

Leonardo Miranda do Nascimento

**Análise de desempenho sequencial de sistemas  
que envelhecem e rejuvenescem**

Rio de Janeiro, Brasil

Março, 2023

Leonardo Miranda do Nascimento

## **Análise de desempenho sequencial de sistemas que envelhecem e rejuvenescem**

**Dissertação de Mestrado** submetida ao Programa de Pós-graduação em Informática do Instituto de Matemática e do Instituto Tércio Pacitti da Universidade Federal do Rio de Janeiro - UFRJ, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Universidade Federal do Rio de Janeiro – UFRJ

Instituto de Matemática

Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais

Programa de Pós-Graduação em Informática

Ph.D. Daniel Sadoc Menasché

Docteur Josefino Cabral Melo Lima

Rio de Janeiro, Brasil

Março, 2023

Leonardo Miranda do Nascimento

Análise de desempenho sequencial de sistemas que envelhecem e rejuvenescem/  
Leonardo Miranda do Nascimento. – Rio de Janeiro, Brasil, Março, 2023-  
81p. : il. (algumas color.) ; 30 cm.

Ph.D. Daniel Sadoc Menasché

Dissertação (Mestrado) – Universidade Federal do Rio de Janeiro – UFRJ  
Instituto de Matemática  
Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais  
Programa de Pós-Graduação em Informática, Março, 2023.

1. Envelhecimento de software. 2. Rejuvenescimento. 3. Modelos de Markov. 4.  
Sistema de controle I. Orientador Ph.D. Daniel Sadoc Menasché. II. Universidade  
Federal do Rio de Janeiro. III. Programa de Pós-Graduação em Informática. IV. Análise  
de desempenho sequencial de sistemas que envelhecem e rejuvenescem

# ERRATA

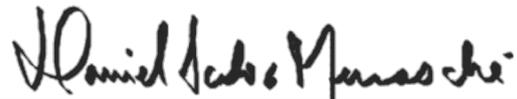
Folha	Linha	Onde se lê	Leia-se
-------	-------	------------	---------

# ANÁLISE DE DESEMPENHO SEQUENCIAL DE SISTEMAS QUE ENVELHECEM E REJUVENESCEM

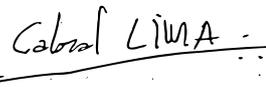
LEONARDO MIRANDA DO NASCIMENTO

**Dissertação de Mestrado** submetida ao Programa de Pós-graduação em Informática do Instituto de Matemática e do Instituto Tércio Pacitti da Universidade Federal do Rio de Janeiro - UFRJ, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Aprovada em 31 de março de 2023 por:



Prof. Daniel Sadoc Menasché, Ph.D., PPGI/UFRJ (Presidente)



Prof. Josefino Cabral Melo Lima, Doutor, PPGI/UFRJ

Documento assinado digitalmente



CLAUDIO MICELI DE FARIAS  
Data: 04/04/2023 15:02:16-0300  
Verifique em <https://validar.iti.gov.br>

Prof. Claudio Miceli de Farias, D.Sc., PESC/UFRJ e PPGI/UFRJ



Prof. Guilherme de Melo Baptista Domingues, D.Sc. UERJ



Alberto Avritzer, Ph.D., eSulabSolutions, Inc.



Vilc Queupe Rufino, Dr., Marinha do Brasil

*A todos que fizeram do conhecimento ferramenta  
para tornar o mundo melhor.*

# AGRADECIMENTOS

Gostaria de agradecer a todos que contribuíram neste processo de aprendizado para que eu pudesse finalizar esta jornada.

Primeiro, meu eterno agradecimento ao meu orientador Prof. Dr. Daniel Sadoc Menasché e o meu co-orientador Prof. Dr. Josefino Cabral Melo Lima, pela paciência, dedicação e generosidade de terem compartilhado comigo tamanho conhecimento e experiência acadêmica.

Agradeço a todos os professores do PPGI que contribuíram com ensinamentos fundamentais para a conclusão deste trabalho.

Agradeço ao Prof. Dr. Guilherme Domingues da UERJ por toda ajuda na elaboração da parte final deste trabalho, particularmente no que toca o modelo de malha fechada apresentado no Capítulo 3.

Um agradecimento especial à Marinha do Brasil por me oportunizar esta experiência única de capacitação de vida e de carreira.

Não poderia deixar de agradecer também aos colegas de curso pelos bons momentos e aprendizado em conjunto.

Por último, mas não menos importante, gostaria de agradecer a minha família, especialmente minha esposa Cristiane, pela ajuda, paciência, dedicação e incentivo.

*“A educação exige os maiores cuidados,  
porque influi sobre toda a vida.”  
(Sêneca)*

# RESUMO

*Análise de desempenho sequencial* visa avaliar indicadores de desempenho de forma online. O processo interrompe a sua execução de acordo com uma regra de parada pré-definida, assim que a quantidade de anomalias observadas atinge a regra de parada, o que deverá produzir um alarme. As técnicas tradicionais de análise de desempenho sequencial incluem CUSUM e teste de razão de probabilidade sequencial (SPRT). Técnicas mais recentes incluem o algoritmo de *bucket*, em que os *tokens* são acumulados em *buckets* quando o sistema se degrada e removidos quando o sistema se recupera naturalmente. Se o número de *tokens* no sistema atingir um limite, um alarme será acionado. O objetivo deste trabalho é analisarmos o algoritmo de análise de desempenho sequencial (algoritmo de *bucket*) aplicado a sistemas que envelhecem e rejuvenescem a luz de três abordagens: 1) Sem informação do sistema, rejuvenescimento aleatório, quando o rejuvenescimento acontece a qualquer momento e sem relação direta com o estado do sistema, 2) Informação parcial do sistema, quando implementamos um modelo de cadeia de Markov com três dimensões (estado da fonte de anomalias, algoritmo de detecção de anomalias, estado da fila do sistema), nesta abordagem o rejuvenescimento é sinalizado ao sistema pelo estouro do *bucket*, representando uma criticidade do sistema e a necessidade de intervenção dos administradores, e 3) Informação completa do sistema, onde, utilizando a abordagem 2, simulando uma situação hipotética de conhecimento do estado da fonte de anomalias e estado da fila do sistema, refletido pelo estado do *bucket*, nesta abordagem, uma vez no estado anômalo o rejuvenescimento é aplicado. Desta forma, as abordagens 2 e 3 pretendem simular um controle de malha fechada, onde os valores de saída do sistema em desacordo com o esperado geram uma ação de correção, visando a normalização do funcionamento do sistema.

**Palavras-chave:** envelhecimento de *software*, rejuvenescimento, modelos de Markov, controle malha fechada, algoritmo de *bucket*.

# ABSTRACT

Sequential performance analysis aims to evaluate performance indicators online. The process interrupts its execution according to a predefined stop rule, as soon as the number of observed anomalies reaches the stop rule, which should produce an alarm. Traditional sequential performance analysis techniques include CUSUM and sequential probability ratio test (SPRT). More recent techniques include the bucket algorithm, where tokens are accumulated in buckets when the system degrades and removed when the system naturally recovers. If the number of tokens in the system reaches a limit, an alarm will be triggered. The objective of this work is to analyze the sequential performance analysis algorithm (bucket algorithm) applied to systems that age and rejuvenate in light of three approaches: 1) Without system information, random rejuvenation, when rejuvenation occurs at any time and without direct relation to the system state, 2) Partial system information, when we implement a Markov chain model with three dimensions (anomaly source state, anomaly detection algorithm, system queue state), in this approach rejuvenation is signaled to the system by the bucket overflow, representing system criticality and the need for administrator intervention, and 3) Complete system information, where, using approach 2, simulating a hypothetical situation of knowledge of the anomaly source state and system queue state, which is reflected by the bucket state, in this approach, once in the anomalous state, rejuvenation is applied. Thus, approaches 2 and 3 aim to simulate a closed-loop control, where system output values that deviate from the expected generate a correction action, aiming at normalizing the system operation.

**Keywords:** Software aging, rejuvenation, Markov models, closed-loop control, bucket algorithm.

# LISTA DE ILUSTRAÇÕES

Figura 1 – Processo MMPP alimentando fila gargalo . . . . .	20
Figura 2 – Processo MMPP, <i>bucket</i> e $M/M/1$ , correspondendo a um sistema com três dimensões (três variáveis de estado) . . . . .	21
Figura 3 – Diagrama do modelo integrado. Cada elemento no diagrama acima corresponde a uma dimensão do modelo: fonte de anomalias, fila gargalo do sistema e estado do algoritmo de detecção de anomalias. . . . .	29
Figura 4 – Cenário 1: Tempo médio de permanência no estado anômalo $A$ , com $D = 12$ , variando taxa $\theta$ , (a) sem o rejuvenescimento, e (b) com rejuvenescimento, e $\delta_{RG} = 0,1$ . . . . .	34
Figura 5 – (a) Comparativo de tempo de permanência no estado anômalo e profundidade do <i>bucket</i> , variando taxa $\theta$ e com $\delta_{RG} = 0,1$ ; (b) Relação entre número de rejuvenescimentos aplicados e profundidade do <i>bucket</i> . . . . .	36
Figura 6 – Relação entre detecção de falso positivos e profundidade do <i>bucket</i> . . . . .	40
Figura 7 – O papel da informação na tomada de decisão: (a) chegadas e partidas de <i>jobs</i> do sistema nos casos de informação parcial e informação completa com $\mu < \lambda_A$ e $\mu > \lambda_A$ ; (b) <i>jobs</i> não atendidos pelo sistema e profundidade do <i>bucket</i> . . . . .	43
Figura 8 – Clássico algoritmo de <i>bucket</i> , com envelhecimento e rejuvenescimento . . . . .	45
Figura 9 – Comparação de quantis sem rejuvenescimento . . . . .	50
Figura 10 – Comparação de quantis com rejuvenescimento . . . . .	50
Figura 11 – O rejuvenescimento ajuda a evitar falsos alarmes: a Figura mostra um aumento no número médio de passos até que o <i>bucket</i> transborde (falso positivo) e contrasta a simulação com um modelo analítico que aproxima o comportamento das simulações por meio de uma distribuição geométrica. . . . .	52
Figura 12 – Abordagem de aplicação do rejuvenescimento aleatório: porcentagem dos rejuvenescimentos aplicados sem que o limite do <i>bucket</i> e tenha sido atingido. . . . .	54
Figura 13 – Abordagem de aplicação do rejuvenescimento aleatório: quantidade de passos até se atingir o limite do <i>bucket</i> . . . . .	54
Figura 14 – Abordagem de aplicação do rejuvenescimento oriundo do estouro do <i>bucket</i> : quantidade de passos até se atingir o limite do <i>bucket</i> . . . . .	55
Figura 15 – Algoritmo de <i>bucket</i> com sobrecarga devido ao rejuvenescimento e anomalias . . . . .	56
Figura 16 – Ilustrando um cenário em que o rejuvenescimento não é benéfico $r_1 = 0,1, r'_1 = 0,1, r_2 = 0,1, r'_2 = 0,1, r_3 = 0,1, \lambda = 0,05$ . . . . .	57

Figura 17 – Ilustrando um cenário em que o rejuvenescimento é benéfico diminuindo $r'_1$ e $\lambda$ para 0,01 e 0,03 e aumentando $r_3$ para 20. Os resultados correspondem e ilustram a aplicabilidade de (4.24). . . . .	57
Figura 18 – Impacto do número de estados de envelhecimento. . . . .	58
Figura 19 – Dinâmica MMPP utilizando CTMC . . . . .	78
Figura 20 – Agregação de fluxos de Poisson: . . . . .	80
Figura 21 – Desagregação de fluxos de Poisson: . . . . .	81

# LISTA DE TABELAS

Tabela 1 – Contribuições . . . . .	18
Tabela 2 – Cadeia de Markov com 3 variáveis de estado: (estado da fonte de anomalias, estado da fila gargalo, estado do algoritmo de <i>bucket</i> ). Assumimos uma fila gargalo $M/M/1/2$ . . . . .	30
Tabela 3 – Tabela de notação . . . . .	33
Tabela 4 – Estatísticas sobre número de transações atendidas e tempo nos estados $G$ e $A$ . . . . .	34
Tabela 5 – Cenário 2: Estatísticas sobre número de transações atendidas e tempo nos estados $G$ e $A$ , variando taxa $\theta$ . . . . .	36
Tabela 6 – Detecção de transições, sem rejuvenescimento . . . . .	39
Tabela 7 – Detecção de transições, com rejuvenescimento . . . . .	39
Tabela 8 – Comparativo de aplicabilidade de rejuvenescimento ajustando taxas . . . . .	42
Tabela 9 – Modelo analítico . . . . .	51
Tabela 10 – Comparação entre as abordagens de rejuvenescimento . . . . .	53

# LISTA DE ABREVIATURAS E SIGLAS

CUSUM	Cumulative Sum Control Chart
SGBD	Sistema Gerenciador de Banco de Dados
MySQL	My Structured Query Language
HPC	High Computation Performance
TLP	Transiente Loss Probability

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b>	<b>17</b>
<b>1.1</b>	<b>Contribuições</b>	<b>18</b>
<b>1.2</b>	<b>Artefatos produzidos</b>	<b>19</b>
<b>1.3</b>	<b>Roteiro</b>	<b>19</b>
<b>2</b>	<b>DESCRIÇÃO DO SISTEMA</b>	<b>20</b>
<b>2.1</b>	<b>Ingredientes básicos</b>	<b>20</b>
2.1.1	Fonte de tarefas e anomalias	20
2.1.2	Rejuvenescimento	20
2.1.3	Algoritmo de <i>bucket</i>	21
2.1.3.1	Amostras de <i>throughput</i>	22
2.1.3.2	Relação entre o estado do sistema e o estado do <i>bucket</i>	22
<b>2.2</b>	<b>Informação disponível para tomada de decisão</b>	<b>22</b>
<b>2.3</b>	<b>Malha aberta versus malha fechada</b>	<b>23</b>
<b>2.4</b>	<b>Análise do sistema em malha aberta e fechada</b>	<b>24</b>
2.4.1	Sistema em malha fechada	24
2.4.2	Sistema malha aberta sem anomalias: avaliando falsos positivos	24
2.4.3	Sistema malha aberta com anomalias: avaliando indisponibilidade	24
<b>2.5</b>	<b>Parâmetros do sistema</b>	<b>25</b>
2.5.1	Parâmetros básicos do sistema	25
2.5.2	Parâmetros do observador	25
<b>2.6</b>	<b>Do sistema ao modelo</b>	<b>25</b>
2.6.1	Modelando suposições de trabalho	26
2.6.2	Resumo	26
<b>3</b>	<b>UM MODELO INTEGRADO</b>	<b>27</b>
<b>3.1</b>	<b>Modelo</b>	<b>27</b>
3.1.1	Sistema aberto versus sistema fechado	27
3.1.2	Estado do sistema	27
3.1.3	Transições	29
3.1.4	Processos estocásticos e simulador	30
<b>3.2</b>	<b>Análise do tempo de permanência em estado anômalo e número de rejuvenescimentos aplicados</b>	<b>32</b>
3.2.1	Tempo de permanência em estado anômalo	32
3.2.2	Relação entre aplicação do rejuvenescimento e profundidade do <i>bucket</i>	35

<b>3.3</b>	<b>Detectando pontos de mudança: tempo de detecção e falsos positivos</b>	<b>37</b>
3.3.1	Trabalhos relacionados: CUSUM para detecção de pontos de mudança em fontes MMPP . . . . .	37
3.3.2	Caracterizando pontos de mudança e falso positivos . . . . .	38
3.3.3	Resultados e análise . . . . .	38
<b>3.4</b>	<b>O papel da informação na tomada de decisão</b> . . . . .	<b>40</b>
<b>4</b>	<b>MODELOS COMPARTIMENTALIZADOS</b> . . . . .	<b>44</b>
<b>4.1</b>	<b>Configuração da linha de base sem anomalias: modelando a taxa de falsos alarmes considerando o rejuvenescimento</b> . . . . .	<b>44</b>
4.1.1	Processos de nascimento-morte com catástrofes são fundamentais para estudar análise de desempenho sequencial com rejuvenescimento . . . . .	44
4.1.2	Exemplo: um modelo de três estados correspondente a um <i>bucket</i> de profundidade um . . . . .	45
4.1.3	Caso de tempo discreto . . . . .	47
4.1.3.1	Voltando ao exemplo simples . . . . .	47
4.1.3.2	Avaliação numérica e aproximação geométrica . . . . .	48
4.1.3.3	Comparação entre rejuvenescimento aleatório e rejuvenescimento oriundo do <i>bucket</i> . . . . .	52
<b>4.2</b>	<b>Contabilizando anomalias</b> . . . . .	<b>55</b>
4.2.1	Descrição do modelo . . . . .	55
4.2.2	Solução do modelo . . . . .	57
<b>4.3</b>	<b>Discussão</b> . . . . .	<b>58</b>
<b>5</b>	<b>TRABALHOS RELACIONADOS</b> . . . . .	<b>61</b>
<b>5.1</b>	<b>Envelhecimento e rejuvenescimento</b> . . . . .	<b>61</b>
5.1.1	Relação entre envelhecimento e esgotamento de recursos . . . . .	61
5.1.2	Esgotamento de recursos em sistemas críticos . . . . .	62
5.1.3	Rejuvenescimento . . . . .	62
<b>5.2</b>	<b>Processos de nascimento e morte</b> . . . . .	<b>64</b>
5.2.1	Processos de nascimento e morte clássicos . . . . .	64
5.2.2	Processos de nascimento e morte com catástrofes . . . . .	65
<b>5.3</b>	<b>Monitoramento</b> . . . . .	<b>65</b>
5.3.1	Análise de desempenho sequencial . . . . .	65
5.3.2	O algoritmo de <i>bucket</i> . . . . .	66
5.3.3	Aplicações do algoritmo de <i>bucket</i> . . . . .	66
5.3.4	CUSUM . . . . .	67
5.3.5	Controle de processos . . . . .	68
<b>6</b>	<b>CONCLUSÃO</b> . . . . .	<b>69</b>
<b>6.1</b>	<b>Trabalhos futuros</b> . . . . .	<b>70</b>

6.1.1	Validação realista . . . . .	70
6.1.2	Rejuvenescimento e trabalho cooperativo em rede . . . . .	70
	<b>REFERÊNCIAS . . . . .</b>	<b>71</b>
	<b>APÊNDICES . . . . .</b>	<b>76</b>
	<b>APÊNDICE A – PROCESSOS DE NASCIMENTO E MORTE . . . . .</b>	<b>77</b>
	<b>APÊNDICE B – SIMULADOR DE EVENTOS DISCRETOS . . . . .</b>	<b>78</b>
B.1	Fundamentos . . . . .	78
B.2	Métricas de interesse . . . . .	79
	<b>APÊNDICE C – AGREGAÇÃO E DESAGREGAÇÃO DE FLUXOS POISSON . . . . .</b>	<b>80</b>

# 1 INTRODUÇÃO

Análise de desempenho sequencial (*sequential performance analysis*) visa avaliar indicadores de desempenho de forma online. Um processo é monitorado, e tal processo é interrompido conforme o desempenho degrada, sendo o desempenho medido, por exemplo, pelo tempo de resposta do usuário ou taxa de transferência. Tais métricas passam por variações dinâmicas em função da carga, envelhecimento e ataques, notando que a carga pode ser impactada pelo envelhecimento ou ataques.

A análise sequencial de desempenho visa rastrear o desempenho do sistema para acionar alarmes quando o desempenho atinge níveis que exigem algum tipo de intervenção rigorosa. Até que os alarmes não sejam acionados, o sistema pode se recuperar naturalmente da degradação transitória de desempenho ou pode estar sujeito a rejuvenescimento conforme agendado pelos administradores. Existem vários algoritmos de análise de desempenho sequencial, incluindo CUSUM e suas variações (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016), e o algoritmo de *bucket* (*bucket algorithm*) (GONÇALVES et al., 2020). Um dos compromissos (*tradeoffs*) fundamentais envolvidos nesses algoritmos pondera o tempo de detecção em relação à taxa de falso positivos. Tal compromisso foi estudado, por exemplo, em (GONÇALVES et al., 2020), considerando, sistemas que envelhecem e se recuperam naturalmente, entretanto sem considerar o rejuvenescimento.

Neste trabalho, o objetivo é estender (GONÇALVES et al., 2020) para analisar a interação entre recuperação natural e rejuvenescimento sob análise de desempenho sequencial. Ao longo deste trabalho pretendemos responder às seguintes questões:

1. na ausência de anomalias, ou seja, sob a carga de trabalho de base (*baseline workload*), quanto tempo leva para um sistema executando análise de desempenho sequencial, sujeito a rejuvenescimento aleatório, acionar um alarme falso?
2. na presença de anomalias, qual é a interação entre envelhecimento, recuperação natural e rejuvenescimento, impactando métricas como tempo de inatividade do sistema e tempo para alarmes?
3. como o sistema se comporta em um cenário que a aplicação do rejuvenescimento pode beneficiar-se de informação estatística sobre quando o sistema encontra-se, com maior chance, no estado anômalo?

Ao responder à primeira e à terceira pergunta, aproveitamos a literatura sobre as filas  $M/M/1$  (KUMAR; ARIVUDAINAMBI, 2000) e  $M/M/1/K$  (BÖHM, 2008) com catástrofes para estudar o impacto conjunto da recuperação natural e rejuvenescimento

Tabela 1 – Contribuições

trabalhos anteriores	recuperação natural	rejuvenescimento	controle malha fechada
CUSUM/algoritmo de <i>bucket</i>	(GONÇALVES et al., 2020)		
envelhecimento rejuvenescimento sistemas de rede		(GROTTKE et al., 2016)	
Controles típicos de equipamentos e processos industriais			(CAMPOS; TEIXEIRA, 2010)
Este trabalho	✓	✓	✓

em métricas de interesse para análise de desempenho sequencial. Ao responder à segunda pergunta, estendemos os modelos clássicos de envelhecimento e rejuvenescimento (HUANG et al., 1995) para explicar a recuperação natural (ver Tabela 1).

