le k \le j,$$

and for $n > 0$,

$$V_n(i,j,k) = \min \Big\{ C(i,j,k,stop),\ \sum_{p=0}^{B} \sum_{r=0}^{j+1} P_{i,j,k,p,j+1,r}(continue) V_{n-1}(p,q,r) \Big\},$$

where $0 \le i \le B$, $0 \le j$, $0 \le k \le j$. If we start in state (i,j,k), $V_n(i,j,k)$ is the minimal expected cost if the process can go at most n stages before stopping. The expected cost cannot increase if we are allowed to go ahead, thus

$$V_n(i,j,k) \ge V_{n+1}(i,j,k) \ge V(i,j,k) \quad 0 \le i \le B,\, 0 \le j,\, 0 \le k \le j.$$

The process is said to be *stable*, if

$$\lim_{n\to\infty} V_n(i,j,k) = V(i,j,k) \quad 0 \le i \le B,\, 0 \le j,\, 0 \le k \le j.$$

Let us also define

$$C_{max}(j) = \max_{i,k} \{ C(i,j,k,stop) \} \quad 0 \le i \le B,\, 0 \le j,\, 0 \le k \le j.$$

Theorem 5 *The difference of the minimal expected cost function and the look-ahead-n cost function satisfies the next inequality:*

$$V_n(i,j,k) - V(i,j,k) \leq C_{max}(n+j) \quad 0 \leq i \leq B,\, 0 \leq j,\, 0 \leq k \leq j.$$

Proof: The same way as in the previous section. □

As in the previous section, we can define an optimal policy f based on the minimal cost function V. V is approximated by the look-ahead cost function series V_n (MDP algorithm). If the cost function that gives the cost of stopping in a state converges to zero with time, then the approximation is stable, and an upper bound is given by Theorem 5 to the speed of the convergence.

Similarly if the conditions of the following theorem hold for a state when the look-ahead policy is calculated for a certain depth, then we know that the look-ahead-n policy's decision is the same as the optimal policy would make.

Theorem 6 1. *If $\exists\, n_0 : f_{n_0}(i,j,k) = continue$ then $\forall n \geq n_0 : f_n(i,j,k) = continue$ and $f(i,j,k) = continue$, i.e. the optimal policy will also decide to continue service in state (i,j,k).*

2. *If $\exists n_0 : f_{n_0}(i,j,k) = stop$ and*

$$C(i,j,k,stop) < \sum_{p=0}^{B} \sum_{r=0}^{j+1} P_{i,j,k,p,j+1,r}(continue)(V_{n_0}(p,q,r) - C_{max}(n_0+q)) \quad (12)$$

then $\forall n \geq n_0 : f_n(i,j,k) = stop$ and $f(i,j,k) = stop$, i.e. the optimal policy will choose action stop in state (i,j,k).

Proof: The same way as in the previous section. □

Theorem 6 can be used to speed up the MDP algorithm, and also as a condition to finish the analysis in the same way as Theorem 2.

The number of operations is higher now due to the additional variable that makes possible to remember the number of lost jobs: $O(nBT^2 + n^2BT + n^3B)$.

4.2 Simple Cost Function

Let the cost function be

$$C(b,t,L,stop) = \frac{b+\lambda T_R + L}{t+T_R},$$

where the time is not discretized. For this cost function $\lim_{t\to\infty} C_{max}(t) = 0$ does not hold, so Theorem 5 cannot be applied[1].

Similarly to Section 3.2 the optimal decision can be derived for a range of the states. However, since the number of lost jobs is not bounded from above, an explicit upper limit for the depth of the necessary analysis cannot be determined. The results contain the r.v. L, the number of lost jobs, so the final formulas can be used to make in operation decisions, since then the number of already lost jobs is known.