## 1.1 Contribuições

Apresentamos três principais contribuições. Primeiro, mostramos maneiras eficientes de calcular o tempo médio até um alarme falso e indicamos as condições sob as quais uma aproximação geométrica para o tempo até um alarme falso é precisa. Em segundo lugar, introduzimos um modelo de Markov para explicar o envelhecimento, recuperação natural e rejuvenescimento, e mostramos como diferentes parâmetros afetam a indisponibilidade do sistema. Em terceiro, simulamos uma cadeia de Markov com três dimensões (três variáveis de estados), englobando um processo MMPP, o algoritmo de *bucket* e uma fila gargalo  $M/M/1/K$ . As chegadas e as retiradas dos *tokens* do *bucket* seguem um processo de Poisson, bem como a geração do sinal de rejuvenescimento do sistema, aplicando assim, um sistema de controle de malha fechada.

Em (MIRANDA et al., 2022) demos um primeiro passo no sentido de responder às perguntas desta dissertação. No Capítulo 3 unificamos o processo de chegada de *jobs* ao sistema MMPP, englobando a análise sequencial de desempenho e o algoritmo de *bucket*, caracterizando uma fila gargalo como uma  $M/M/1/K$ . Esta análise dá origem a um sistema de controle de malha fechada onde a aplicação do rejuvenescimento ao sistema é restrita a momentos específicos. No Capítulo 4 fazemos uma retrospectiva dos resultados apresentados em (MIRANDA et al., 2022).

## 1.2 Artefatos produzidos

A pesquisa apresentada nesta dissertação produziu dois artefatos:

- artigo (MIRANDA et al., 2022), apresentado em *The 33rd IEEE International Symposium on Software Reliability Engineering, WOSAR 2022: 14TH International Workshop on Software Aging and Rejuvenation*.
- repositório no Github, contendo os resultados completos da pesquisa, bem como o simulador utilizado para gerar os resultados, que podem ser acessados em <https://github.com/lleomiranda/UFRJ>. Utilizamos a linguagem de programação Python versão 3.9 para desenvolver o simulador e o *Mathematica* para gerar os resultados analíticos dos passos até o estouro do *bucket* na abordagem do rejuvenescimento aleatório.

## 1.3 Roteiro

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2 descrevemos o sistema sendo considerado no tocante às abordagens utilizadas na pesquisa, rejuvenescimento aleatório e rejuvenescimento oriundo do *bucket*. No Capítulo 3 apresentamos o novo modelo integrado de malha fechada, onde a saída gerada pelo sistema sinaliza o rejuvenescimento e restabelece as condições iniciais do sistema, e o Capítulo 4 faz uma retrospectiva dos resultados apresentados em (MIRANDA et al., 2022), via modelos compartimentalizados. No Capítulo 5 apresentamos trabalhos relacionados e que ajudaram a nortear os caminhos desta pesquisa como, envelhecimento/rejuvenescimento, processos de nascimento/morte e monitoramento. Finalmente, no Capítulo 6 apresentamos conclusões e trabalhos futuros.

## 2 DESCRIÇÃO DO SISTEMA

Neste capítulo ilustramos um sistema simples no qual as contribuições desta dissertação são aplicáveis.

### 2.1 Ingredientes básicos

#### 2.1.1 Fonte de tarefas e anomalias

Consideramos um fluxo de tarefas que chegam ao sistema, onde as chegadas são caracterizadas por um processo de Poisson modulado por Markov (MMPP) (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016; ACHARYA; VILLARREAL-RODRÍGUEZ et al., 2013; SINGH; ACHARYA, 2022). Como já mencionado em (ANDRÉOLETTI et al., 2022) os processos de nascimento-morte são processos estocásticos usados para modelar a dinâmica populacional. Estes processos são uma classe flexível de cadeias de Markov de tempo contínuo que, em nosso caso, irá modelar o fluxo de chegada dos eventos ao sistema, onde a chegada de um evento ao sistema (nascimento) seguirá uma taxa  $\lambda_G$  ou  $\lambda_A$ , dependendo do fato de o sistema estar no estado *Good* ou *Anomalous*, e a partida de um evento do sistema (morte) seguirá uma taxa  $\mu$  (ou  $\mu_S$ , dependendo do contexto). O sistema resultante, juntamente com seus parâmetros, é ilustrado na Figura 1.

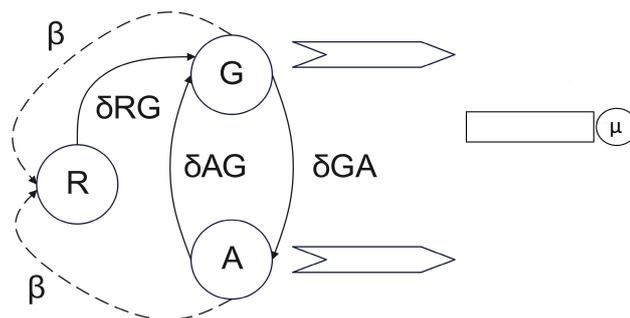


Figura 1 – Processo MMPP alimentando fila gargalo

#### 2.1.2 Rejuvenescimento

Neste nosso modelo, incorporamos o conceito de catástrofe que representa declínios súbitos na população (CAIRNS; POLLETT, 2004). Desta forma, a catástrofe representa um

rejuvenescimento do sistema, onde o mesmo retornará ao seu estado inicial e a fila de eventos será reinicializada. A catástrofe seguirá uma taxa  $\beta$ . Nosso objetivo é detectar pontos de mudança no tráfego MMPP, ou seja, visamos detectar quando e se a intensidade de chegada mudou. A única quantidade observada é o estado da fila, que serve como um proxy para 1) o atraso percebido pelo usuário em um sistema aberto (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016) ou 2) taxa de transferência de uma fila finita e/ou de um sistema fechado (MARIN et al., 2022). O principal desafio está relacionado ao fato de que a fila pode crescer devido à degradação transitória (que se resolverá naturalmente) ou devido a uma anomalia (que supomos corresponder a uma mudança no estado do MMPP). A configuração é semelhante à considerada em (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016), exceto que consideramos que o sistema está sujeito a rejuvenescimento, que no contexto desta abordagem, ocorre em momentos aleatórios não relacionados ao estado atual do sistema.

### 2.1.3 Algoritmo de *bucket*

Ao longo desta dissertação, utilizaremos como modelo de análise de desempenho sequencial o algoritmo de *bucket*, onde *tokens* são acumulados em *buckets* quando o sistema se degrada e removidos quando o sistema se recupera naturalmente. Se o número de *tokens* no sistema atingir um limite, um alarme será acionado.

Introduzimos a figura do observador que pode adicionar ou remover um *token* do *bucket* a cada observação. O algoritmo de *bucket* gera possíveis sinais de rejuvenescimento do sistema, como ilustrado na Figura 2.

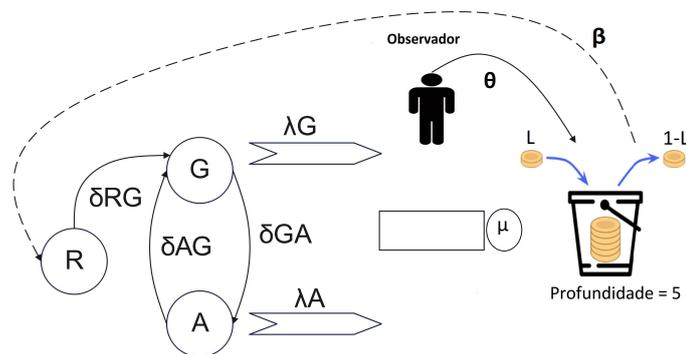


Figura 2 – Processo MMPP, *bucket* e  $M/M/1$ , correspondendo a um sistema com três dimensões (três variáveis de estado)

### 2.1.3.1 Amostras de *throughput*

Cada vez que uma nova amostra é coletada do sistema, ela é comparada com um limiar. A partir de agora, assumimos que as amostras correspondem a medidas de *throughput* (instantâneas), e que valores pequenos podem ser devidos a anomalias. Do ponto de vista analítico, a diminuição na taxa de transferência de pacotes na presença de pacotes ruins pode ser devido 1) ao fato de que no estado ruim temos  $\lambda_A > \mu_S$  em um sistema aberto como na Figura 1 (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016); 2) fila finita, ou seja,  $M/M/1/K$  em oposição a  $M/M/1$  na Figura 1; ou 3) um sistema fechado (MARIN et al., 2022).

No sistema proposto existem *buckets*  $B$  com profundidade máxima  $D$  cada. O *bucket* atual é referido como  $b$ , e a profundidade atual do *bucket* atual, ou seja, o número de *tokens* nele, é  $d$ . Para cada *bucket* existe um limite correspondente, cujo valor é uma função de  $b$ . Se uma amostra for menor que o limite atual, um *token* será adicionado ao *bucket* atual (taxa de transferência pequena). Caso contrário, um *token* é removido (taxa de transferência normal). Isto segue a lógica proposta por (AVRITZER, 2010).

Se o *bucket* atual estourar, o algoritmo define o próximo *bucket* como o *bucket* atual. Da mesma forma, se o *bucket* atual for insuficiente, ou seja, se o número de *tokens* se tornar negativo, o algoritmo define o *bucket* anterior como o *bucket* atual ou permanece inalterado se  $b = 1$  e  $d = 0$ . Quando  $b = 1$  e  $d = 0$ , o sistema recuperou seu estado original. Em contraste, no extremo, quando todos os *buckets* transbordam,  $b = B$  e  $d = D + 1$ , um alarme é disparado sinalizando uma anomalia.

### 2.1.3.2 Relação entre o estado do sistema e o estado do *bucket*

Observe que na Figura 1 o estado do sistema é caracterizado pelo estado da fila, ou seja, um número inteiro que representa o número de pacotes na fila e o estado do processo MMPP. O estado do *bucket*, por outro lado, é simplesmente o *bucket* atual e o número de *tokens* contidos nele. O estado do sistema e o estado do *bucket* estão intimamente relacionados entre si, porém não estão ligados diretamente, pois o processo de adicionar/remover um *token* no *bucket*, bem como o rejuvenescimento não são markovianos: o número de *tokens*  $d$  deve refletir tamanhos de fila maiores e/ou uma transição da origem para o estado anômalo  $A$ . Caso a fonte ainda esteja no estado bom  $G$ , um aumento em  $d$  corresponde a uma maior probabilidade de emissão de um falso positivo.

## 2.2 Informação disponível para tomada de decisão

Consideramos três níveis de informação disponível para tomada de decisão:

- **ausência de informação:** neste cenário não há relação direta do estouro do *bucket*

e a ação de rejuvenescer o sistema, bem como a ação de chegada e retirada do *bucket* não é regulada por uma variável aleatória exponencial. O sistema está sujeito a rejuvenescimento, que irá ocorrer em momentos aleatórios não relacionados ao estado atual do sistema.

- **informação parcial:** neste caso, o administrador do sistema pode, em certas situações, favorecer o rejuvenescimento tendo informação parcial sobre o estado da fonte, e.g., usando o algoritmo de *bucket*.
- **informação completa:** neste caso, o administrador do sistema tem alto grau de confiança sobre o estado da fonte de anomalias, e o sistema é rejuvenescido quando estritamente necessário.

No primeiro cenário, a ação de inserção e remoção de *tokens* no *bucket* e a ação de rejuvenescimento são ortogonais e independentes. Na segunda e terceira abordagens, implementamos uma cadeia de Markov com três variáveis de estado, a saber:  $s$  estado da fonte de anomalias representando o estado bom ou anômalo que o sistema possa estar,  $j$  estado da fila do sistema, o número de *jobs* no sistema e  $b$  algoritmo de detecção de anomalias representando a quantidade de *tokens* no *bucket*. Tentamos unificar todo o processo de forma que o sinal de rejuvenescimento tenha relação direta com o estado do sistema.

## 2.3 Malha aberta versus malha fechada

Cabe ressaltar algumas mudanças fundamentais entre as duas abordagens, apresentadas na Figura 1 e Figura 2.

Na Figura 1 o *bucket* deve refletir indiretamente o estado do sistema. Entretanto, o rejuvenescimento ocorre em momentos aleatórios, sem relação com o número de *tokens* do *bucket*. Consequentemente, o sistema pode ser rejuvenescido seguidamente e sem que esteja em um momento crítico. É possível que o sistema seja rejuvenescido quando ainda poderia se recuperar naturalmente de alguma anomalia. Este fato pode levar o sistema a uma indisponibilidade excessiva, por conta do número de rejuvenescimentos precoces. Nesta abordagem, o *bucket* não possui uma ligação direta com o processo MMPP, que será explicado em detalhes posteriormente.

Na Figura 2, simulamos uma cadeia de Markov de 3 dimensões (estado da fonte de anomalias, algoritmo de detecção de anomalias, estado da fila do sistema). O algoritmo de detecção de anomalias irá refletir diretamente o estado do sistema, espelhando as perturbações no transcorrer do seu funcionamento. Caso o número de perturbações ultrapasse o limite estabelecido pelos administradores do sistema, o algoritmo de detecção de anomalias sinalizará uma anormalidade e o rejuvenescimento ocorrerá, restabelecendo as

condições iniciais do sistema. Desta forma, simulamos um controle de malha fechada, onde anormalidades na saída do sistema geram correções no processo.

## 2.4 Análise do sistema em malha aberta e fechada

Analisamos os processos descritos nas Figuras 1 e 2.

### 2.4.1 Sistema em malha fechada

A Figura 2 ilustra o sistema malha fechada. Procuramos caracterizar todo o processo como sendo markoviano, e delegamos detalhes para o Capítulo 3.

### 2.4.2 Sistema malha aberta sem anomalias: avaliando falsos positivos

Na Seção 4.1 consideramos o caso em que não há anomalias (por exemplo, o estado  $A$  na Figura 1 nunca é alcançado). Neste caso, a questão principal diz respeito ao tempo médio até um falso positivo. O sistema transita entre os estados  $R$  e  $G$ , mas um alarme ainda pode ser acionado quando a fila cresce além de um determinado valor, fato intimamente relacionado com (ZHAO, 2010). Ao longo do tempo o sistema passa por degradações transientes (que não são anomalias propriamente ditas) que passo a passo vão comprometendo seu desempenho. Em contrapartida, o sistema consegue se recuperar de algumas destas degradações transientes de forma natural. O problema ocorre quando o sistema não consegue se recuperar de forma natural do volume de degradações transientes existentes. Este acúmulo de degradações leva a um esgotamento dos recursos do sistema, o que faz com que o algoritmo de análise de desempenho sequencial acione um alerta.

### 2.4.3 Sistema malha aberta com anomalias: avaliando indisponibilidade

Então, na Seção 4.2, consideramos o caso em que as anomalias estão presentes (todos os estados do sistema são alcançáveis), neste caso, pretendemos estudar a indisponibilidade do sistema devido a anomalias e rejuvenescimento, assumindo que ambos são causas de downtime do sistema. Na Figura 1, os estados  $R$  e  $A$  levam à indisponibilidade e o estado  $G$  corresponde a um sistema disponível. Claramente, aumentar a taxa de rejuvenescimento  $\beta$  faz com que o sistema passe menos tempo no estado anômalo, mas ao custo de uma potencial indisponibilidade durante o rejuvenescimento. Um de nossos objetivos é estudar os *tradeoffs* envolvidos no ajuste do rejuvenescimento, à luz da recuperação natural (capturada através de  $\delta_{AG}$  e através da dinâmica do tamanho da fila enquanto a fonte está no estado  $G$ ) e anomalias (capturadas por  $\delta_{GA}$ ). Em um segundo momento, fazemos um comparativo com o proposto na Figura 2, no sentido de entendermos os impactos das duas abordagens de rejuvenescimento, a primeira quando o rejuvenescimento ocorre em momentos aleatórios

não relacionados ao estado atual do sistema e a segunda quando o rejuvenescimento ocorre no momento que o *bucket* ultrapassa o seu limite.

## 2.5 Parâmetros do sistema

### 2.5.1 Parâmetros básicos do sistema

Um passo importante e fundamental é definir as características do nosso sistema e os estados do processo MMPP. No contexto deste trabalho, iremos trabalhar com 3 estados possíveis para o processo MMPP que são  $G$ ,  $A$  e  $R$ , correspondendo aos estados, bom, anômalo e rejuvenescimento, respectivamente (ver Figura 1). No estado bom (resp., anômalo), a taxa de chegada para a fila do servidor central é  $\lambda_G$  (resp.,  $\lambda_A$ ). Assumimos  $\lambda_A > \lambda_G$ , onde  $\lambda_A = \lambda_G + \lambda_B$  onde  $\lambda_B$  é a taxa de chegada de *background packets* que consomem recursos do sistema. No contexto de anomalias devido a ataques, os *background packets* correspondem a pacotes ruins injetados pelo invasor, por exemplo, para causar um ataque de negação de servidor (DoS). Seja  $\beta$  a taxa de rejuvenescimento, denotamos por  $1/\delta_{RG}$  o tempo médio de rejuvenescimento, e por  $\delta_{AG}$  e  $\delta_{GA}$  as taxas de transição do estado  $A$  a  $G$  e  $G$  a  $R$ , respectivamente. Finalmente,  $\mu$  é a taxa de serviço da fila.

### 2.5.2 Parâmetros do observador

No caso da Figura 2, introduzimos o conceito do observador que com uma taxa  $\theta$  fará a inserção e remoção de *tokens* no *bucket*, seguindo uma variável aleatória exponencial e o sinal de rejuvenescimento será disparado quando o sistema estiver em situação crítica (quando o *bucket* estourar seu limite). Desta forma, pretendemos caracterizar todo o processo como markoviano, dando origem a um controle de malha fechada, onde o sinal enviado pelo *bucket* gera uma ação de reconfiguração do sistema, no sentido de restabelecer suas condições normais novamente (CAMPOS; TEIXEIRA, 2010).

## 2.6 Do sistema ao modelo

A seguir, com base na Figura 1, indicamos como parametrizar um modelo de envelhecimento e rejuvenescimento com base nas características do sistema.

No Capítulo 3 tratamos da cadeia de Markov com três variáveis de estado e seus resultados. Na Seção 3.1 descrevemos o modelo de cadeia de Markov com 3 variáveis de estado, na Seção 3.2 analisamos o tempo de permanência no estado anômalo e a relação de número de rejuvenescimentos aplicados e profundidade do *bucket*. Já na Seção 3.3 implementamos um modelo de detecção de transição do estado da fonte de anomalia com base no preconizado em (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016). Na

Seção 3.4 analisamos a situação hipotética em que o administrador do sistema tenha conhecimento geral do estado do sistema. Em especial, o administrador sabe se o sistema está no estado bom ou no estado anômalo.

Na Seção 4.1, quando  $\delta_{GA} = \delta_{AG} = \lambda_A = 0$ , e  $\delta_{RG} = \infty$ , o modelo compreende três parâmetros: a taxa na qual os *tokens* são adicionados ao *bucket* do chamado algoritmo de *bucket*, a taxa na qual os *tokens* são removidos do *bucket* e a taxa de rejuvenescimento. Estes três parâmetros são  $\lambda$ ,  $\mu$  e  $\xi$ . De fato, os dois primeiros crescem em função de  $\lambda_G$  e  $\mu_s$ , respectivamente. A terceira é uma função de  $\beta$ . Nosso objetivo é calcular o tempo até que o algoritmo de *bucket* transborde, na ausência de anomalias, o que corresponderá a um alarme falso.

Na Seção 4.2 consideramos o caso em que as anomalias estão presentes. Neste caso, podemos novamente parametrizar o modelo a partir de sistemas. Nesse cenário, também aproveitamos  $\lambda_A$  e  $\delta_{AG}$  para definir os parâmetros do modelo e contabilizar a fração de tempo em que o sistema está nos estados  $R$  e  $A$  para caracterizar a indisponibilidade.

### 2.6.1 Modelando suposições de trabalho

Como em qualquer exercício de modelagem, temos que equilibrar a complexidade do modelo e a capacidade expressiva. Em nossos modelos de tempo contínuo, fazemos as seguintes suposições simplificadoras, para o caso da Figura 1: 1) os tempos entre eventos são todos distribuídos exponencialmente, independentemente do modelo usado para caracterizar o sistema, por exemplo, MMPP apresentado na Figura 1; 2) o rejuvenescimento pode ocorrer a partir de qualquer estado do sistema e, exceto quando indicado de outra forma, ocorre com uma taxa constante independente do estado do sistema.

No caso da Figura 2, 1) os tempos entre todos os eventos são gerados por variáveis aleatórias exponenciais; 2) o rejuvenescimento ocorre em decorrência do estouro do *bucket*, que sinaliza uma anormalidade e dispara o rejuvenescimento, levando o sistema para sua condição inicial, caracterizando, assim, um sistema de controle de malha fechada (CAMPOS; TEIXEIRA, 2010), onde com base em parâmetros definidos uma situação indesejada é remediada.

### 2.6.2 Resumo

Nesta seção indicamos como os parâmetros dos modelos considerados na pesquisa se relacionam com um sistema simples. A discussão possibilita ilustrar como o modelo é aplicável em um cenário simples. Deixamos uma metodologia para produzir uma relação funcional direta entre o sistema e os parâmetros do modelo, em cenários mais realistas, como assunto para trabalhos futuros.

## 3 UM MODELO INTEGRADO

Neste capítulo, apresentamos uma visão integrada sobre processos de envelhecimento e de detecção de envelhecimento para fins de rejuvenescimento.

### 3.1 Modelo

A seguir, introduzimos o modelo integrado, ilustrado na Figura 3. Na Figura 3(a) apresentamos a versão do modelo levando um observador que não afeta o sistema (malha aberta) e na Figura 3(b) consideramos um sistema no qual o observador afeta o sistema (malha fechada).

#### 3.1.1 Sistema aberto versus sistema fechado

Se o sistema for implementado sem rejuvenescimento, ou se o rejuvenescimento for gerado por fatores exógenos, sem ser causado, por exemplo, pelo algoritmo de *bucket*, temos um sistema de malha aberta.

Quando o resultado do sistema de detecção de anomalias gera um sinal de alerta que causa uma perturbação na fonte de anomalias, temos um controle de malha fechada. Isso ocorre, por exemplo, quando o tempo médio de espera experimentado pelas tarefas está acima do esperado. O administrador usa-se então de medidas para regularizar o funcionamento do sistema, por exemplo, forçando-o a voltar ao seu estado inicial, ou seja, fazendo rejuvenescimento.

#### 3.1.2 Estado do sistema

Caracterizamos o estado da fonte de anomalias, o estado do algoritmo de detecção de anomalias e o estado de uma fila (gargalo) do sistema.

- **estado da fonte de anomalias:** caracterizamos o estado da fonte de anomalias por meio de um processo *on-off* que pode ser generalizado para um *Markov Modulated Poisson Process*, também conhecido como MMPP.
- **algoritmo de detecção de anomalias:** caracterizamos o estado do *bucket* como um processo de nascimento e morte.
- **estado de uma fila (gargalo) do sistema:** finalmente, caracterizamos o estado da fila gargalo como uma fila  $M/M/1/K$ .

Usamos uma cadeia de Markov, com três dimensões, correspondendo aos três pontos acima:

- **nível de envelhecimento** ( $s \in \{0, 1\}$ ):<sup>1</sup> uma das variáveis de estado representa a condição do sistema, ou seja, o seu nível de envelhecimento, que pode ser, por exemplo, bom (“*Good*”) ou anômalo (“*Anomalous*”). Essa variável, em princípio, não está acessível para o administrador do sistema. O administrador do sistema precisa usar mecanismos especiais de detecção de envelhecimento para tomar ações;
- **estado do “*bucket*”** ( $b \in \{0, 1, \dots, L_2\}$ ): outra variável de estado caracteriza o mecanismo de detecção de envelhecimento (número de “*tokens*” no “*bucket algorithm*”), tal que quanto mais “*tokens*”, maior a chance do sistema estar envelhecido, e ser necessário soar um alarme. Essa variável, em princípio, está acessível para o administrador do sistema, e
- **estado da fila** ( $j \in \{0, 1, \dots, K\}$ ): uma variável de estado caracteriza o número de (“*jobs*”) na fila (“*buffer*”) do sistema, que pode variar rapidamente, sendo afetado pelo nível de envelhecimento do sistema e impacta o estado do “*bucket*”. Ou seja, o estado da fila é influenciado pelo nível de envelhecimento (quanto mais envelhecido o sistema, maior a sua fila) e influencia o estado do “*bucket*” (quanto mais elementos na fila, mais “*tokens*” no “*bucket*”, notando que enquanto o estado da fila muda rapidamente, o estado do “*bucket*” varia de forma mais suave para evitar muitos falso positivos).

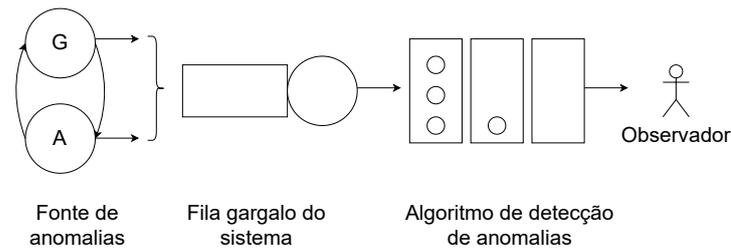
**Cadeia de Markov com 3 variáveis de estado.** Iremos simular uma cadeia de Markov com 3 variáveis de estado, e para isto, consideramos um processo estocástico  $\{\tau_i\}_{i \geq 0}$ , onde  $\tau_i = t_i - t_{i-1}$  é o comprimento do  $i$ -ésimo intervalo entre eventos. Temos então 3 processos estocásticos subjacentes, correspondes às três dimensões apontadas acima:

- $\{s_i\}_{i \geq 0}$  representa os estados  $G$  e  $A$ , que para efeitos desta representação serão referidos de forma intercambiável como 0 e 1, respectivamente,
- $\{b_i\}$  representa o número de *tokens* no *bucket* e
- $\{j_i\}$  representa o número de *jobs* no sistema.

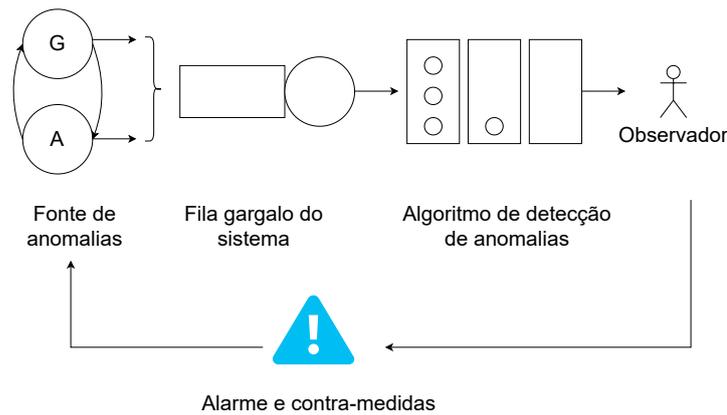
Dois parâmetros-chave são adotados:

1. *Transient Loss Probability* (TLP), explicado na 4.1.3.2, que é a probabilidade de adicionarmos um *token* ao *bucket*, e será representado por  $L_1$ , e

<sup>1</sup>  $G$  corresponde ao valor 0 e  $A$  corresponde ao valor 1.



(a) sistema de malha aberta



(b) sistema de malha fechada

Figura 3 – Diagrama do modelo integrado. Cada elemento no diagrama acima corresponde a uma dimensão do modelo: fonte de anomalias, fila gargalo do sistema e estado do algoritmo de detecção de anomalias.

2. o limite do *bucket*, que dispara o sinal de rejuvenescimento do sistema, sendo representado por  $L_2$ . Desta forma, quando  $b_i > L_2$  o sistema passa por rejuvenescimento, voltando ao seu estado inicial (3.10) simulando um controle de malha fechada (CAMPOS; TEIXEIRA, 2010).
3. No caso do controle em malha aberta, as perturbações ocorridas no decorrer do funcionamento do sistema não iriam interferir em uma redefinição dos parâmetros do sistema. No nosso caso, mesmo que o limite do *bucket* seja atingido, o rejuvenescimento não seria acionado, pois as perturbações tanto de ordem externa como internas não são consideradas, é esperada uma saída de acordo com parâmetros definidos na entrada, é assumido um funcionamento regular do sistema.

### 3.1.3 Transições

A Tabela 2 ilustra a transição de estados da cadeia de Markov proposta neste trabalho. Note que na tabela consideramos o caso especial de uma fila  $M/M/1/2$ , com espaço para armazenar no máximo 2 *jobs*.

As taxas que caracterizam o sistema são representadas por letras gregas.

Tabela 2 – Cadeia de Markov com 3 variáveis de estado: (estado da fonte de anomalias, estado da fila gargalo, estado do algoritmo de *bucket*). Assumimos uma fila gargalo  $M/M/1/2$

$s=0,$ $j=0,$ $b=0$	$(0, 0, 0) \xrightarrow{\lambda_G} (0, 1, 0)$ $(0, 0, 0) \xrightarrow{\theta \cdot L_1} (0, 0, 1)$ $(0, 0, 0) \xrightarrow{\delta_{GA}} (1, 0, 0)$	$s=1,$ $j=0,$ $b=0$	$(1, 0, 0) \xrightarrow{\lambda_A} (1, 1, 0)$ $(1, 0, 0) \xrightarrow{\theta \cdot L_1} (1, 0, 1)$ $(1, 0, 0) \xrightarrow{\delta_{AG}} (0, 0, 0)$
$s=0,$ $j>0,$ $b=0$	$(0, 1, 0) \xrightarrow{\lambda_G} (0, 2, 0)$ $(0, 1, 0) \xrightarrow{\mu} (0, 0, 0)$ $(0, 1, 0) \xrightarrow{\theta \cdot L_1} (0, 1, 1)$ $(0, 1, 0) \xrightarrow{\delta_{GA}} (1, 1, 0)$	$s=1,$ $j>0,$ $b=0$	$(1, 1, 0) \xrightarrow{\lambda_A} (1, 2, 0)$ $(1, 1, 0) \xrightarrow{\mu} (1, 0, 0)$ $(1, 1, 0) \xrightarrow{\theta \cdot L_1} (1, 1, 1)$ $(1, 1, 0) \xrightarrow{\delta_{AG}} (0, 1, 0)$
$s=0,$ $j=0,$ $b>0$	$(0, 0, 1) \xrightarrow{\lambda_G} (0, 1, 1)$ $(0, 0, 1) \xrightarrow{\theta \cdot L_1} (0, 0, 2) \xrightarrow{\delta_{RG}} (0, 0, 0)$ $(0, 0, 1) \xrightarrow{\theta \cdot (1-L_1)} (0, 0, 0)$ $(0, 0, 1) \xrightarrow{\delta_{GA}} (1, 0, 1)$	$s=1,$ $j=0,$ $b>0$	$(1, 0, 1) \xrightarrow{\lambda_A} (0, 1, 1)$ $(1, 0, 1) \xrightarrow{\theta \cdot L_1} (0, 0, 2) \xrightarrow{\delta_{RG}} (0, 0, 0)$ $(1, 0, 1) \xrightarrow{\theta \cdot (1-L_1)} (0, 0, 0)$ $(1, 0, 1) \xrightarrow{\delta_{GA}} (1, 0, 1)$
$s=0,$ $j>0,$ $b>0$	$(0, 1, 1) \xrightarrow{\lambda_G} (0, 2, 1)$ $(0, 1, 1) \xrightarrow{\theta \cdot L_1} (0, 1, 2) \xrightarrow{\delta_{RG}} (0, 0, 0)$ $(0, 1, 1) \xrightarrow{\mu} (0, 0, 1)$ $(0, 1, 1) \xrightarrow{\theta \cdot (1-L_1)} (0, 1, 0)$ $(0, 1, 1) \xrightarrow{\delta_{AG}} (1, 1, 1)$	$s=1,$ $j>0,$ $b>0$	$(1, 1, 1) \xrightarrow{\lambda_A} (1, 2, 1)$ $(1, 1, 1) \xrightarrow{\theta \cdot L_1} (1, 1, 2) \xrightarrow{\delta_{RG}} (0, 0, 0)$ $(1, 1, 1) \xrightarrow{\mu} (1, 0, 1)$ $(1, 1, 1) \xrightarrow{\theta \cdot (1-L_1)} (1, 1, 0)$ $(1, 1, 1) \xrightarrow{\delta_{AG}} (0, 1, 1)$

- a taxa de chegada de pacotes para a fila, quando o sistema está no estado *Good* ( $G$ ) é dada por  $\lambda_G$ . No estado  $(0,0,0)$ , por exemplo, temos uma transição com taxa  $\lambda_G$  para o estado  $(0,1,0)$ . Similarmente, quando o sistema está no estado *Anômalo* ( $A$ ) a taxa de chegada de pacotes é  $\lambda_A$
- a taxa de chegada de *tokens* para o *bucket* é dada por  $\theta$ . No estado  $(0,0,0)$ , por exemplo, temos uma transição com taxa  $\theta \cdot L_1$  para o estado  $(0,0,1)$ , correspondendo à inclusão de um *token*. Note que com taxa  $\theta \cdot (1 - L_1)$  ocorreria a remoção de um *token*, mas como o número de *tokens* não pode ser negativo ela não é representada
- a taxa de transição dos estados  $G$  para  $A$  e de  $A$  para  $G$  é dada por  $\delta_{GA}$  e  $\delta_{AG}$ , respectivamente. Estas transições, ocorrem, por exemplo, a partir dos estados  $(0,0,0)$  e  $(1,0,0)$
- a taxa de serviço de pacotes da fila é dada por  $\mu$ .

### 3.1.4 Processos estocásticos e simulador

A seguir, descrevemos o simulador de um processo MMPP, acoplado ao algoritmo *bucket* e fila gargalo  $M/M/1/K$  com envelhecimento e rejuvenescimento, implementando

controle de malha fechada. Em particular, descrevemos de forma mais detalhada a ideia básica de funcionamento da simulação proposta. Na  $i$ -ésima iteração do algoritmo, uma “disputa” é estabelecida entre as variáveis aleatórias  $V_i^A \sim \exp(\lambda_A)$ ,  $V_i^G \sim \exp(\lambda_G)$ ,  $M_i \sim \exp(\mu)$ ,  $O_i \sim \exp(\theta)$ ,  $W_i^{AG} \sim \exp(\delta_{AG})$ ,  $W_i^{GA} \sim \exp(\delta_{GA})$  simuladas. A vencedora da disputa será a de menor valor e será atribuído a  $\tau_i$  o valor da variável aleatória exponencial vencedora, ou seja, o tempo de simulação é incrementado. A simulação ocorrerá enquanto  $t_i \leq T_{\text{limite}}$ . No estado inicial, temos que

$$\tau_0 = 0, \quad t_0 = 0. \quad (3.1)$$

Em geral, para  $i > 0$ ,

$$t_i = t_{i-1} + \tau_i \quad (3.2)$$

e

$$\tau_i = \begin{cases} \min(V_i^G, M_i, O_i, W_i^{GA}) & \text{se } s_i = 0, \\ \min(V_i^A, M_i, O_i, W_i^{AG}) & \text{se } s_i = 1. \end{cases} \quad (3.3)$$

O estado de envelhecimento do sistema na  $i$ -ésima iteração é atualizado seguindo a seguinte regra:

$$s_0 = 0, \quad (3.4)$$

$$s_i = \begin{cases} 1, & \text{se } s_{i-1} = 0 \text{ e } \tau_i = W_i^{GA} \\ 0, & \text{se } s_{i-1} = 1 \text{ e } \tau_i = W_i^{AG} \\ s_{i-1}, & \text{caso contrário} \end{cases} \quad (3.5)$$

O número de *jobs* no sistema na  $i$ -ésima iteração é atualizado seguindo a seguinte regra:

$$j_0 = 0, \quad (3.6)$$

$$j_i = \begin{cases} j_{i-1} + 1, & \text{se } \tau_i = V_i^A \text{ ou } \tau_i = V_i^G \\ j_{i-1} - 1, & \text{se } \tau_i = M_i \\ j_{i-1}, & \text{caso contrário} \end{cases} \quad (3.7)$$

O número de *tokens* no *bucket* na  $i$ -ésima iteração é atualizado somente quando  $\tau_i = O_i$ , caso contrário  $b_i = b_{i-1}$ ; sempre que  $\tau_i = O_i$ , é criada uma variável  $U \sim \text{Unif}(0, 1)$  e a atualização acontece segundo a seguinte regra:

$$b_i = \begin{cases} b_{i-1} + 1 & \text{se } U \leq L \\ b_{i-1} - 1 & \text{se } U > L \text{ e } b_{i-1} > 0 \\ 0, & \text{caso contrário} \end{cases} \quad (3.8)$$

O rejuvenescimento é aplicado quando o limite do *bucket* é ultrapassado, e  $\tau_i$  é incrementado por  $W_{RG} \sim \exp(\delta_{RG})$  (3.9) que representa o tempo de transição do estado R

para o estado G. Neste período o sistema ficará inativo e, decorrido este tempo, o sistema retornará a sua condição inicial (3.10):

$$\tau_i = \tau_{i-1} + (R + W_{RG}), \quad (3.9)$$

$$s_i = 0, \quad j_i = 0, \quad b_i = 0 \quad (3.10)$$

Para análise dos algoritmos a serem apresentados a seguir, é conveniente definir um vetor  $F_P$  que armazena os instantes de falso positivos:

$$F_P = (t_i : s_i = 0 \text{ e } b_i > L_2) \quad (3.11)$$

Note que um falso positivo ocorre quando acontece um estouro do “bucket” ( $b_i > L_2$ ) mas o sistema não está num estado anômalo ( $s_i = 0$ ). Nesse caso, temos uma degradação transiente, em que a fila pode ter aumentado devido a flutuações naturais do sistema, sem que seja caracterizada uma anomalia de médio ou longo prazo.

## 3.2 Análise do tempo de permanência em estado anômalo e número de rejuvenescimentos aplicados

Nesta seção, abordamos a análise de desempenho sequencial de sistemas que envelhecem e rejuvenescem, sob a perspectiva de um controle de malha fechada (CAMPOS; TEIXEIRA, 2010) a partir do momento que estendemos (MIRANDA et al., 2022) e incluímos os processos da fonte de anomalias (MMPP) e de detecção de anomalias (*bucket*) como parte integral de um modelo que também considera o estado da fila gargalo (uma  $M/M/1/K$ ), dando origem a uma cadeia de Markov com 3 variáveis de estado, como já explicado na Seção 2.5 e ilustrado na Figura 2.

### 3.2.1 Tempo de permanência em estado anômalo

**Questão 6:** *Com a figura do observador sendo considerada no modelo integrado, englobando o processo MMPP, o algoritmo de bucket, e a fila gargalo  $M/M/1/K$ , a aplicação do rejuvenescimento pode ajudar o sistema a permanecer menos tempo no estado anômalo?*

Para responder esta pergunta, usamos como base a cadeia de Markov descrita na Tabela 2. Em um sistema estável temos  $\mu > \lambda \gg \delta$  que representam, respectivamente, a taxa de serviço, taxa de chegada de processos, e taxa de transição de estados. No nosso exemplo consideramos três estados para a fonte de anomalias:

- $G$ , estado bom, onde processos são gerados com taxa  $\lambda_G$
- $A$ , estado anômalo, onde processos são gerados com taxa  $\lambda_A$

Tabela 3 – Tabela de notação

Variável	Descrição
$\lambda_A$	taxa de chegada de processos à fila, quando sistema no estado anômalo
$\lambda_G$	taxa de chegada de processos à fila, quando sistema no estado bom
$\mu$	taxa de serviço da fila gargalo
$\delta_{GA}$	taxa de transição de $G$ para $A$ (chegada de anomalia)
$\delta_{AG}$	taxa de transição de $A$ para $G$ (recuperação natural)
$\delta_{RG}$	taxa de transição de $R$ para $G$ ( $1/\delta_{RG}$ é tempo médio de rejuvenescimento)
$\theta$	taxa de observação (observações colhidas por unidade de tempo)
$L_1$	probabilidade de observação adicionar <i>token</i> ao <i>bucket</i> (TLP)
$L_2 = D$	limiar do número de <i>tokens</i> no <i>bucket</i> antes de soar alarme ( <i>depth</i> )

- estado  $R$ , de rejuvenescimento, sendo que a taxa de saída do estado  $R$  é igual a  $\delta_{RG}$ .

Ao sair do estado  $R$  o sistema sempre vai para o estado  $G$ . Temos três possíveis transições de estados:  $GA$ ,  $AG$  e  $RG$  que ocorrem com taxas  $\delta_{GA}$ ,  $\delta_{AG}$  e  $\delta_{RG}$  (ver Tabela 3). Finalmente, recordamos que a taxa de serviço é dada por  $\mu$  e que o observador colhe amostras à taxa  $\theta$ , sendo que cada amostra pode adicionar ou remover *tokens* do *bucket*.

Para responder à pergunta chave desta seção, consideramos dois cenários:

- **cenário 1)**  $\mu > \lambda_A > \lambda_G \gg \delta_{GA} = \delta_{AG}$ , com valores respectivamente iguais a 0,16, 0,12, 0,08, 0,002, quando  $\lambda_A$  pode causar uma instabilidade ao sistema, entretanto, por ser menor que  $\mu$ , esperamos que o sistema ainda apresente uma certa resiliência frente às anomalias presentes e não entre em colapso.
- **cenário 2)**  $\lambda_A > \mu > \lambda_G \gg \delta_{GA} = \delta_{AG}$ , com valores respectivamente iguais a 0,25, 0,16, 0,08, 0,002, quando  $\lambda_A$  pode causar uma instabilidade grande ao sistema, devido ao aumento do fluxo de chegadas de *jobs*, com resiliência do sistema baixa, uma vez que temos  $\lambda_A > \mu$ . Neste cenário o sistema pode entrar em colapso caso permaneça no estado anômalo por tempo grande o suficiente. O limite do *bucket* pode ser atingido rapidamente, e a recuperação natural pode não ser eficiente.

Nosso propósito é entender o papel do observador e do rejuvenescimento, no sentido de ajudar o sistema a permanecer menos tempo no estado anômalo, onde  $\lambda_A$  pode aumentar o fluxo de chegada de *jobs* ao sistema. Neste sentido, a taxa  $\theta$  irá ser incrementada gradativamente durante o tempo da simulação, que será de 5.000. Fixamos a taxa  $\delta_{RG} = 0,1$ , de forma que o rejuvenescimento e a transição do estado  $R$  para o estado  $G$  sejam breves. Os resultados obtidos são referentes a valores médios obtidos após a execução de 500 iterações da simulação.