Theorem 7

1. *If $b \geq (\lambda - \mu(t))t - \mu(t)T_R - L$ holds for $1 \leq b \leq B$, then $f(b,t) = continue$.*
2. *If $L \geq \lambda t$ then $f(0,t) = continue$.*

Proof: The condition for continuing the service is

$$C(b,t,L,stop) \geq \sum_{k=0}^{B}\sum_{L=0}^{\infty} P_{b,t,L,k,t+\delta,l}(continue)V(k,t+\delta,l).$$

Since $V(k,t+\delta,l) \leq C(k,t+\delta,l,stop)$, if

$$C(b,t,l,stop) \geq \sum_{k=0}^{B}\sum_{L=0}^{\infty} P_{b,t,L,k,t+\delta,l}(continue)C(k,t+\delta,l,stop).$$

holds, then the service should be continued.
Substituting the cost function, we have

- $b = B$

$$\begin{aligned}\frac{B+\lambda T_R+L}{t+T_R} \quad &\geq \quad \lambda\delta\frac{B+\lambda T_R+L+1}{t+T_R+\delta} + \mu(t)\delta\frac{B-1+\lambda T_R+L}{t+T_R+\delta}\\ &\quad +(1-(\lambda+\mu(t))\delta)\frac{B+\lambda T_R+L}{t+T_R+\delta}\end{aligned}$$

[1]However, if the cost function is modified to

$$C(b,t,L,stop) = \frac{b+\lambda T_R+L}{t^{1+\varepsilon}+T_R},$$

where $\varepsilon > 0$, $C_{max}(t)$ tends to zero with t, i.e. the condition of Theorem 5 holds.

- $1 \leq b \leq B-1$

$$\begin{aligned}\frac{b+\lambda T_R+L}{t+T_R} &\geq \lambda\delta\frac{b+1+\lambda T_R+L}{t+T_R+\delta}+\mu(t)\delta\frac{b-1+\lambda T_R+L}{t+T_R+\delta}\\ &+(1-(\lambda+\mu(t))\delta)\frac{b+\lambda T_R+L}{t+T_R+\delta}\end{aligned}$$

- $b=0$

$$\frac{\lambda T_R+L}{t+T_R} \geq \lambda\delta\frac{1+\lambda T_R+L}{t+T_R+\delta}+(1-\lambda\delta)\frac{\lambda T_R+L}{t+T_R+\delta}$$

Simplifying the results we have:

- $b=B$

$$B \geq (\lambda-\mu(t))t-\mu(t)T_R-L$$

- $1 \leq b \leq B-1$

$$b \geq (\lambda-\mu(t))t-\mu(t)T_R-L$$

- $b=0$

$$L \geq \lambda t$$

□

A rule has been derived also for the empty buffer case, however it is unlikely that it will hold for $t>0$. We can notice that the derived decision rule for $b=B$ and $1 \leq b \leq B$ cases is the same, and if we substitute $L=0$ to the final results, we get the results of Section 3.2.

Theorem 8 *If $\exists t_{limit}$ such that in t_{limit} the system will be stopped anyway, then if $B+L \leq (\lambda-\mu(t))t-\mu(t)T_R$ then $f(b,t)=stop$ for $\forall b: 0 \leq b \leq B$.*

Proof: Suppose that $f(b,t+\delta)=stop$ for $\forall b: 0 \leq b \leq B$. The condition for stopping the service in t is

$$C(b,t,L,stop) \geq \sum_{k=0}^{B}\sum_{L=0}^{\infty} P_{b,t,L,k,t+\delta,l}(continue)V(k,t+\delta,l).$$

Since $V(k,t+\delta,l)=C(k,t+\delta,l,stop)$, if

$$C(b,t,l,stop) \leq \sum_{k=0}^{B}\sum_{L=0}^{\infty} P_{b,t,L,k,t+\delta,l}(continue)C(k,t+\delta,l,stop)$$

holds, then the service should be continued.