A seguir, analisamos o tempo de permanência no estado  $A$  (ver Tabela 4). Começamos considerando o cenário 1) e comparamos as abordagens sem a aplicação do

Tabela 4 – Estatísticas sobre número de transações atendidas e tempo nos estados  $G$  e  $A$ 

	Cenário 1: $\mu > \lambda_A$		Cenário 2: $\mu < \lambda_A$	
	Sem rejuv.	Com rejuv.	Sem rejuv.	Com rejuv.
Transações não atendidas	2,857	1,174	62,816	8,902
Utilização do sistema	58%	51%	86%	58%
Tempo no estado $G$	2.806,596	3.494,309	2.824,067	3.475,115
Tempo no estado $A$	2.500,215	1.005,812	2.609,406	1.015,735
Número de Rejuvenescimentos	0	25,972	0	26,340
Inatividade por Rejuvenescimento	0	10,36%	0	10,55%

rejuvenescimento e com aplicação do rejuvenescimento. Percebemos um tempo de permanência similar nos estados  $G$  e  $A$  no caso da não aplicação do rejuvenescimento. Entretanto, quando analisamos o caso onde aplicamos o rejuvenescimento, podemos observar que o tempo de permanência no estado  $A$  apresenta números significativamente menores em relação ao estado  $G$ . Já analisando o cenário 2), percebemos que a tendência de queda no estado anômalo, com a aplicação do rejuvenescimento, é mantida mesmo quando temos  $\lambda_A > \mu$ . Entretanto, o número médio de transações não resolvidas pelo sistema sobe consideravelmente quando temos este cenário, como podemos observar, comparando este indicador nas respectivas colunas da Tabela 4.

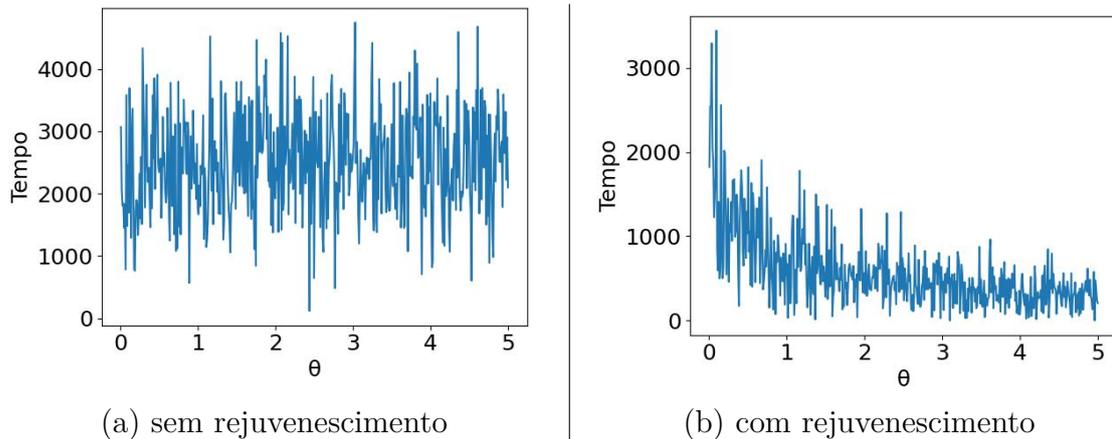


Figura 4 – Cenário 1: Tempo médio de permanência no estado anômalo  $A$ , com  $D = 12$ , variando taxa  $\theta$ , (a) sem o rejuvenescimento, e (b) com rejuvenescimento, e  $\delta_{RG} = 0,1$

A Figura 4 mostra como o tempo de permanência no estado  $A$  varia em função de  $\theta$ . A Figura 4(a) considera o cenário 1), sem a aplicação do rejuvenescimento. Constatamos que apenas o crescimento da taxa do observador  $\theta$  não contribui para uma permanência

menor no estado  $A$ , demonstrando que apenas o fato de aumentarmos a probabilidade de inserção e remoção de *tokens* no *bucket* não ajuda no cenário de evitarmos uma permanência excessiva no estado anômalo. Já a Figura 4(b) considera também o cenário 1), entretanto no caso da aplicação do rejuvenescimento. Verifica-se que, a medida que a taxa  $\theta$  aumenta, o tempo de permanência no estado  $A$  diminuiu, pois com um aumento em  $\theta$ , temos uma probabilidade maior de atingirmos o limite do *bucket* e assim dispararmos o rejuvenescimento. Isso se mostra eficiente para uma permanência menor no estado anômalo. O mesmo comportamento foi percebido nos gráficos para o cenário 2, é importante notar que taxa  $\delta_{RG}$  mais agressiva propicia uma diminuição no tempo de inatividade do sistema por conta de rejuvenescimento, entretanto este fato pode acarretar custos computacionais, que devem ser considerados na montagem de uma estratégia adequada de rejuvenescimento.

Até então trabalhamos com taxas iguais para as transições de estados  $GA$  e  $AG$ . Agora analisaremos o cenário 2, ou seja,  $\lambda_A > \mu$ , e com taxas assimétricas de transição entre os estados, i.e.,  $\delta_{GA} > \delta_{AG}$  e  $\delta_{AG} > \delta_{GA}$ . O caso onde não aplicamos rejuvenescimento pode ser analisado nas primeiras colunas da Tabela 5 onde  $\delta_{AG} > \delta_{GA}$  propicia números melhores no tocante a transações restantes no sistema, porcentagem de utilização do sistema e tempo de permanência no estado  $A$ , fator este ocasionado por uma permanência maior do sistema no estado  $G$  em relação ao estado  $A$ . Já as últimas colunas da Tabela 5 consideram o caso em que aplicamos o rejuvenescimento e podemos perceber um ganho em todos os indicadores apontados tanto quando temos  $\delta_{AG} > \delta_{GA}$  e  $\delta_{GA} > \delta_{AG}$ , quando comparamos com os indicadores sem rejuvenescimento.

### 3.2.2 Relação entre aplicação do rejuvenescimento e profundidade do *bucket*

**Questão 7:** Qual a relação entre rejuvenescimentos aplicados e a profundidade do *bucket*?

A seguir, nosso objetivo é determinar a relação entre o tempo de permanência no estado anômalo, frequência de rejuvenescimento e profundidade do *bucket*. A Figura 5 ilustra um comparativo destas métricas de interesse em cenários com rejuvenescimento.

Na Figura 5(a) consideramos quatro casos, variando em função dos dois cenários considerados na seção anterior. No caso onde temos  $\lambda_A > \mu > \lambda_G > \delta_{AG} > \delta_{GA}$ , por exemplo, o sistema tende a permanecer menos tempo no estado anômalo. Isto pode parecer contra-intuitivo, já que  $\lambda_A > \mu$  corresponde a um sistema que eventualmente passa por instabilidades. Entretanto, como a taxa de transição do estado  $A$  para  $G$  é alta ( $\delta_{AG} > \delta_{GA}$ )

Tabela 5 – Cenário 2: Estatísticas sobre número de transações atendidas e tempo nos estados  $G$  e  $A$ , variando taxa  $\theta$

	Sem rejuvenescimento		Com rejuvenescimento	
	$\delta_{GA} > \delta_{AG}$	$\delta_{AG} > \delta_{GA}$	$\delta_{GA} > \delta_{AG}$	$\delta_{AG} > \delta_{GA}$
Transações não atendidas	109,150	36,660	10,452	4,520
Utilização do sistema	90%	74%	59%	53%
Tempo no estado G	2.032,345	3.683,590	3.383,330	3.928,453
Tempo no estado A	3.412,354	1.692,620	1.114,578	569,882
Número de Rejuvenescimentos	0	0	26,084	26,070
Inatividade por Rejuvenescimento	0	0	10,90%	10,40%

os períodos de instabilidade são curtos, favorecendo menor permanência no estado anômalo.

Comparando os números indicados de tempo no estado A da Tabela 5 percebemos que todos os casos em que aplicamos o rejuvenescimento obtivemos um tempo de permanência no estado anômalo menor do que nos casos onde não aplicamos o rejuvenescimento, como era de se esperar.

A Figura 5 ilustra a tendência de queda no número de rejuvenescimentos aplicados à medida que a profundidade do *bucket* aumenta. Na figura assumimos  $\delta_{AG} = \delta_{GA}$ , sendo que o mesmo comportamento foi observado para  $\delta_{AG} > \delta_{GA}$  e  $\delta_{AG} < \delta_{GA}$ .

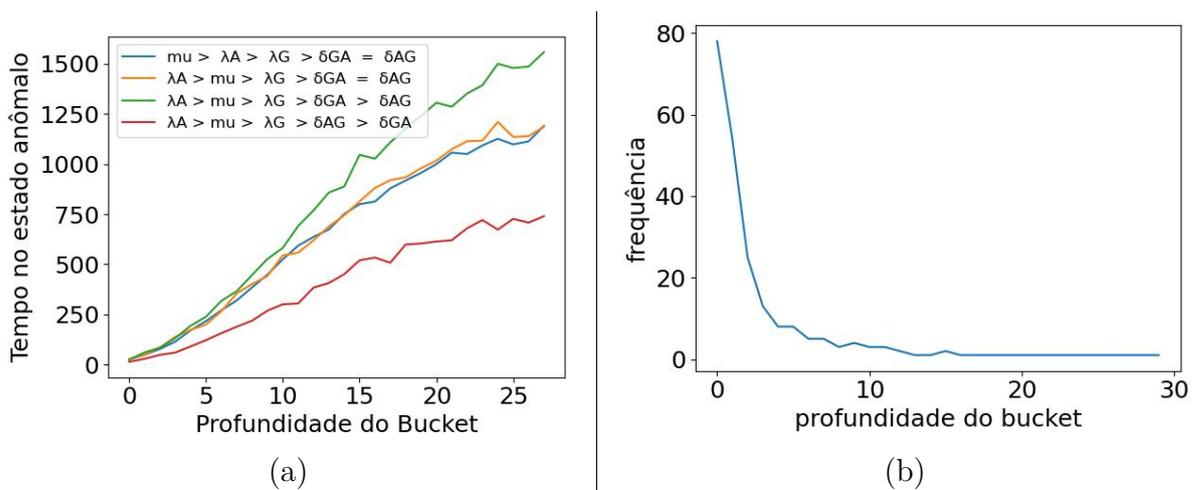


Figura 5 – (a) Comparativo de tempo de permanência no estado anômalo e profundidade do *bucket*, variando taxa  $\theta$  e com  $\delta_{RG} = 0,1$ ; (b) Relação entre número de rejuvenescimentos aplicados e profundidade do *bucket*

### 3.3 Detectando pontos de mudança: tempo de detecção e falsos positivos

Nesta seção, avaliamos o tempo de detecção de pontos de mudança, bem como a taxa de falsos positivos.

#### 3.3.1 Trabalhos relacionados: CUSUM para detecção de pontos de mudança em fontes MMPP

O CUSUM visa resolver o problema clássico de detecção de ponto de mudança sequencial. O CUSUM e os algoritmos derivados do CUSUM são amplamente utilizados para resolver este problema devido à sua simplicidade computacional. Além disso, não exigem guardar todas as observações colhidas ao longo do tempo.

Em (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016), os autores se dedicam ao problema de identificação de pontos de mudanças em fontes MMPP. Os tempos de chaveamento entre dois estados, que podemos considerar como normal e anômalo, são controlados por uma cadeia de Markov e o fluxo de eventos possui taxas distintas e respeita uma distribuição de Poisson. Identificando estes instantes de mudanças pode-se tomar decisões em tempo real sobre soar alarmes ou tomar contra-medidas.

Neste sentido, (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016) desenvolveu um algoritmo de detecção de ponto de mudança, no estilo do CUSUM, para identificar o estado atual da cadeia oculta Marcoviana que controla a taxa de chegada de um sistema de filas. Para construir o algoritmo, é usada a estatística de máxima verossimilhança e valores positivos são estabelecidos como limiares do algoritmo na construção das somas cumulativas (daí o nome CUSUM), que são recalculadas a cada nova chegada. Cada vez que o respectivo limite é atingido é feito um registro de transição de estado na cadeia. Assume-se que um falso positivo ocorre quando uma das somas cumulativas atinge o limite correspondente, mesmo que a intensidade do processo de chegada não corresponda a uma anomalia.

**Diferenças entre nosso trabalho e trabalhos anteriores.** É preciso destacar algumas diferenças entre as abordagens adotadas. Em (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016) o algoritmo CUSUM considera apenas as taxas de chegadas de *jobs* no sistema e as taxas de transição de estados para o cálculo dos tempos exponenciais do simulador, apenas estes tempos contribuem para o tempo de simulação. Já a proposta do MMPP com *bucket*  $M/M/1$ , abordado nesta pesquisa, além de considerar as mesmas taxas usadas em (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016), também considera outras taxas, como explicado em (3.3) e (3.9). Todos os tempos gerados contribuem para o tempo de simulação. Desta forma, a abordagem proposta, nesta pesquisa, visa entender

como outras variáveis podem interferir no resultado apresentado pelo sistema, em especial, o observador ( $\theta$ ) e a transição de  $RG$  ( $\delta_{RG}$ ).

### 3.3.2 Caracterizando pontos de mudança e falso positivos

O nosso objetivo é, com base nas informações geradas pela simulação do MMPP, com *bucket algorithm* e fila  $M/M/1/K: 1$ ) caracterizar falsos positivos gerados pelo sistema que, no contexto desta pesquisa, ocorrem quando o *bucket* atinge o seu limite, no estado bom  $G$  e 2) detectar as transições entre os estados. Em particular, a transição  $GA$  indica que o sistema passou para o estado anômalo e, conseqüentemente, a taxa de chegada de pacotes sobe. Já a transição  $AG$  indica o instante em que o sistema passou para o estado bom e, conseqüentemente, a taxa de chegada de pacotes tende a ser considerada normal. Estamos também particularmente interessados em contabilizar falsos positivos. Para tal, em nossa simulação, criamos um vetor que armazena os instantes que o sistema detectou falsos positivos (ver Equação (3.11)).

No Apêndice B caracterizamos formalmente os instantes de transição entre estados. Os instantes de transição entre estados são dados pelas Eqs. (B.1)-(B.3). Com base nos instantes de transições detectados podemos, de forma semelhante ao preconizado em (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016), calcular o tamanho e o número de intervalos das transições entre os estados (ver Equações (B.4) e (B.5)) e os estimadores das taxas de transição entre os estados da cadeia (ver Eq. (B.6)).

### 3.3.3 Resultados e análise

Visando entender como o algoritmo de *bucket* se comporta para detectar as transições de estados, e o papel do rejuvenescimento neste cenário, faremos dois estudos: 1) sem a aplicação de rejuvenescimento e 2) com a aplicação do rejuvenescimento.

A configuração utilizada nestes cenários será:  $\mu > \theta > \lambda_A > \lambda_G \gg \delta_{AG} > \delta_{GA}$ , em que os parâmetros são, respectivamente, iguais a 0,16; 0,14; 0,12; 0,08; 0,002 e 0,001. Esta configuração favorece o tempo de permanência no estado G, na medida em que  $\delta_{GA} < \delta_{AG}$ .

**Cenário sem rejuvenescimento.** A Tabela 6 mostra que o algoritmo de *bucket* consegue detectar todos os instantes em que houve uma transição de estados. Ao longo das rodadas de simulação consideradas, a média de transições  $GA$  e  $AG$  foi de 3,966 e 3,245, respectivamente, com estimadores de taxa dados por  $\hat{\delta}_{AG} = 0,0017$  e  $\hat{\delta}_{GA} = 0,0011$ .

**Cenário com rejuvenescimento.** A Tabela 7 ilustra o caso em que aplicamos o rejuvenescimento ( $\delta_{RG} = 0,5$ ). Podemos perceber que o algoritmo de *bucket* continua detectando todas as transições ocorridas e que o número médio de transição entre os estados foi menor que no caso anterior: o número médio de transições  $GA$  e  $AG$  foi 2,802 e 0,938, respectivamente, com estimadores de taxas dados por  $\hat{\delta}_{AG} = 0,0023$  e  $\hat{\delta}_{GA} = 0,0015$ .

A Figura 6 ilustra a tendência de queda no número de falsos positivos, à medida que a profundidade do *bucket* aumenta. Quanto maior a profundidade do *bucket*, menor o número de falsos positivos, mas maior o tempo para detectar anomalias (GONÇALVES et al., 2020).

A Figura 6 também ilustra o impacto da taxa de amostragem do sistema,  $\theta$ , no número de falsos positivos. Quanto maior a taxa de amostragem, maior a quantidade de falsos positivos, mas menor o tempo para detectar anomalias. De todo modo, em todos os cenários podemos observar um número baixo de falso positivos nos cenários considerados, para  $D \geq 12$  como ilustrado na Figura 6. <sup>2</sup>

Tabela 6 – Detecção de transições, sem rejuvenescimento

Transição	Instante real em que ocorreu transição	Instante em que foi detectada transição	Número de falsos positivos no intervalo
GA	2.325,030	2.351,356	4 (falsos positivos ocorreram antes do instante 2.325,030)
AG	2.371,800	2.384,766	0
GA	3.136,038	3.167,499	2 (falsos positivos ocorreram entre os instantes 2.371,800 e 3.136,038)
AG	4.027,041	4.062,535	0
GA	4.127,405	4.143,214	0
AG	4.770,306	4.772,476	1 (falso positivo ocorreu após instante 4.770,306)

Tabela 7 – Detecção de transições, com rejuvenescimento

Transição	Instante real em que ocorreu transição	Instante em que foi detectada transição	Número de falso positivos no intervalo
GA	1.282,052	1.284,400	2 (falsos positivos ocorreram antes do instante 1.282,052)
AG	1.711,152	1.712,132	0
GA	2.023,167	2.026,752	0
AG	2.725,171	2.728,275	9 (falsos positivos ocorreram após instante 2.725,171)

<sup>2</sup> Estes resultados concordam com aqueles reportados em (GONÇALVES et al., 2020), onde os autores também observaram que a profundidade de  $D = 12$  é adequada para os cenários lá considerados.

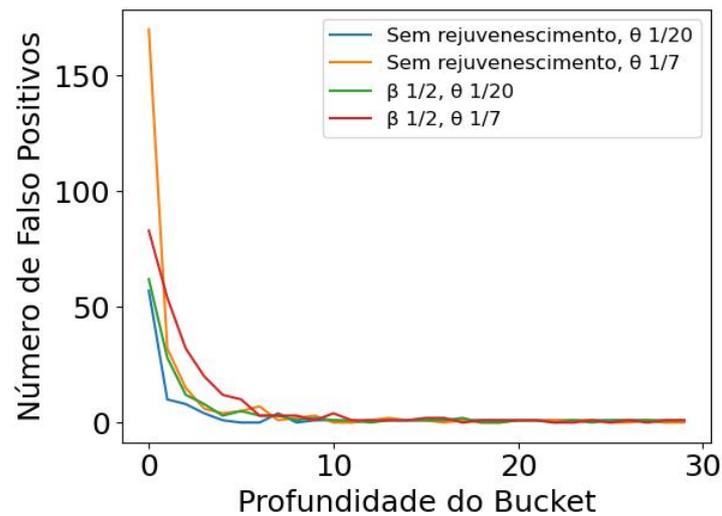


Figura 6 – Relação entre detecção de falso positivos e profundidade do *bucket*

### 3.4 O papel da informação na tomada de decisão

Até o momento abordamos duas estratégias para rejuvenescimento:

- **rejuvenescimento sem informação de estado:** a primeira estratégia, coberta no Capítulo 4 e em (MIRANDA et al., 2022), abordou o rejuvenescimento aleatório, onde nenhuma informação a respeito da criticidade do sistema era considerada para a aplicação do rejuvenescimento e portanto, o sistema poderia ser rejuvenescido em momentos onde não tenha chegado em um ponto crítico, podendo ainda se recuperar naturalmente. Em que pese o fato desta abordagem prolongar a vida útil do sistema, evitando que o *bucket* atinja seu limite, observamos que o custo de indisponibilidade, por conta do rejuvenescimento, é alta e na sua grande maioria desnecessária.
- **rejuvenescimento com informação parcial de estado:** a segunda abordagem considera o caso onde temos informação parcial do sistema (*bucket* atingindo o limite definido pelo administrador e disparando o rejuvenescimento no estado  $G$  ou no estado  $A$ ), percebemos que, embora o limite do *bucket* seja atingido mais rapidamente, os rejuvenescimentos ocorridos são menores e mais assertivos, desta forma, obtemos um tempo de inatividade, por conta dos rejuvenescimentos, menor. Esta abordagem foi descrita na Seção 4.1.3.3, bem como também coberta no presente capítulo, até a seção anterior.

O objetivo agora é analisarmos a situação hipotética em que o administrador do sistema tenha conhecimento geral do estado do sistema. Em especial, o administrador sabe se o sistema está no estado bom ou no estado anômalo. No estado bom (resp., anômalo), a taxa de chegada para a fila do servidor central é  $\lambda_G$  (resp.,  $\lambda_A$ ). Assumimos  $\lambda_A > \lambda_G$ , onde

$\lambda_A = \lambda_G + \lambda_B$ . A taxa  $\lambda_B$  corresponde à chegada de *background packets* que consomem recursos do sistema.

**Rejuvenescimento com informação completa.** Se o sistema está no estado bom, ele não está em situação de risco e eventuais atrasos são ocasionados por degradações transientes do sistema. Tais degradações são resolvidas naturalmente, sem a necessidade de intervenção do administrador. O rejuvenescimento será aplicado somente quando o sistema estiver no estado anômalo e o algoritmo de *bucket* atingir o seu limite, levando o sistema para sua situação inicial (Eq. (3.10)). Caso o sistema atinja o seu limite no estado bom, apenas o número de *tokens* do *bucket* será zerado, mas o sistema permanecerá no estado corrente e com seu número de *jobs* inalterados, sinalizando que o algoritmo de *bucket* potencialmente causaria um falso positivo.