Substituting the cost function, we have

- if $b = B$

$$\begin{aligned}\frac{B+\lambda T_R+L}{t+T_R} &\le \lambda\delta\frac{B+\lambda T_R+L+1}{t+T_R+\delta}+\mu(t)\delta\frac{B-1+\lambda T_R+L}{t+T_R+\delta}\\ &+(1-(\lambda+\mu(t))\delta)\frac{B+\lambda T_R+L}{t+T_R+\delta},\end{aligned}$$

- if $1 \le b \le B-1$

$$\begin{aligned}\frac{b+\lambda T_R+L}{t+T_R} &\le \lambda\delta\frac{b+1+\lambda T_R+L}{t+T_R+\delta}+\mu(t)\delta\frac{b-1+\lambda T_R+L}{t+T_R+\delta}\\ &+(1-(\lambda+\mu(t))\delta)\frac{b+\lambda T_R+L}{t+T_R+\delta},\end{aligned}$$

- if $b = 0$

$$\frac{\lambda T_R+L}{t+T_R} \le \lambda\delta\frac{1+\lambda T_R+L}{t+T_R+\delta}+(1-\lambda\delta)\frac{\lambda T_R+L}{t+T_R+\delta}.$$

Simplifying the results we have:

- if $b = B$

$$B \le (\lambda-\mu(t))t-\mu(t)T_R-L,$$

- if $1 \le b \le B-1$

$$b \le (\lambda-\mu(t))t-\mu(t)T_R-L, \tag{13}$$

- if $b = 0$

$$L \le \lambda t. \tag{14}$$

Since $b \le B$ and (13) implies (14), the theorem is proven. □

The assumption that the system will be stopped once is justified in this case as well, however we can not state that the condition of this theorem will be fulfilled as time progresses, so the problem is not reduced to a finite time problem.

Similarly to the previous section's results, we know the optimal decision for $t \ge \frac{B+L+\mu(t)T_R}{\lambda-\mu(t)}$ and for $t \le \frac{b+L+\mu(t)T_R}{\lambda-\mu(t)}$, where b is the buffer content at time t, and L is the number of lost customers in $(0,t)$. We have no answer for the question of optimal decision when

$$\frac{b+L+\mu(t)T_R}{\lambda-\mu(t)} \le t \le \frac{B+L+\mu(t)T_R}{\lambda-\mu(t)}.$$

Naturally this theorem can be used to make in operation decisions, when L is known.