**Cenários considerados.** Para a nossa análise configuramos nosso simulador da seguinte forma:

- **no cenário de conhecimento parcial do sistema:**  $\lambda_A > \mu > \lambda_G \gg \delta_{GA} \gg \delta_{AG}$ , (0,25, 0,16, 0,08, 0,01, 0,001), respectivamente, nesta configuração o sistema tende a estar em risco frequentemente, pois temos  $\lambda_A > \mu$  e  $\delta_{GA} \gg \delta_{AG}$ , ou seja, taxa de chegada de pacotes superior à taxa de serviço e permanência maior no estado *A*.
- **informação completa, caso 1:** uma vez conhecendo todas as variáveis do sistema, consideramos como primeiro cenário de ajustes de taxas o seguinte:  $\lambda_A > \mu > \lambda_G \gg \delta_{GA} = \delta_{AG}$ , (0,25, 0,16, 0,08, 0,002), respectivamente, onde mantemos uma carga de chegada de pacotes superior à taxa de serviço e ajustamos apenas as taxas de transições de  $\delta_{GA}$  e  $\delta_{AG}$  e
- **informação completa, caso 2:** uma vez conhecendo todas as variáveis do sistema, consideramos como segundo cenário de ajustes de taxas o seguinte:  $\mu > \lambda_A > \lambda_G \gg \delta_{GA} = \delta_{AG}$ , (0,16, 0,12, 0,08, 0,002), onde colocamos a taxa de serviço superior à carga de chegada de pacotes anômalos e igualamos as taxas de transição entre os estados.

Para esta análise usaremos como profundidade do *bucket*  $D = 12$ , ( $\delta_{RG} = 0,1$ ) e  $\theta$  será incrementado a cada iteração do simulador.<sup>3</sup>

A Tabela 8 mostra que no cenário onde temos informação parcial do sistema, os números de transações não atendidas e utilização do sistema são elevados, comparados aos cenários onde possuímos informação completa do sistema, o que demonstra: 1) dificuldade do sistema em atender a demanda de *jobs*, fato este devido a  $\lambda_A > \mu$  e, 2) que o sistema permanecer um tempo excessivo no estado anômalo. Já no caso, onde temos informação

<sup>3</sup> Cabe destacar que uma vez que  $\lambda_A$  é, na grande maioria das vezes, desconhecida, em trabalhos futuros podemos considerar também a manipulação de  $\mu$  no intuito de estabilizar o sistema.

completa do sistema e configuramos  $\delta_{GA} = \delta_{AG}$  e o rejuvenescimento para ocorrer apenas no estado anômalo, observamos uma queda no número de transações não atendidas pelo sistema e uma utilização média do sistema menor, bem como um tempo médio de permanência no estado anômalo, este último devendo ser atribuído a  $\delta_{GA} = \delta_{AG}$ . Em um terceiro cenário temos,  $\delta_{GA} = \delta_{AG}$  e  $\mu > \lambda_A$ , e repetimos a estratégia de aplicação do rejuvenescimento apenas no estado anômalo e podemos observar na Tabela 8 uma queda expressiva apenas no número de transações não atendidas pelo sistema, em decorrência de termos nesta configuração  $\mu > \lambda_A$ .

Tabela 8 – Comparativo de aplicabilidade de rejuvenescimento ajustando taxas

	Informação parcial	Informação completa	
		caso 1, $\lambda_A > \mu$	caso 2, $\mu > \lambda_A$
<i>Jobs</i> não atendidos	25,541	6,893	1,385
Utilização do sistema	84%	62%	54%
Tempo no estado G	1.771,180	3.784,677	3.775,599
Tempo no estado A	2.728,188	1.114,570	1.117,929
Rejuvenescimento no estado G	11,404	0	0
Rejuvenescimento no estado A	8,990	5,340	5,660
Inatividade por Rejuvenescimento	15,056%	2%	2%

**Número de tarefas atendidas e não atendidas.** A Figura 7 apresenta o número médio dos *jobs* que chegaram e partiram do sistema, nos três casos considerados (informação parcial, e informação completa casos 1 e 2). No caso, onde temos informação parcial do sistema, obtemos um alto número de chegadas e partidas de *jobs* do sistema. Entretanto, a diferença entre esses números também é grande, indicando um alto número de *jobs* não finalizados pelo sistema. Isto reforça a dificuldade do sistema em lidar com  $\lambda_A > \mu$ . Fazendo um comparativo com os casos onde temos a informação completa do sistema e restringimos a aplicação do rejuvenescimento apenas ao estado  $A$ , percebemos que a diferença entre chegadas e partidas tende a diminuir, em especial, no caso onde temos  $\delta_{GA} = \delta_{AG}$  e  $\mu > \lambda_A$ . Neste caso, o número médio de *jobs* não atendidos pelo sistema apresenta tendência de queda. A Figura 7 deixa claro que  $\lambda_A > \mu$  favorece o aumento de *jobs* não atendidos pelo sistema, à medida que a profundidade do *bucket* cresce. Entretanto, quando temos  $\mu > \lambda_A$ , este indicador tende a apresentar uma regularidade.

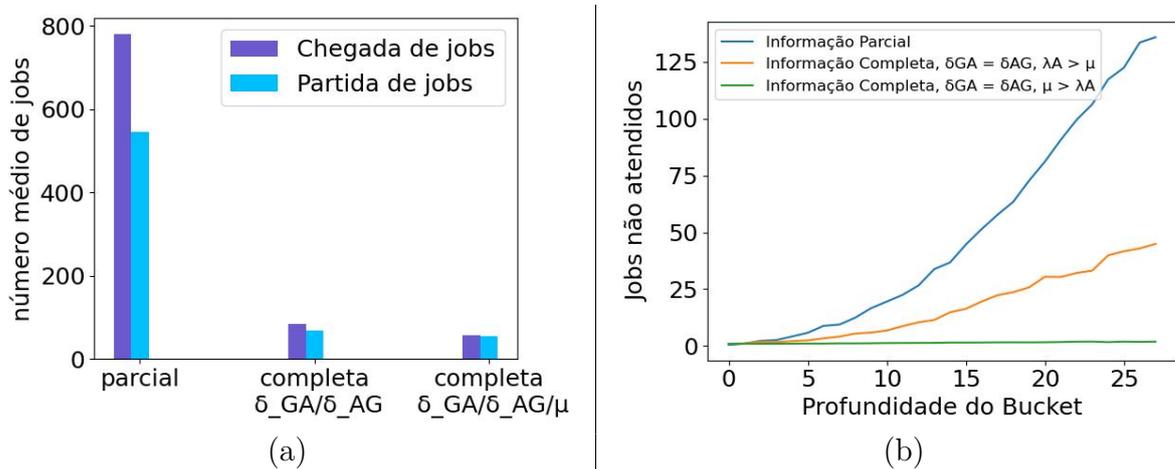


Figura 7 – O papel da informação na tomada de decisão: (a) chegadas e partidas de *jobs* do sistema nos casos de informação parcial e informação completa com  $\mu < \lambda_A$  e  $\mu > \lambda_A$ ; (b) *jobs* não atendidos pelo sistema e profundidade do *bucket*

**Resumo.** Obtemos melhores indicadores para o sistema quando possuímos conhecimento completo do mesmo e quando isolamos a aplicabilidade do rejuvenescimento para situações nas quais este é de fato necessário. Observamos que, para obtermos uma permanência menor no estado  $A$  em relação ao estado  $G$ , é útil conhecer ou estimar as taxas de transição  $\delta_{GA}$  e  $\delta_{AG}$ . Além disso, manter  $\mu > \lambda_A$  garante um fluxo mais regular entre chegadas e partidas, mesmo em situações anômalas, motivando em trabalhos futuros fazermos um controle dinâmico da taxa de serviço  $\mu$ . No entanto, manter informação completa sobre o sistema pode envolver custos: 1) quando o sistema mostra-se instável, é interessante averiguar a causa do aumento de  $\lambda_A$  em relação a  $\mu$ , que pode ter diversas origens; e 2) de forma mais geral, é possível investir em infraestrutura para atender um grande fluxo de chegada de *jobs* no sistema, mesmo em situações de anomalia, aumentando a capacidade de pico do sistema caso exista orçamento para tal.

## 4 MODELOS COMPARTIMENTALIZADOS

Neste capítulo, consideramos modelos compartimentalizados para análise sequencial de desempenho de sistemas que rejuvenescem. Começamos considerando a análise de sistemas sem anomalias, com o propósito de entender a taxa de falsos positivos (Seção 4.1). Em seguida, contabilizamos anomalias, mas ainda de forma compartimentalizada, ou seja, sem caracterizar explicitamente o estado das filas e da fonte de anomalias do sistema (Seção 4.2).

### 4.1 Configuração da linha de base sem anomalias: modelando a taxa de falsos alarmes considerando o rejuvenescimento

Nesta seção, consideramos a análise de desempenho sequencial de sistemas que envelhecem e rejuvenescem, sob a suposição de que nenhuma anomalia ocorrerá durante o horizonte de tempo de interesse. Este corresponde a *gold run* em (GONÇALVES et al., 2020). Como assumimos que não ocorre nenhuma anomalia, os alarmes correspondem a falsos alarmes. Pretendemos responder à seguinte questão:

**Questão 1:** *Quanto tempo leva para um falso positivo ser acionado, ao executar uma análise de desempenho sequencial, por exemplo, usando o algoritmo de bucket, de um sistema que envelhece e rejuvenesce?*

Sem levar em conta o rejuvenescimento, a questão acima foi respondida em (GONÇALVES et al., 2020). A seguir, 1) mostramos que os resultados da literatura sobre processos de nascimento-morte com catástrofes são fundamentais para a análise do algoritmo de *bucket* sob rejuvenescimento e 2) indicamos como o rejuvenescimento aumenta o tempo até um falso positivo.

#### 4.1.1 Processos de nascimento-morte com catástrofes são fundamentais para estudar análise de desempenho sequencial com rejuvenescimento

Os processos de nascimento-morte com catástrofes são estudados sob diferentes configurações, por exemplo, contabilizando  $M/M/1$  com catástrofes (KUMAR; ARIVUDAINAMBI, 2000) e  $M/M/1/K$  com catástrofes (BÖHM, 2008). A seguir, mostramos que esses resultados são fundamentais para caracterizar o algoritmo de *bucket* para fins de análise de desempenho sequencial com rejuvenescimento.

Considerando o estado  $k$  correspondendo ao estado do alarme, o tempo até um

alarme falso é o tempo para atingir o estado  $k$  a partir do estado 0. Seja  $\bar{T}_{0,k}$  o tempo médio para chegar ao estado  $k$  a partir do estado 0. A equação (15) in (CRESCENZO et al., 2003) caracteriza o tempo médio para atingir um determinado estado em um processo de nascimento-morte com catástrofes, a partir de qualquer estado dado. Curiosamente, segue da Equação (15) em (CRESCENZO et al., 2003) que

$$\bar{T}_{0,k} = \frac{1 - \hat{\gamma}_{0,k}(\xi)}{\xi \hat{\gamma}_{0,k}(\xi)} = \frac{1}{\xi \hat{\gamma}_{0,k}(\xi)} - \frac{1}{\xi} \quad (4.1)$$

onde

$$\hat{\gamma}_{0,k}(\xi) = \prod_{i=0}^{k-1} \frac{\alpha_i}{\alpha_i + \xi} \quad (4.2)$$

e  $\alpha_i$  é o  $i$ -ésimo autovalor da matriz geradora infinitesimal negativa truncada do processo nascimento-morte sem catástrofes (FILL, 2009; BROWN; SHAO, 1987).

#### 4.1.2 Exemplo: um modelo de três estados correspondente a um *bucket* de profundidade um

A Figura 8 mostra as cadeias de Markov correspondentes ao algoritmo de *bucket* original sob envelhecimento e recuperação natural, Figura 8(a), e sob envelhecimento e rejuvenescimento, Figura 8(b). Recuperação natural e rejuvenescimento são combinados na Figura 8(c), onde o estado 0 é o estado inicial e os estados 1 e 2 representam estágios de envelhecimento.

Sejam  $\lambda$ ,  $\mu$  e  $\xi$  a taxa de degradação, recuperação natural e rejuvenescimento, e  $\rho = \lambda/\mu$ .

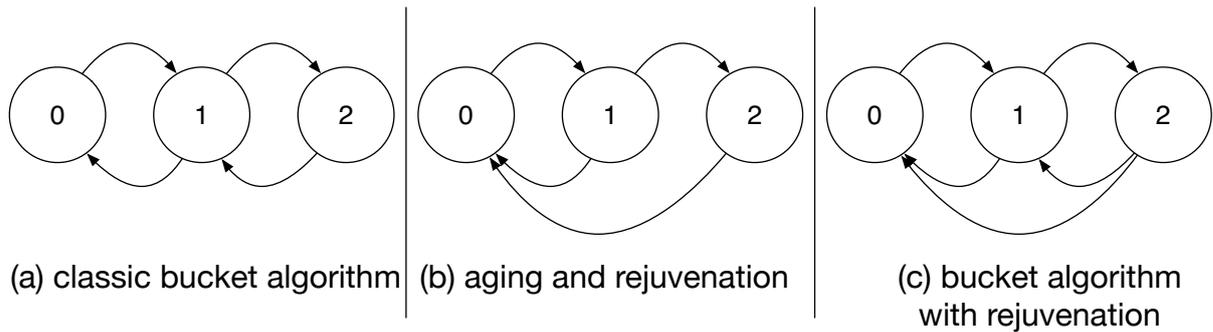


Figura 8 – Clássico algoritmo de *bucket*, com envelhecimento e rejuvenescimento

A seguir, consideramos os três casos acima:

- **algoritmo de *bucket*: Figura 8(a).** A solução de estado estacionário do algoritmo de *bucket* é dada por

$$\pi_0 = \frac{1}{1 + \rho + \rho^2}, \quad \pi_1 = \pi_0 \rho, \quad \pi_2 = \pi_0 \rho^2 \quad (4.3)$$

e o tempo médio para alcançar o estado 2 do estado 0 é  $T^{(a)} = 1/\tilde{\pi}_2 - 1$ . Neste caso,

$$\frac{1}{\tilde{\pi}_2} = \frac{\mu}{\lambda^2} + \frac{2}{\lambda} + 1. \quad (4.4)$$

- **envelhecimento e rejuvenescimento: Figura 8(b).** A solução de estado estacionário é dada por

$$\pi_0^{(b)} = \frac{1}{1 + \frac{\lambda}{\lambda+\xi} + \frac{\lambda}{\lambda+\xi} \frac{\lambda}{\xi}}, \quad \pi_1^{(b)} = \pi_0^{(b)} \frac{\lambda}{\lambda + \xi}, \quad (4.5)$$

$$\pi_2^{(b)} = \pi_0^{(b)} \frac{\lambda}{\lambda + \xi} \frac{\lambda}{\xi} \quad (4.6)$$

e o tempo médio para alcançar o estado 2 do estado 0 é  $T^{(b)} = 1/\tilde{\pi}_2^{(b)} - 1$  onde  $\tilde{\pi}_2^{(b)}$  é dado por (4.4) substituindo  $\mu$  por  $\xi$ .

- **algoritmo de *bucket* com envelhecimento e rejuvenescimento: Figura 8(c).** A solução de estado estacionário é

$$\pi_0^{(c)} = \pi_2^{(c)} \left( \left( \frac{\mu + \xi}{\lambda} \right)^2 + \frac{1}{\lambda} \right), \quad \pi_1^{(c)} = \pi_2^{(c)} \left( \frac{\mu + \xi}{\lambda} \right)$$

$$\pi_2^{(c)} = \frac{1}{1 + \frac{\mu+\xi}{\lambda} + \frac{\xi}{\lambda} + \left( \frac{\mu+\xi}{\lambda} \right)^2} \quad (4.7)$$

e o tempo médio para alcançar o estado 2 a partir do estado 0 é

$$T^{(c)} = \frac{1}{\tilde{\pi}_2^{(c)}} - 1 \quad (4.8)$$

onde  $\tilde{\pi}_2^{(c)}$  é dado por (4.4) substituindo  $\mu$  por  $\mu + \xi$ .

Observe também que a matriz geradora infinitesimal negativa truncada do processo nascimento-morte sem catástrofes é dada por

$$\tilde{Q} = \begin{pmatrix} \lambda & -\lambda \\ -\mu & \lambda + \mu \end{pmatrix} \quad (4.9)$$

cujos autovalores são dados por

$$\alpha_{1,2} = \lambda + \frac{\mu}{2} \pm \frac{\sqrt{\mu(4\lambda + \mu)}}{2} \quad (4.10)$$

Então, segue de (4.1) e (4.8) que

$$T^{(c)} = \frac{(\alpha_1 + \xi)(\alpha_2 + \xi)}{\xi \alpha_1 \alpha_2} - \frac{1}{\xi} = \frac{\mu + \xi}{\lambda^2} + \frac{2}{\lambda}. \quad (4.11)$$

**Resumo:** nesta seção, consideramos uma configuração simples em que o algoritmo de *bucket* está sujeito a rejuvenescimento, na ausência de anomalias (falhas ou ataques). Por meio deste exemplo conseguimos ilustrar como os resultados dos processos de nascimento-morte com catástrofes são aplicáveis à análise do rejuvenescimento. A seguir, consideramos o modelo de tempo discreto correspondente ao modelo de tempo contínuo considerado nesta seção e indicamos como uma distribuição geométrica parametrizada pode ser instrumental para aproximar o tempo até um falso positivo.

### 4.1.3 Caso de tempo discreto

Até agora, consideramos o tempo médio para falso positivo medido para um processo de tempo contínuo. Um modelo de tempo discreto foi considerado em (GONÇALVES et al., 2020). Em seguida, ilustramos como o rejuvenescimento afeta o comportamento do algoritmo de *bucket* na configuração de tempo discreto.

**Questão 2:** *Como a solução para o modelo de tempo contínuo se relaciona com a de um modelo de tempo discreto?*

O número médio de transições  $N$  até um falso positivo se relaciona com o tempo contínuo médio  $T$  até um falso positivo como

$$N_0 \frac{1}{\tau_0} + N_{>0} \frac{1}{\tau_{>0}} = T \quad (4.12)$$

$$N_0 + N_{>0} = N \quad (4.13)$$

onde  $N_0$  é o número médio de visitas ao estado 0, antes da absorção, e  $N_{>0}$  é o número médio de visitas a outros estados. Temos duas equações e três incógnitas. Para resolver o sistema de equações acima, precisamos considerar as equações de equilíbrio adicionais, conforme ilustrado em nosso exemplo.

#### 4.1.3.1 Voltando ao exemplo simples

A seguir, revisitamos nosso exemplo Figura 8(c) à luz do modelo de tempo discreto. Nós temos:

$$N_0 \frac{1}{\lambda} + N_{>0} \frac{1}{\lambda + \mu + \xi} = \frac{\mu + \xi}{\lambda^2} + \frac{2}{\lambda} \quad (4.14)$$

$$N_0 + N_{>0} = N \quad (4.15)$$

$$1 + N_{>0} \frac{\xi + \mu}{\lambda + \mu + \xi} = N_0 \quad (4.16)$$

As duas primeiras equações acima correspondem a (4.12) e (4.13). A última equação é a equação de equilíbrio para o estado 1.

Segue da solução do conjunto de equações acima que o número de amostras até um falso positivo no exemplo é dado por

$$N = 2 \left( \frac{\mu + \xi + \lambda}{\lambda} \right). \quad (4.17)$$

### 4.1.3.2 Avaliação numérica e aproximação geométrica

Em seguida, consideramos um modelo simples para aproximar o número de amostras até um alarme falso, sob o modelo de tempo discreto.

**Questão 3:** *Podemos aproximar o número de amostras até um falso positivo por meio de uma distribuição geométrica?*

Para responder à questão acima, consideramos uma simulação simples do sistema com envelhecimento e rejuvenescimento. Em seguida, indicamos um regime em que o modelo geométrico captura com precisão a distribuição do número de amostras até um falso positivo, e ilustramos como parametrizá-lo para fins de estimativa do número médio de amostras até que o falso positivo seja acionado.

---

**Algorithm 1** Simulação: algoritmo de *bucket* com rejuvenescimento

---

```

0: while  $d \leq D$  do
   $x_1 \leftarrow U[0, 1]$  (variável uniforme aleatória  $x_1 \in [0, 1]$ )
  if  $x_1 < R$  (probabilidade de rejuvenescimento) then
     $d \leftarrow 0$ 
  else
     $x_2 \leftarrow U[0, 1]$  (variável uniforme aleatória  $x_2 \in [0, 1]$ )
    if  $x_2 < L$  (probabilidade de perda transitória) then
       $d \leftarrow d + 1$  (adicionar token ao bucket, devido à perda)
    else
       $d \leftarrow d - 1$  (remover token do bucket, devido ao sucesso)
    end if
  end if
end while=0

```

---

**Configuração da simulação.** Desenvolvemos um simulador de análise de desempenho sequencial que implementa a lógica do algoritmo de *bucket*, aplicando rejuvenescimento aleatório e a partir de qualquer estado do sistema, sem relação direta com o estado do *bucket*, que indica a criticidade do sistema (ver algoritmo 1). Trabalhamos com um *bucket* ( $B = 1$ ), variando sua profundidade de 1 a 28 ( $1 \leq D \leq 28$ ). Para os propósitos desta seção, consideramos o ambiente livre de anomalias, visando responder questões relacionadas ao tempo que leva para um falso positivo ser acionado, variando a probabilidade de rejuvenescimento  $R$ .

Probabilidade de Perda Transitória (TLP) refere-se à probabilidade de adicionar um *token* ao *bucket*, devido ao envelhecimento, e  $(1 - \text{TLP})$  é a probabilidade de remover um *token*. Motivado por (GONÇALVES et al., 2020), em nossos experimentos consideramos um TLP de 0,46 (esta é a probabilidade de adicionar *tokens* ao *bucket*, parametrizado com base em dados reais do *benchmark* TPCx-V). Em seguida, estendemos (GONÇALVES et al., 2020) contabilizando o rejuvenescimento. Neste sentido, trabalhamos com uma

probabilidade de rejuvenescimento que variou de 1% a 10%, nos vários exemplos que foram feitos.

Cada iteração do algoritmo 1 corresponde a uma nova amostra (por exemplo, *throughput*) sendo coletada. No algoritmo 1, amostramos um número entre 0 e 1 uniformemente de forma aleatória, para determinar se o rejuvenescimento ocorrerá. Nesse caso, o número de *tokens* do *bucket* é redefinido para zero. Caso contrário, amostramos outro número entre 0 e 1 uniformemente de forma aleatória, para determinar se um *token* deve ser adicionado ou removido, correspondendo aos casos em que o *throughput* amostrado está abaixo ou acima do TLP, respectivamente. Observe que, para fins do algoritmo, o limite é capturado indiretamente por meio do TLP denotado por  $L$ . Finalmente, quando  $d$  ultrapassa  $D$ , um alarme é acionado.

**Comparativo entre Distribuição exponencial e Distribuição de tempo para um alarme falso.** Para comparar o comportamento da distribuição exponencial em relação ao tempo de um falso positivo obtido por simulações, consideramos os quantis das duas distribuições.

As amostras de tempo até um falso positivo foram obtidas usando o algoritmo 1. A distribuição exponencial foi parametrizada usando os dados simulados e o erro dos mínimos quadrados (LSE), para isso usamos as funções SciPy `expon.fit` e `expon.rvs`. A pdf da distribuição exponencial é dada por

$$f(x) = e^{-(x-l)/s}/s \quad (4.18)$$

onde  $l$  e  $s$  são a localização e a escala da distribuição, derivadas usando LSE. Observe não haver significado físico direto para os parâmetros da distribuição exponencial em nosso domínio de interesse. Contornamos essa deficiência usando a distribuição geométrica na seção a seguir.

Parametrizamos  $D = 12$  e notamos que a distribuição exponencial se aproxima muito do comportamento do sistema. A Figura 9 considera a distribuição exponencial em função do tempo para um falso positivo sem rejuvenescimento com  $D = 12$  e  $R = 0$ , e a Figura 10 considera a distribuição exponencial em função do tempo para um falso positivo com rejuvenescimento  $D = 12$  e  $0,05$ . Comportamento semelhante foi observado em outras profundidades e com diferentes taxas de rejuvenescimento, na medida em que  $D \leq 28$  e  $R$  é menor que  $0,5$ . Observamos que para valores maiores de  $R$  e  $D$  (não mostrados na figura), o erro da distribuição exponencial se torna mais significativo.

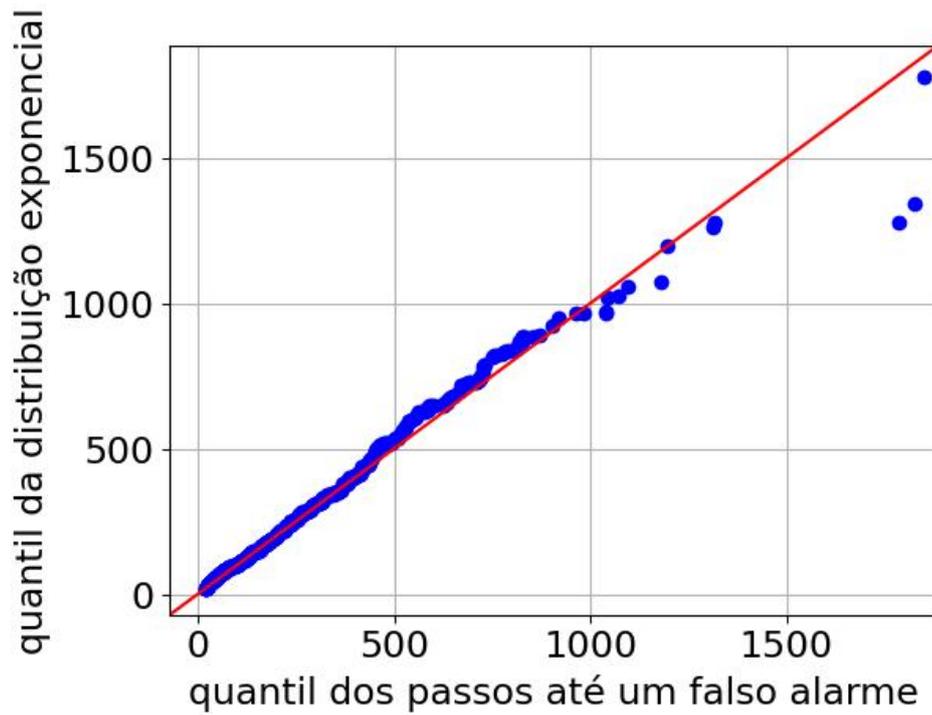


Figura 9 – Comparação de quantis sem rejuvenescimento

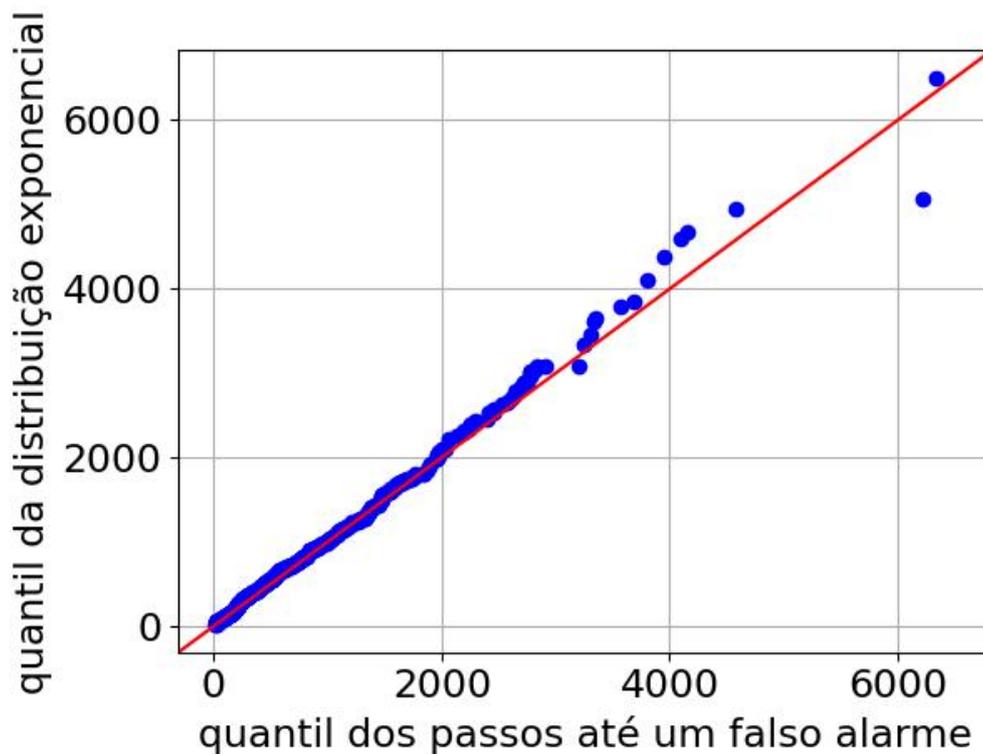


Figura 10 – Comparação de quantis com rejuvenescimento

**Um modelo geométrico de tempo médio para falso positivo.** Consideramos o seguinte modelo geométrico para aproximar o tempo médio até um alarme falso:

$$\tilde{T} \approx \frac{\kappa_1}{(L \cdot (1 - R))^{\kappa_2 \cdot D}} \quad (4.19)$$

onde  $\kappa_1$  e  $\kappa_2$  são constantes parametrizadas para ajustar o modelo aos dados de simulação. A lógica por trás do modelo consiste em assumir que o tempo médio até um falso positivo pode ser aproximadamente aproximado pelo tempo médio até que uma sequência de *tokens*  $D$  seja adicionada ao *bucket*, onde cada *token* é adicionado com probabilidade  $L \cdot (1 - R)$ , ou seja, a probabilidade de que ocorra uma probabilidade de perda transitória e não ocorra rejuvenescimento.

No cenário considerado acima, temos  $L = 0,46$  e  $R$  variando entre 0 e 10%. A Tabela 9 mostra os valores de  $\kappa_1$  e  $\kappa_2$  que melhor se ajustam aos nossos dados, e a Figura 11 indica que de fato um modelo geométrico pode capturar o comportamento do sistema para  $D > 3$ .

**Aprendizado.** A Figura 11 mostra o impacto do rejuvenescimento no tempo médio até um falso positivo. Como esperado, quanto mais agressivo for o rejuvenescimento, maior será o tempo até um falso positivo. Aumentando a probabilidade de rejuvenescimento de 0% para 10%, por exemplo, quando  $D = 12$ , observamos um aumento de quatro vezes no número médio de amostras até um falso positivo. Na seção 4.2, também consideramos os potenciais efeitos colaterais do rejuvenescimento em termos de indisponibilidade do sistema durante o rejuvenescimento.

Tabela 9 – Modelo analítico

Rejuvenation	$\kappa_1$	$\kappa_2$	Rejuvenation	$\kappa_1$	$\kappa_2$
1%	33,789	0,312	6%	20,088	0,509
2%	24,729	0,372	7%	9,334	0,566
3%	12,791	0,442	8%	13,923	0,570
4%	15,555	0,464	9%	9,964	0,606
5%	12,736	0,503	10%	14,169	0,609

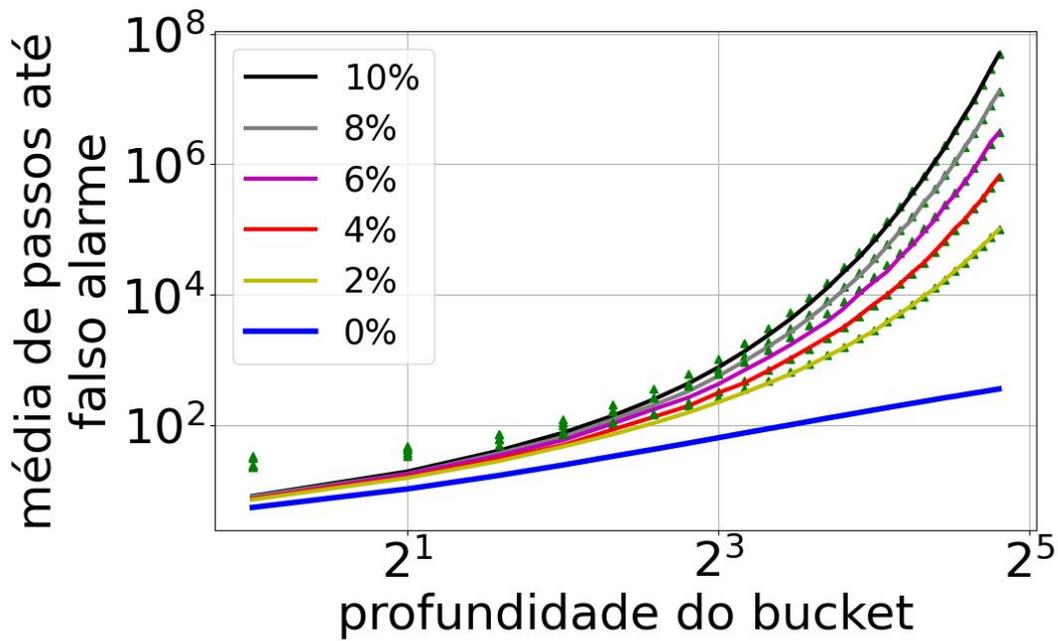


Figura 11 – O rejuvenescimento ajuda a evitar falsos alarmes: a Figura mostra um aumento no número médio de passos até que o *bucket* transborde (falso positivo) e contrasta a simulação com um modelo analítico que aproxima o comportamento das simulações por meio de uma distribuição geométrica.

#### 4.1.3.3 Comparação entre rejuvenescimento aleatório e rejuvenescimento oriundo do *bucket*

Agora faremos um breve comparativo entre duas abordagens de rejuvenescimento: 1) rejuvenescimento aleatório, sem relação com o estado do sistema, e 2) rejuvenescimento advindo do estouro do *bucket*, sinalizando a criticidade do sistema. Em (MIRANDA et al., 2022) o rejuvenescimento ocorre de forma aleatória e sem relação direta com o estado do sistema, este cenário nos leva a reflexão sobre a real efetividade destes rejuvenescimentos, uma vez que eles podem ocorrer de forma subsequentes uns dos outros e quando o *bucket* não chegou no seu limite, o que indicaria que o sistema ainda possuiu certa resiliência frente as anomalias, podendo se recuperar de forma natural, sem a necessidade de uma intervenção dos administradores. Este fato, impacta diretamente na disponibilidade do sistema, uma vez que, sempre que ocorre um rejuvenescimento, é imposta uma indisponibilidade ao sistema de tempo aleatório. Desta forma, estendemos (MIRANDA et al., 2022) e procuramos integrar o processo MMPP ao *bucket*, tornando o *bucket* uma fila  $M/M/1$  como descrito em 2.6, assim, o rejuvenescimento é disparado a partir do estouro do *bucket*, indicando uma situação crítica, situação em que o sistema não consegue mais se recuperar de forma natural, pretendemos responder a seguinte questão:

**Questão 4:** Qual o impacto do rejuvenescimento nas duas abordagens propostas?

Para efeitos desta comparação, no caso do rejuvenescimento aleatório, utilizamos como profundidade do *bucket*  $D = 12$  e com  $R = 0,1$ , no caso do rejuvenescimento oriundo

do *bucket* seguimos a mesma configuração do caso anterior, entretanto temos que considerar a taxa do observador  $\theta = 0,1$ , que irá regular a probabilidade de inserirmos ou removermos um *token* do *bucket*, para ambos os casos o tempo de inatividade do sistema, em decorrência de rejuvenescimento aplicado, será  $R + \delta_{RG}$  que são respectivamente, rejuvenescimento, momento em que o sistema é restaurado a sua condição inicial e transição do estado  $R$  para o estado  $G$ , os valores constantes da Tabela 10 são referentes a médias obtidas em 500 iterações da simulação.

Quando fixamos atenção ao caso do rejuvenescimento aleatório, sem relação com o estouro do *bucket*, percebemos que o sistema apresenta um número elevado de rejuvenescimentos precoces  $\approx 99\%$ , antes do *bucket* atingir seu limite e, conseqüentemente, quando o sistema poderia ainda possuir uma certa resiliência, como mostrado na Figura 12. Verificando as médias obtidas nas simulações constantes da Tabela 10 percebemos, no caso da aplicação do rejuvenescimento aleatório, um número baixo de rejuvenescimento ocorridos quando o *bucket* atingiu seu limite e realmente chegou em um nível de criticidade, o que necessitaria de intervenção dos administradores. Analisando esta abordagem, percebemos que levamos mais tempo para atingir o limite do *bucket* e, conseqüentemente, disparar um alerta de situação crítica ilustrado na Figura 13, entretanto este atraso em disparar o sinal de criticidade do sistema é decorrente de rejuvenescimentos excessivos, quando o *bucket* não atingiu seu limite e ainda poderia se recuperar naturalmente de uma degradação transiente ou até mesmo de um possível ataque.

Ao analisarmos os resultados do rejuvenescimento oriundo do *bucket* percebemos que, o limite do *bucket* é atingido mais rapidamente se comparado ao caso do rejuvenescimento aleatório, como mostra a Figura 14, fato este ocasionado pelos números reduzidos de rejuvenescimentos aplicados nesta abordagem. Os rejuvenescimentos advindos do *bucket* são consideravelmente menores e mais assertivos, como mostrado na Tabela 10. Como consequência, o tempo de inatividade do sistema, em decorrência de rejuvenescimentos desnecessários, é menor, e os rejuvenescimentos sem que o sistema esteja em uma situação crítica são nulos. O tempo de inatividade do sistema, por decorrência de rejuvenescimento, irá variar segundo a configuração de  $\theta$ ,  $R$  e  $\delta_{RG}$ , quanto maiores forem estas taxas, o tempo de inatividade do sistema, devido a rejuvenescimento, será menor.

Tabela 10 – Comparação entre as abordagens de rejuvenescimento

	rejuvenescimento aleatório	rejuvenescimento <i>bucket</i>
Média de rejuvenescimentos	731,732	1,330
Média de rejuvenescimentos corretos	0,282	1,330
Média de rejuvenescimentos incorretos	731,450	0

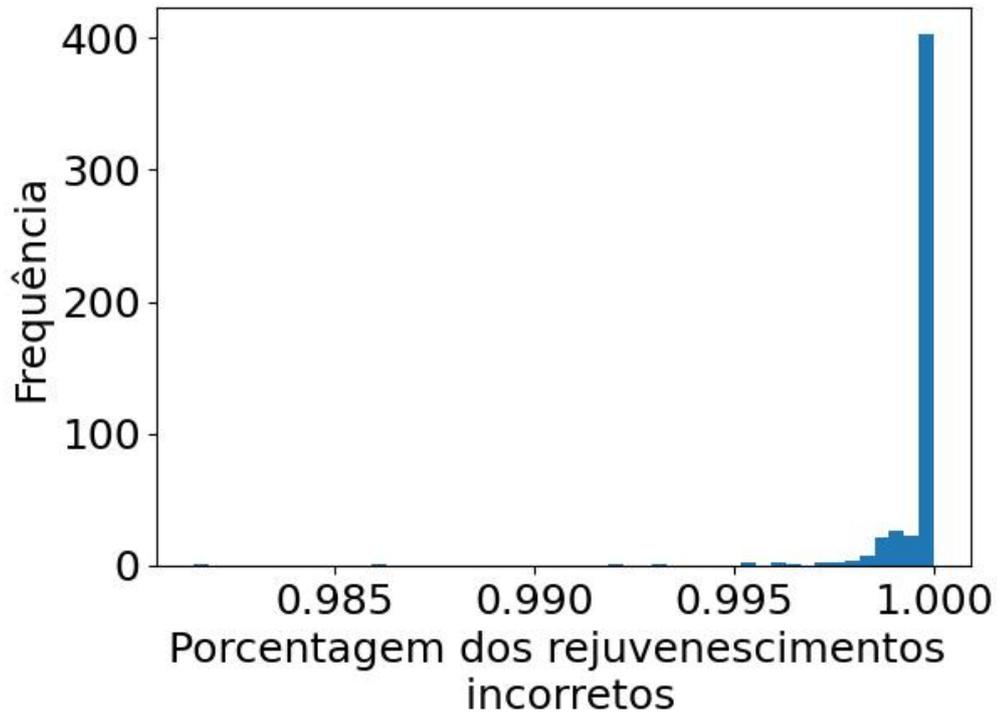


Figura 12 – Abordagem de aplicação do rejuvenescimento aleatório: porcentagem dos rejuvenescimentos aplicados sem que o limite do *bucket* e tenha sido atingido.

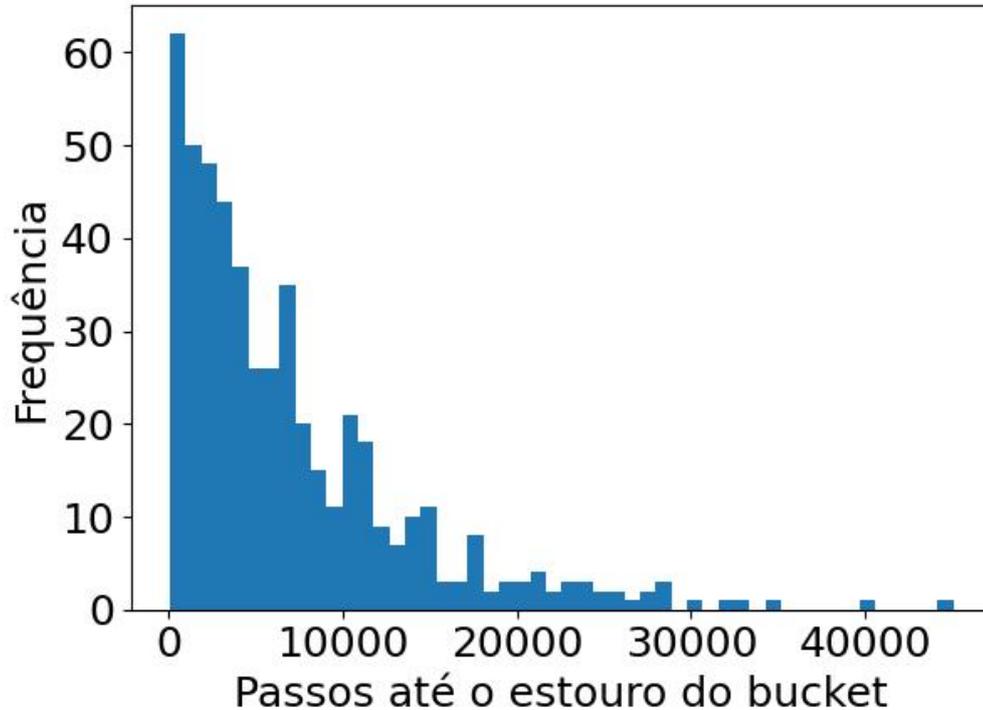


Figura 13 – Abordagem de aplicação do rejuvenescimento aleatório: quantidade de passos até se atingir o limite do *bucket*.