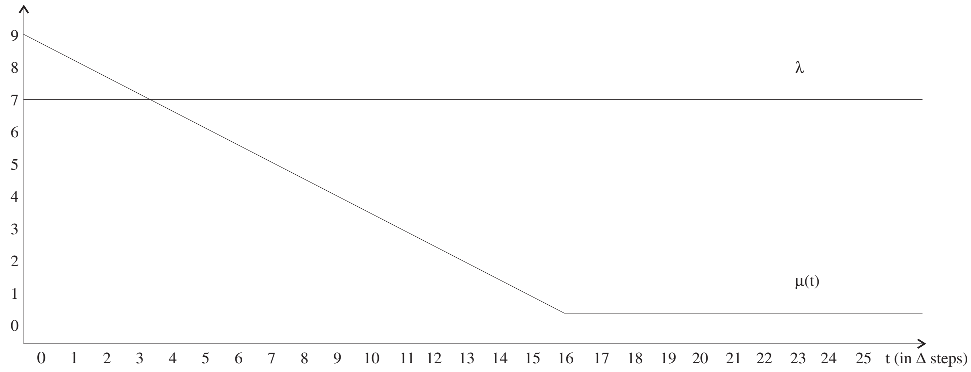

Figure 1: Arrival rate (λ) and service rate ($\mu(t)$) of the analyzed system

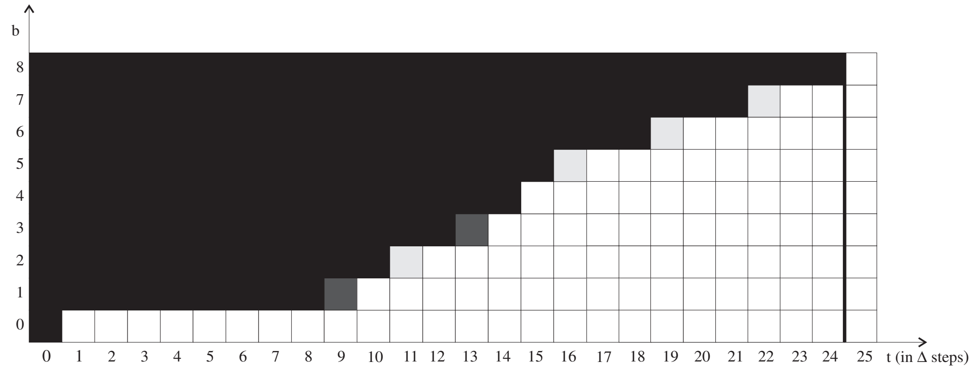

Figure 2: Decision map of the analyzed system

5 Numerical Example

A simple system was analyzed to demonstrate the discussed methods for the non-overflow case, using the analysed simple cost function. The buffer length was 8, and the analysis included the first 26 time steps where $\Delta = 0.05$ and $T_R = 2\Delta$ (it is not a real application, only an illustration of the analysis). The arrival rate and service rate are illustrated in Figure 1. The decision map is illustrated in Figure 2. The black area refers to the states where Theorem 3 yields "continue" decision. On the other hand, using the result of Theorem 4 we can predict the time limit of the "continue" decisions. Suppose that this limit will be where $\mu(t) = \mu = 0.5$ (see Figure 1):

$$t \geq \frac{B + \mu T_R}{\lambda - \mu} = \frac{8 + 0.0025}{7 - 0.5} \approx 1.23115 \approx 24.6\Delta,$$

i.e. we expect no "continue" decision beyond 24Δ, that is represented by the thick vertical line. From Theorem 4 and Theorem 3 the uncertain region is between the black area and the vertical line, the optimal policy is not predicted for these states.

As we can see the results are verified by the MDP algorithm. The MDP method has been programmed in *Mathematica 2.1*, and it was run for the above system with several look-ahead depths. The light grey area (three states) refers to the states where (in addition to the black area) the MDP algorithm with depth 1 yielded "continue" decision, and the dark grey area (two states) refers to the states where (in addition to the black and light grey area) the MDP algorithm with depth 3 yielded "continue" decision. The algorithm was run with look-ahead-25 policy as well, but the decision map did not differ from the look-ahead-3 map. (We know from Theorem 4 that there is no point in running the algorithm for higher depths.) Unfortunately, we could not make use of Theorem 2/2 since the condition of the statement was not fulfilled in any of the cases.

6 Conclusion

The problem of determining the optimal time to rejuvenate a server software is addressed in the paper, when it experiences soft failures due to aging. Poisson arrival process is assumed, and the service rate of the software is assumed to be gradually decreasing with time. Two queueing policies are analysed, namely when buffer overflow is allowed or not. The considered problem is to choose an optimal time interval for rejuvenation to minimize the cost of lost jobs.

The problem is formulated as a Markov Decision Process, more specifically as an optimal stopping problem. The general algorithm to present the optimal policy is proved to work if the cost function tends to zero with time. Additional criteria are derived for the states that can be used to make the algorithm faster.

A simple intuitive cost function is analyzed for each queueing policy. For the case when no buffer overflow is allowed, simple explicit rules are derived determining the optimal policy for most of the states. For the case when buffer overflow is allowed the rules are not explicit since contain the number of lost jobs as a variable, so in this case these rules can be used to make in operation decisions.

The results are demonstrated in a simple numerical example for the case when buffer overflow is not allowed.