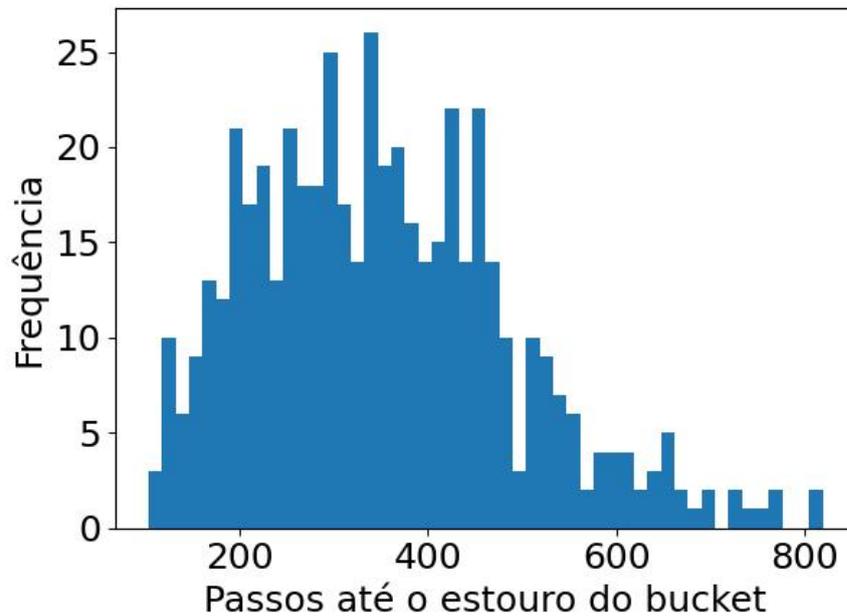


Figura 14 – Abordagem de aplicação do rejuvenescimento oriundo do estouro do *bucket*: quantidade de passos até se atingir o limite do *bucket*.

## 4.2 Contabilizando anomalias

Em seguida, apresentamos um modelo de Markov para capturar o impacto integrado do envelhecimento, recuperação natural, rejuvenescimento e ataques no comportamento do sistema. Em particular, estendemos o (HUANG et al., 1995), que corresponde à Figura 15(a), para considerar a recuperação natural e a possibilidade de anomalias no estado original, que corresponde à Figura 15(b).

**Questão 5:** *Como contabilizar anomalias, por exemplo, devido a ataques, ao analisar o papel do envelhecimento, rejuvenescimento e recuperação natural sob análise de desempenho sequencial?*

### 4.2.1 Descrição do modelo

Nosso modelo é obtido a partir da cadeia de Markov de tempo contínuo proposta pelo (HUANG et al., 1995), e ilustrado na Figura 15(a). Os estados 0, 1, *A* e *R* correspondem ao estado inicial, um estado de envelhecimento que é propenso a falhas, o estado anômalo onde ocorreu a falha e o estado de rejuvenescimento onde ocorre o rejuvenescimento, respectivamente. O envelhecimento ocorre na taxa  $r_2$ , e após atingir o estado 1 os processos de anomalia e rejuvenescimento competem entre si, ocorrendo com as taxas  $\lambda$  e  $r_4$ , respectivamente. Finalmente, uma vez que ocorre uma anomalia (resp., rejuvenescimento),

leva em média  $1/r_1$  (resp.,  $1/r_3$ ) para ser concluído.

Um dos principais objetivos é analisar o impacto das anomalias e rejuvenescimento na indisponibilidade do sistema. Para tanto, assume-se que a indisponibilidade do sistema é dada por  $U = \pi_A + \pi_R$  onde  $\pi_A$  e  $\pi_R$  são as probabilidades de estado estacionário dos estados  $A$  e  $R$ .

Primeiro, estendemos o modelo acima de duas maneiras, 1) adicionando a possibilidade de que o sistema se recupere naturalmente do envelhecimento, que ocorre com a taxa  $r'_2$ , e 2) permitindo que uma anomalia ocorra quando o sistema estiver no estado 0, que ocorre com a taxa  $r'_1$ . O modelo correspondente é mostrado na Figura 15(b).

Em segundo lugar, estendemos ainda mais o modelo permitindo vários estados de envelhecimento, que capturam o número de *tokens* no algoritmo de *bucket*. Embora possa não haver um mapeamento de um para um entre o número de *tokens* e o estado de envelhecimento, assumimos que o último é um *proxy* do primeiro. Nesse caso, os parâmetros de taxa são os mesmos descritos acima, mas o número de estados captura os vários níveis de envelhecimento, Figura 15(c).

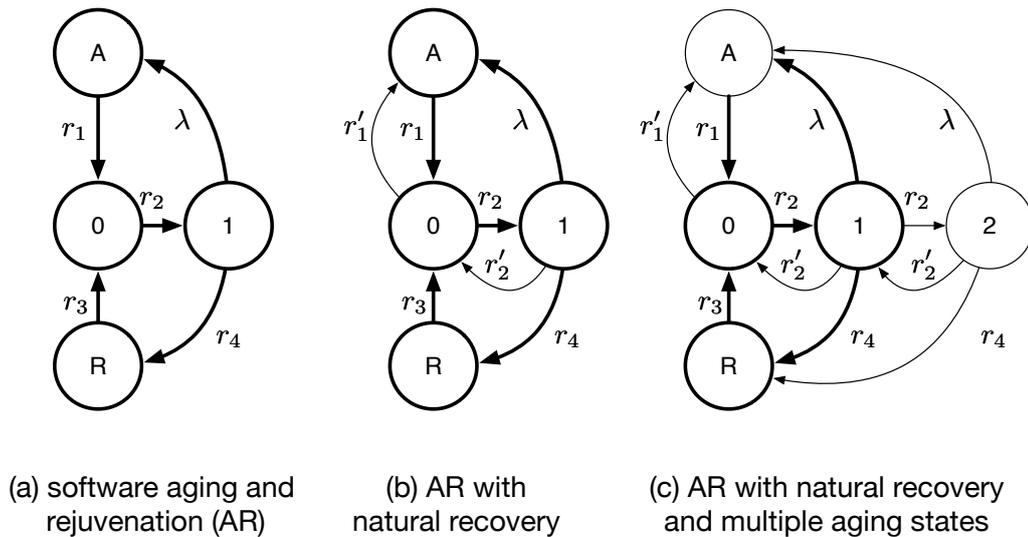


Figura 15 – Algoritmo de *bucket* com sobrecarga devido ao rejuvenescimento e anomalias

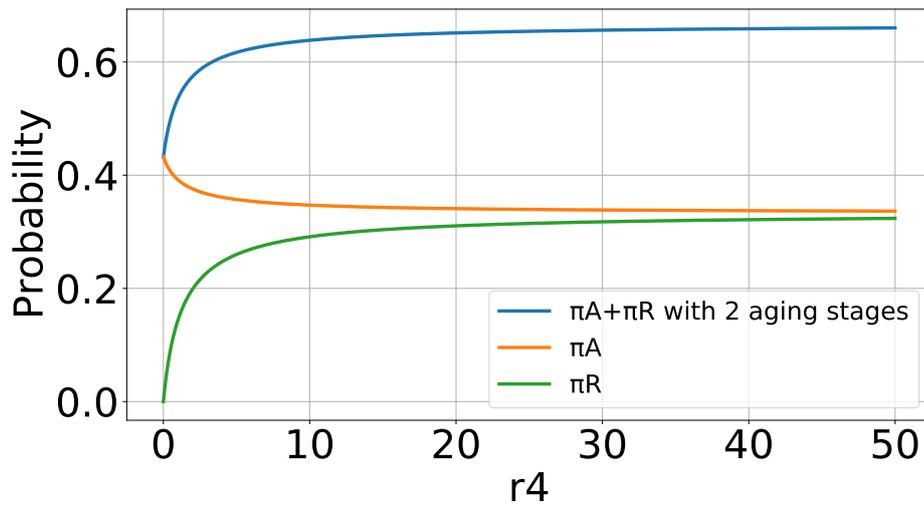


Figura 16 – Ilustrando um cenário em que o rejuvenescimento não é benéfico  $r_1 = 0,1, r'_1 = 0,1, r_2 = 0,1, r'_2 = 0,1, r_3 = 0,1, \lambda = 0,05$

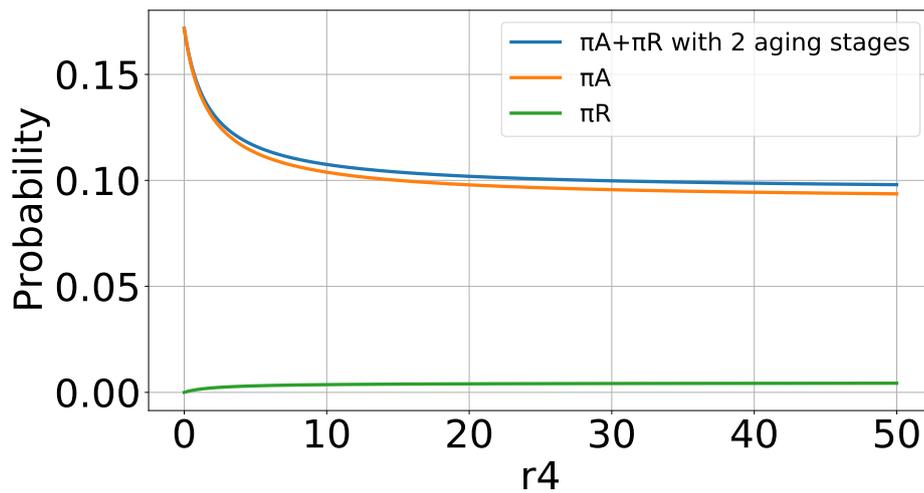


Figura 17 – Ilustrando um cenário em que o rejuvenescimento é benéfico diminuindo  $r'_1$  e  $\lambda$  para 0,01 e 0,03 e aumentando  $r_3$  para 20. Os resultados correspondem e ilustram a aplicabilidade de (4.24).

#### 4.2.2 Solução do modelo

Em seguida, resolvemos analiticamente os modelos da Figura 15(a) e Figura 15(b). O modelo na Figura 15(c) é avaliado numericamente na próxima seção e comparado com os dois primeiros.

A solução do modelo de Markov apresentada na Figura 15(b) é dada por:

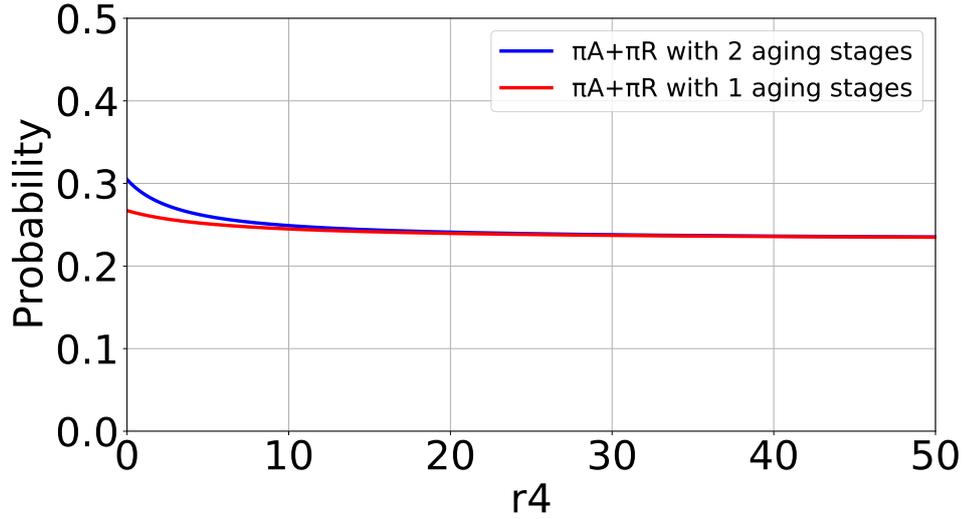


Figura 18 – Impacto do número de estados de envelhecimento.

$$\pi_0^{-1} = 1 + \frac{r_2}{r'_2 + r_4 + \lambda} \left( 1 + \frac{r_4}{r_3} + \frac{\lambda}{r_1} \right) + \frac{r'_1}{r_1} \quad (4.20)$$

$$\pi_1 = \pi_0 \frac{r_2}{r'_2 + r_4 + \lambda}, \quad \pi_R = \pi_0 \frac{r_2}{r'_2 + r_4 + \lambda} \frac{r_4}{r_3} \quad (4.21)$$

$$\pi_A = \pi_0 \left( \frac{r'_1}{r_1} + \frac{r_2}{r'_2 + r_4 + \lambda} \frac{\lambda}{r_1} \right) \quad (4.22)$$

Então, a indisponibilidade do sistema é dada por  $\pi_A + \pi_R$ ,

$$U = \frac{\frac{r'_2 + r_4 + \lambda}{r_2} \frac{r'_1}{r_1} + \frac{r_4}{r_3} + \frac{\lambda}{r_1}}{\frac{r'_2 + r_4 + \lambda}{r_2} \left( 1 + \frac{r'_1}{r_1} \right) + 1 + \frac{r_4}{r_3} + \frac{\lambda}{r_1}} \quad (4.23)$$

A derivada de  $U$  em relação a  $r_4$  é

$$\frac{dU}{dr_4} = \kappa \left( r_1 \left( 1 + \frac{r'_2 + r_2}{\lambda} \right) - r_3 \left( 1 - \frac{r'_1}{\lambda} \right) \right) \quad (4.24)$$

onde  $\kappa$  é uma constante positiva. A equação acima se reduz a

$$\frac{dU}{dr_4} = \kappa \left( r_1 \left( 1 + \frac{r_2}{\lambda} \right) - r_3 \right) \quad (4.25)$$

quando  $r'_2 = r'_1 = 0$  que corresponde à Figura 15(a).

Observe que se  $dU/dr_4 < 0$  o rejuvenescimento deve ser aplicado na taxa máxima. Caso contrário, o rejuvenescimento não deve ser aplicado.

### 4.3 Discussão

A seguir, comparamos o modelo de recuperação natural com o modelo original de envelhecimento e rejuvenescimento, discutimos o impacto dos diferentes parâmetros

do sistema na taxa de rejuvenescimento ideal  $r_4$ . De acordo com (4.25), à medida que  $r_1$  ou  $r_2$  aumentam, ou seja, à medida que se torna mais fácil resolver uma anomalia ou à medida que o sistema se torna mais propenso ao envelhecimento, o rejuvenescimento torna-se menos atraente, pois o sistema está conseguindo apresentar uma resposta adequada, neste caso o tempo que o sistema passa aplicando o rejuvenescimento não apresenta vantagem significativa. Deste modo, à medida que a taxa de ocorrência de anomalias,  $\lambda$  tende a aumentar, o rejuvenescimento se torna mais atraente, tendo em vista que o sistema apresentará uma maior dificuldade em se recuperar naturalmente, não conseguindo solucionar uma anomalia num tempo satisfatório. De forma análoga, se elevarmos a taxa de rejuvenescimento  $r_3$  passaremos a ter um tempo médio para executar um rejuvenescimento,  $1/r_3$ , menor, o que também passa a tornar o rejuvenescimento mais atraente.

Para estender a análise acima vamos entender o papel da recuperação natural, avaliando ainda mais o papel de  $r'_1$ , (taxa de um rejuvenescimento ocorrido do estado R para o estado 0, ou uma recuperação natural ocorrida do estado 1 para o estado 0 ser imediatamente seguido de uma anomalia) e  $r'_2$ , taxa de recuperação natural do sistema. De acordo com (4.24),  $r'_2$  que é a taxa de recuperação natural, desempenha um papel aditivo em relação à taxa de envelhecimento  $r_2$  ao determinar a atratividade do rejuvenescimento. De fato, se a recuperação natural já resolver a maioria dos efeitos do envelhecimento, o rejuvenescimento pode não ser necessário. Finalmente, um aumento em  $r'_1$  torna o rejuvenescimento menos atraente, pois aumenta a chance de um rejuvenescimento ocorrido do estado R para o estado 0 ou uma recuperação natural ocorrida do estado 1 para o estado 0 ser imediatamente seguido por uma nova anomalia.

**Avaliação numérica.** A seguir, ilustramos os pontos discutidos no parágrafo anterior, sobre o papel de diferentes parâmetros do sistema na taxa ótima de rejuvenescimento. Para isso, avaliamos numericamente a cadeia de Markov na Figura 15(c) aplicando as seguintes taxas de transições  $r_1 = 0,1, r'_1 = 0,1, r_2 = 0,1, r'_2 = 0,1, r_3 = 0,1$  e  $\lambda = 0,05$ , neste modelo, procuramos trabalhar com as mesmas taxas para o envelhecimento e recuperação natural e de execução do rejuvenescimento, a única taxa diferente foi a de anomalia 0,05 e  $r_4$  variou ao longo do tempo. Para esses parâmetros, observamos na Figura 16 que  $U$  aumenta com  $r_4$  de acordo com (4.24), em que pese  $\pi_A$  apresentar uma queda, o tempo de indisponibilidade do sistema, por ocasião do rejuvenescimento  $\pi_R$ , se eleva. A soma dos dois resulta em uma indisponibilidade do sistema acima de 60% conforme  $r_4$  varia no tempo.

Em um novo cenário procuramos aumentar  $r_3$ , taxa de execução do rejuvenescimento, para observarmos os benefícios do rejuvenescimento para diminuir a indisponibilidade do sistema, trabalhamos com uma taxa para  $r_3$  de 20. De forma paralela, reduzimos  $r'_1$ , taxa de um rejuvenescimento ocorrida do estado R para o estado 0, ou uma recuperação natural ocorrida do estado 1 para o estado 0 ser seguida imediatamente de uma anomalia,

para 0,01 e  $\lambda$ , taxa de ocorrência de anomalias, para 0,03. Todas essas mudanças favoreceram os benefícios do rejuvenescimento,  $\pi_A$  apresentou uma queda que com o decorrer de  $r_4$  ficou abaixo de 10%, enquanto o  $\pi_R$  se manteve baixo. Neste cenário, a indisponibilidade do sistema ficou abaixo dos 10% com o decorrer de  $r_4$ , conforme indicado na Figura 17, novamente de acordo com (4.24).

**Contabilização de vários estágios de envelhecimento.** Nos cenários das Figuras 16 e 17, podemos demonstrar que o uso de um ou dois estágios de envelhecimento teve um impacto insignificante nas probabilidades de interesse. A seguir, ilustramos um cenário em que o número de estágios de envelhecimento impacta nossas métricas de interesse. Para ilustrar o impacto do número de estágios de envelhecimento, aplicamos as seguintes taxas de transições  $r_1 = 0,1, r'_1 = 0,1, r_2 = 4, r'_2 = 2, r_3 = 20$ , e  $\lambda = 0.05$ . Isso corresponde ao último cenário como linha base e aumentando a taxa de envelhecimento,  $r_2$ , e a taxa de recuperação natural,  $r'_2$ . A Figura 18 mostra a indisponibilidade do sistema, em função de  $r_4$ . Com 2 estágios de envelhecimento, a indisponibilidade para  $r_4 = 0$  é maior do que para 1 estágio de envelhecimento: a taxa de falhas em estados diferentes de 0 é maior que em 0, e o sistema com 1 estágio de envelhecimento visita o estado 0 com mais frequência do que o sistema com 2 estágios de envelhecimento. À medida que  $r_4$  cresce, a indisponibilidade converge para um valor assintótico que não depende do número de estágios, dado por  $3/13 \approx 0,23$  (ver (4.23)).

# 5 TRABALHOS RELACIONADOS

## 5.1 Envelhecimento e rejuvenescimento

### 5.1.1 Relação entre envelhecimento e esgotamento de recursos

A literatura na área de envelhecimento e rejuvenescimento de software é vasta, remontando a década de 90 (HUANG et al., 1995). O envelhecimento de software foi descrito por (COTRONEO et al., 2011) como o acúmulo de erros que ocorrem em sistemas operacionais de software de longa duração que levam ao esgotamento progressivo de recursos, degradação do desempenho e, eventualmente, ao travamento ou falha. Segundo (AVRITZER et al., 2012) a degradação de desempenho do sistema é chamada de envelhecimento do software, que ocorre pelo esgotamento dos recursos como memória do sistema ou estruturas do kernel, ponteiros inválidos, acúmulo de erros de arredondamento, *deadlocks* de banco de dados e a disputa por um pool de software limitado. Portanto, falhas transientes de aplicativos e do sistema operacional podem ser uma das principais fontes de degradação do desempenho do sistema.

Exemplos de abordagens de rejuvenescimento de software aplicam reinicialização de processos, liberação de memória, a eliminação de um impasse ou a execução de qualquer outra ação rápida que impeça que o envelhecimento do software se manifeste como uma falha em todo o sistema, podendo levar a uma falha generalizada do sistema. Essas falhas em todo o sistema podem causar danos significativos à missão que o software se destina e, conseqüentemente, à infraestrutura que está sendo usada para dar suporte ao sistema de software. Por exemplo, um banco de dados corrompido pode levar um tempo significativo para se recuperar.

Em (AVRITZER; WEYUKER, 1997) é apresentado um problema comum para grandes sistemas de software em geral e sistemas de telecomunicações que é a degradação gradativa culminando em falhas, em que pese o fato de haver disponibilidade de recursos. Estas falhas começam a afetar o desempenho do sistema levando à perda de pacotes e usuários. O autor caracteriza esta situação em que a capacidade utilizável de um sistema de software é monotonicamente não crescente como um sistema de degradação suave. A degradação, neste estágio, é considerada normal devido ao vazamento de recursos como memória, por exemplo, fato este que ocorre em qualquer sistema. Os desafios, neste cenário, são determinar quais dados coletar e como determinar os tempos apropriados para restaurar um sistema degradado à capacidade total para minimizar a interrupção e o custo do sistema e do cliente, antes que o usuário perceba uma falha, degradação na disponibilidade ou desempenho inadequado do sistema.

Em (BOVENZI COTRONEO; RUSSO, 2012) são investigados os efeitos de envelhecimento de software causados pela ativação de *bugs* de simultaneidade em um conhecido sistema de gerenciamento de banco de dados (SGBD), o MySQL. Experimentos com diferentes cargas de trabalho foram realizados para reproduzir as condições prováveis para ativação de *bugs* de simultaneidade. Além dos efeitos típicos de envelhecimento observados em muitos sistemas operacionais (ou seja, uma degradação gradual ao longo do tempo), os resultados destacam que tanto os recursos disponíveis quanto o desempenho do SGBD (por exemplo, taxa de serviço, tempo de serviço e latência de conexão) estão sujeitos à diminuição com o tempo em um ambiente de difícil acesso. Foi observado que, devido à ativação do *bug* de simultaneidade, o SGBD entrou em um estado de degradação no qual: i) a estimativa do *Time-To-Failure* (TTF) por meio da análise de tendência de depleção de memória se mostrou altamente imprecisa, e ii) a taxa de falha não dependia do trabalho acumulado instantâneo e/ou médio. Os resultados sugeriram que, nesses casos, indicadores mais refinados e/ou técnicas diferentes precisariam ser consideradas para prevenir falhas adequadamente.

### 5.1.2 Esgotamento de recursos em sistemas críticos

Zhao et al. (ZHAO, 2010) caracterizam em sua pesquisa o envelhecimento do software como um fenômeno de degradação progressiva do desempenho do software em execução, o que pode levar a falhas no sistema ou travamentos indesejados, podendo acontecer devido ao esgotamento dos recursos do sistema. Esse fenômeno indesejado existe amplamente não apenas em softwares geralmente usados, como servidores da *Web* e de aplicativos, mas também em sistemas de aplicativos críticos que exigem alta confiabilidade/disponibilidade. O envelhecimento do software pode causar grandes perdas nos sistemas críticos de segurança, incluindo a perda de vidas humanas, este fato não faz com que o software falhe imediatamente uma vez iniciado, mas, em vez disso, leva à degradação progressiva do desempenho do sistema até ocasionar a parada do sistema por completo.

Em (ARAUJO et al., 2011) é abordada a necessidade de confiabilidade e disponibilidade em aplicativos modernos, a fim de lidar com demandas em rápido crescimento enquanto fornece um serviço ininterrupto. Os sistemas de computação em nuvem fornecem fundamentalmente acesso a grandes conjuntos de dados e recursos computacionais por meio de uma variedade de interfaces de maneira semelhante aos sistemas de programação e gerenciamento de recursos de grade e HPC existentes.

### 5.1.3 Rejuvenescimento

Uma contra medida apresentada e amplamente pesquisada é a aplicação do rejuvenescimento do sistema. O rejuvenescimento é implementado em vários sistemas, como

sistemas de coleta de dados de faturamento, sistemas de telecomunicações, sistemas de processamento de transações e sistemas de naves espaciais. Esta medida visa aplicar o encerramento ocasional de um aplicativo ou sistema, limpeza de seu estado interno e reiniciá-lo para liberar recursos do sistema, para que o desempenho do software seja recuperado.

Com base na taxa de degradação do software, e considerando a relação entre desempenho e degradação do software, Avritzer et al. (AVRITZER, 2007) procuram encontrar o melhor intervalo de tempo quanto ao acionamento do rejuvenescimento durante o processo de execução do software, visando reduzir os custos de manutenção. (ARAUJO et al., 2011) utiliza séries temporais para programar o rejuvenescimento, para reduzir o tempo de inatividade, prevendo o momento adequado para realizar o rejuvenescimento. Os autores reportam resultados da abordagem proposta por meio de experimentos, usando o *framework* de computação em nuvem *Eucalyptus*.

Em (WANG; LIU, 2020), foi proposto um novo método de decisão de rejuvenescimento de software em tempo real, denominado HARRD, onde a distribuição Weibull clássica no campo da análise de confiabilidade (uma distribuição de probabilidade contínua) é utilizada para simular e modelar o processo de transição de estado do envelhecimento de software. Então, com base nesse modelo com uso de recursos em tempo real de parâmetros de monitoramento de hardware e, juntos, integrando três indicadores de modelo, foi construído a função de decisão de rejuvenescimento usando o processo de hierarquia analítica (AHP) para ponderar os parâmetros acima, que poderiam ser finalmente usados como base de decisão de rejuvenescimento para sistemas de software envelhecidos. O método de decisão de rejuvenescimento pode equilibrar os fatores imprevisíveis no processo de envelhecimento de software usando modelos de simulação precisos e considerar mais indicadores para decisão de tempo de rejuvenescimento. Os resultados experimentais mostraram que o sistema de software baseado no método proposto pode alcançar melhores efeitos de rejuvenescimento do software em termos de desempenho de consumo de tempo, velocidade média de processamento de tarefas e estabilidade do sistema.

Em (LI et al., 2020) o fato do rejuvenescimento incorrer em sobrecarga extra no sistema e custo de tempo de inatividade é citado. Neste sentido, era importante determinar o momento ideal para acionar o rejuvenescimento. Um modelo de duas camadas foi proposto para caracterizar o comportamento de falha do sistema e o processo de rejuvenescimento sob condição de envelhecimento, planejando uma política de rejuvenescimento baseada no tempo, maximizando a disponibilidade do sistema e minimizando o custo do tempo de inatividade. A camada inferior consiste num modelo de falha. A camada superior consiste num modelo de rejuvenescimento que usa a distribuição de falhas da camada inferior como entrada para formular a função de disponibilidade e a função de custo de tempo de inatividade. Tomando essas duas funções como alvos de otimização, foi possível obter o

tempo ideal de rejuvenescimento. Comparando com o modelo analítico tradicional, a falha do software de modelagem de modelo de duas camadas considerou medições de tempo de execução, que podiam descrever o comportamento de envelhecimento com mais precisão. Na parte experimental, foi avaliado de forma abrangente o modelo de duas camadas estudando o envelhecimento do sistema de busca na web. Os resultados mostraram que o modelo de duas camadas reduz a indisponibilidade em 18,18% e reduz o custo de downtime em 31,22% em comparação com o modelo analítico tradicional.

Romagnoli (ROMAGNOLI; KROGH; SINOPOLI, 2019) considera o problema de determinar a taxa ideal de rejuvenescimento, contabilizando seus benefícios em termos de maior segurança e proteção. É natural supor que nesses sistemas um teste de hipótese sequencial será executado continuamente para detectar anomalias. Então, de certa forma, a presente pesquisa estende este conceito, pois serve para lançar *insights* sobre os impactos do rejuvenescimento na taxa de falso positivos.

**Nosso trabalho.** No contexto desta pesquisa, desenvolvemos um modelo que busca lidar com situações de esgotamento de recursos do sistema e a possíveis situações de ataques. O modelo engloba o monitoramento do estado do sistema. Se este apresenta uma não conformidade com o esperado, uma ação de reparo/ajuste (rejuvenescimento) é sinalizada, no intuito de restabelecer o funcionamento esperado do sistema.

## 5.2 Processos de nascimento e morte

### 5.2.1 Processos de nascimento e morte clássicos

A seguir, indicamos a relação entre processos de nascimento e morte e os modelos apresentados neste trabalho. Discussões adicionais sobre processos de nascimento e morte encontram-se no Apêndice A.

**Nosso trabalho.** No contexto desta pesquisa, os processos de nascimento e morte surgem em pelo menos três esferas. Primeiro, para capturar o estado da fonte de anomalias, usamos uma MMPP, que pode ser modelada como um processo de nascimento e morte do tipo on-off. Segundo, o estado da fila de gargalo também pode ser capturado como uma M/M/1/K, em que a taxa de nascimento é  $\lambda$  a taxa de morte é  $\mu$ . O rejuvenescimento restabelece o sistema a sua condição inicial com taxa  $\beta$ . Neste âmbito, o rejuvenescimento está associado a catástrofe, quando ocorre um declínio súbito nos indicadores do sistema, re-estabelecendo sua condição inicial, ou seja, trazendo o estado da fonte para  $G$  (*Good*), número de *jobs* na fila gargalo para 0 e número de *tokens* no *bucket* também para 0. Terceiro, o estado do algoritmo de *bucket* também pode ser caracterizado como um processo de nascimento e morte, conforme discutido em (GONÇALVES et al., 2020).

## 5.2.2 Processos de nascimento e morte com catástrofes

A relação entre rejuvenescimento e catástrofes, no âmbito das *G-Networks*, foi estudada em (FOURNEAU; MAJHOUB, 2017). No âmbito desta pesquisa, aproveitamos a relação entre catástrofes e rejuvenescimento, já citada. Entretanto, trabalhamos no âmbito dos processos de nascimento-morte em oposição às *G-Networks*.

A contabilização de eventos catastróficos tornou-se uma parte importante da modelagem populacional estocástica, particularmente em ecologia, mas também em uma série de outros campos, incluindo economia, química e telecomunicações. No contexto dos processos populacionais, as catástrofes são declínios súbitos na população, fato este que se torna fundamental no tempo de extinção de uma população (CAIRNS; POLLETT, 2004). Na literatura de Física, catástrofes são chamadas de *resets* (MEYLAHN; SABHAPANDIT; TOUCHETTE, 2015). Em (GROTTKE et al., 2016; ALTMAN et al., 2014) foi considerado o rejuvenescimento após anomalias, representando o envelhecimento.

**Nosso trabalho.** Neste trabalho, estendemos (GROTTKE et al., 2016; ALTMAN et al., 2014) para considerar não apenas o rejuvenescimento (DOHI; TRIVEDI; AVRITZER, 2020), mas também a recuperação do sistema natural, dando origem a um modelo de nascimento-morte com catástrofes que ocorrem quando o rejuvenescimento é aplicado e o sistema recupera o seu estado original. Note que no contexto deste trabalho as catástrofes correspondem ao rejuvenescimento, portanto a um acontecimento com uma conotação positiva.

## 5.3 Monitoramento

### 5.3.1 Análise de desempenho sequencial

Os sistemas de software usados para dar suporte a sistemas de missão crítica de alta confiabilidade são geralmente monitorados quanto a defeitos de software e desempenho. Além disso, várias camadas de defesas de segurança são comumente implementadas para proteger esses sistemas contra invasões de usuários não autorizados (AVRITZER et al., 2010).

Avritzer et al. (AVRITZER, 2013) apresentam resultados empíricos que mostram que a intrusão de segurança de software pode deixar o software em um estado ainda operacional, entretanto com a capacidade disponível do sistema reduzida. Essa condição às vezes é chamada de falha suave. CUSUM e o algoritmo de *bucket* foram estudados sob o domínio geral dos algoritmos de análise de desempenho sequencial.

**Nosso trabalho.** Em trabalhos anteriores, assumiu-se que os sistemas onde esses algoritmos de análise de desempenho sequencial são implantados ocorre uma recuperação natural da degradação transiente de desempenho apresentada. O objetivo agora é tam-

bém considerar o impacto das intervenções de rejuvenescimento na dinâmica da análise sequencial do desempenho.

### 5.3.2 O algoritmo de *bucket*

O algoritmo de *bucket* foi proposto como um elemento importante no gerenciamento baseado em taxa de largura de banda em redes de comunicação de pacotes integradas de alto desempenho por (HLUCHYJ, 1993). Neste artigo, o algoritmo de *bucket* foi analisado para fontes de dados *ON-OFF*, que podem ser caracterizadas por sua taxa média de dados, taxa de dados de pico e número médio de bits gerados durante um período *ON*. Uma fila fictícia foi usada para modelar o comportamento do algoritmo de *bucket* algoritmo e um processo de taxa modulada por Markov de dois estados é usado para modelar a fonte de dados *ON-OFF*. A análise se aplica a *buckets* com vazamento em buffer e sem buffer com rejeição ou marcação de dados violados. São derivadas expressões simples e fechadas que relacionam os parâmetros do *bucket* com vazamento (tamanho e taxa do *bucket*) e as características da fonte *ON-OFF* às probabilidades de perda e marcação e o atraso de fila do *bucket* com vazamento. Mostra-se que o tamanho do *bucket* necessário aumenta linearmente com o número médio de *bits* gerados durante um período *ON* e aumenta logaritmicamente com a diminuição da probabilidade de perda ou marca. Para um *bucket* com vazamento em *buffer*, também mostramos que o tamanho do *bucket* aumenta logaritmicamente com a diminuição do atraso médio na fila de *bucket* com vazamento.

**Nosso trabalho.** Neste ponto cabe uma diferenciação fundamental entre o proposto no artigo citado e o que está sendo abordado neste trabalho, em que pese ambos terem o objetivo de controle de tráfego, eles abordam esta questão de forma distinta o *bucket* com vazamento faz controle de congestionamento assumindo um tráfego cooperativo, já o *bucket* algoritmo faz o controle de admissão assumindo tráfego anômalo. Além disso, o *bucket* com vazamento consome *tokens* a medida que o tráfego é transmitido e quando o número de *tokens* chega a zero o tráfego é bloqueado, já o algoritmo de *bucket* espera o número de *tokens* chegar a um limiar superior e aciona um alarme sinalizando uma situação crítica do sistema.

### 5.3.3 Aplicações do algoritmo de *bucket*

Em (AVRITZER; BONDI; WEYUKER, 2005) e (AVRITZER, 2011) os autores utilizam propõem uma abordagem para identificar e eliminar a degradação de desempenho que ocorre em softwares antigos. Uma métrica que afeta o cliente é usada para iniciar a restauração de tal sistema para a capacidade total. É descrito um estudo de caso no qual, simulando um sistema de software industrial, podemos mostrar que monitorando uma métrica afetada pelo cliente e comparando frequentemente sua degradação visando desempenho, podemos garantir a estabilidade do sistema a um custo muito baixo.

(AVRITZER; COLE; WEYUKER, 2007) sugere uma nova abordagem para mitigar a propagação de **worms** por meio de redes móveis *ad-hoc* táticas (MANETs) que se baseia em assinaturas de desempenho e rejuvenescimento de software. Três algoritmos de assinatura de desempenho de aplicativos e rejuvenescimento de software são propostos e analisados. Esses algoritmos monitoram a capacidade de resposta dos aplicativos críticos e acionam ações para rejuvenescimento do software quando os recursos do host se degradam devido a um *worm* co-residente competindo pelos recursos do host. É analisada a eficácia de nossos algoritmos por meio de modelagem analítica e estudos detalhados e extensos de simulação. As principais métricas de desempenho investigadas são o tempo de resposta do aplicativo, o tempo médio entre rejuvenescimentos e a probabilidade de estado estável de infecção do host. Também usamos modelos de simulação para investigar vários problemas de design e ajuste de parâmetros. É investigada a relação entre a taxa na qual os monitores de desempenho de aplicativos podem detectar aplicativos fora de especificação e a taxa de propagação de *worms* na rede.

Foi construído um novo modelo dinâmico para o algoritmo *Token Bucket* (TB) utilizado em redes de computadores e utilizando a abordagem de sistemas para sua análise em (AHMED; WANG; OROZCO-BARBOSA, 2002). Este modelo é então aumentado pela adição de um modelo dinâmico para um multiplexador em um nó de acesso onde o TB exerce uma função de policiamento. No modelo, o policiamento de tráfego, a multiplexação e a utilização da rede são formalmente definidos. Com base no modelo, foram estudadas questões como QoS (qualidade de serviço) (AVRITZER, 2013), dimensionamento de tráfego e dimensionamento de rede. Também foi proposto um algoritmo usando controle *feedback* para melhorar a QoS e a utilização da rede. Foi aplicado rastreamentos de vídeo MPEG como tráfego de entrada para o modelo, verificando a utilidade e eficácia do modelo proposto.

### 5.3.4 CUSUM

Em (GUEPIE; FILLATRE; NIKIFOROV, 2012) um algoritmo de detecção de mudança transiente sequencial sub-ótimo é proposto. Ele é baseado em um teste de soma cumulativa limitada por janela (CUSUM). Parte-se do princípio que uma mudança ocorre em um ponto de mudança desconhecido (mas não aleatório) e existe uma duração do período pós-mudança finita e conhecida. Uma detecção latente - ou seja, uma detecção que ocorre após o desaparecimento do sinal - é considerada uma detecção perdida. Um novo critério de otimização adaptado à detecção de mudanças transitórias envolve a minimização da probabilidade do pior caso de detecção perdida sob restrição da taxa de falsos alarmes por um determinado período. São propostos um limite superior para a probabilidade do pior caso de detecção perdida e um limite inferior e superior para a taxa de falsos alarmes. Com base nesses limites, o teste CUSUM limitado por janela é otimizado em relação

ao critério proposto. O algoritmo desenvolvido e os achados teóricos são aplicados ao monitoramento da rede de distribuição de água potável.

### 5.3.5 Controle de processos

O termo “controle de processos” costuma ser utilizado para se referir a sistemas que têm por objetivo manter certas variáveis entre os seus limites operacionais desejáveis. Estes sistemas de controle podem necessitar constantemente da intervenção humana, ou serem automatizados. (CAMPOS; TEIXEIRA, 2010) destaca a importância da melhoria contínua nas áreas tecnológicas de controle, automação e otimização de processos. O autor destaca os vários ganhos da aplicação destas tecnologias nos processos como: aumento da confiabilidade dos sistemas, do nível de segurança da unidade e do nível de qualidade. Uma abordagem destacada é medir uma variável de interesse e importante para o processo e implementar um controle automático em malha fechada. Caso essa variável apresente comportamento indesejado, o controlador emite uma ação de correção no intuito de restabelecer a normalidade do sistema.

**Controlador.** Com o sistema de controle em malha fechada surge a figura do “controlador”, que compara o valor desejado com o valor medido, e se houver um desvio entre estes valores, manipula a sua saída para eliminar este desvio ou erro. Desta maneira, o controle em malha fechada mantém a variável do processo no seu valor desejado, compensando as perturbações externas e as possíveis não linearidades do sistema. Os campos de atuações para um controle de malha fechada são muitos, (SANTOS et al., 2015) apresenta os desafios da implementação de um sistema de controle de malha fechada na comunicação sem fio. Neste sentido, é apresentada uma avaliação do comportamento de uma rede *WirelessHART* em processos de controle de nível de água em um sistema de tanques acoplados. (CASADO-VARA et al., 2019) introduz um novo sistema adaptativo de controle de malha fechada e um modelo de busca acelerada para melhorar a eficiência de monitoramento e controle em redes IoT, especialmente aquelas baseadas em blockchain. O modelo de controle não linear em consideração, inclui uma nova maneira de avaliar o número ótimo de blocos que devem estar na fila da rede dos mineradores para tornar o processo eficiente por meio do uso da teoria das filas.

**Nosso trabalho.** Neste trabalho, procuramos unificar o monitoramento *online* a um controle de malha fechada. O controle do estado do sistema será feito pelo algoritmo de *bucket*, que sinalizará uma criticidade no sistema. Sempre que o sistema não apresentar um comportamento esperado pelo administrador, um *tokens* será adicionado ao *bucket*, caso contrário, um *tokens* é retirado do *bucket*. Assim que o *bucket* ultrapassar o limite definido pelo administrador, um sinal de rejuvenescimento é enviado ao sistema e a sua condição inicial é estabelecida, implementando assim um sistema de controle de malha fechada.

## 6 CONCLUSÃO

Neste trabalho, apresentamos modelos de envelhecimento e rejuvenescimento considerando a recuperação natural, presente em virtualmente todos os sistemas reais. Inicialmente, no Capítulo 4, consideramos a abordagem de aplicação aleatória do rejuvenescimento no domínio da análise de desempenho sequencial, na ausência de anomalias, e calculamos o tempo médio até um falso positivo. Podemos observar que a aplicação de rejuvenescimento propicia uma resiliência maior ao sistema (o sistema leva um tempo maior até acionar um falso positivo). Entretanto, devido ao rejuvenescimento ser aleatório e sem relação direta com o estado do sistema, ocorrem muitos rejuvenescimentos desnecessários, quando o sistema ainda possui resiliência para se recuperar naturalmente de uma degradação transientes, fato este que leva o sistema a uma indisponibilidade desnecessária.

Ainda no Capítulo 4, na presença de anomalias, computamos a indisponibilidade do sistema, que no nosso modelo ocorre quando o sistema está no estado  $A$  e no estado  $R$ , contabilizando envelhecimento, recuperação natural e rejuvenescimento. Notamos, que em determinados cenários o rejuvenescimento não se mostra tão vantajoso, mais precisamente quando o sistema consegue tratar bem as degradações transientes ocorridas ao longo do tempo de vida do sistema, por meio de recuperação natural. Em um cenário discutido previamente, o tempo de execução do rejuvenescimento foi fundamental para um crescimento da indisponibilidade do sistema. Entretanto, quando há um ajuste das diversas taxas do sistema e uma elevação da taxa de execução do rejuvenescimento, podemos observar uma queda significativa na indisponibilidade do sistema, demonstrando o benefício da aplicação do rejuvenescimento. Podemos concluir que o rejuvenescimento tem um papel fundamental no tocante a indisponibilidade do sistema, entretanto uma combinação adequada das taxas restantes se faz necessária, caso queiramos potencializar seus benefícios para a disponibilidade de um sistema.

No Capítulo 3 implementamos uma nova abordagem integrando o processo MMPP e o *bucket*, junto com uma fila gargalo  $M/M/1$ , dando origem a um sistema com envelhecimento e rejuvenescimento simulado via controle de malha fechada. Com isso os rejuvenescimentos passam a ser oriundos do estouro do *bucket*, sinalizando que o sistema está em situação crítica e não consegue se recuperar sozinho, necessitando de intervenção do administrador. Considerando a abordagem de aplicação do rejuvenescimento advinda do estouro do *bucket*, sinalizando uma situação em que o sistema não consegue se recuperar naturalmente, percebemos uma otimização do número de rejuvenescimentos aplicados com maior assertividade, em comparação com a abordagem do rejuvenescimento aleatório. No que diz respeito ao tempo médio de permanência no estado  $A$ , podemos verificar

que uma estratégia adequada de rejuvenescimento e de transição de estados contribui para uma permanência menor no estado  $A$ . A aplicação do rejuvenescimento apenas ao estado anômalo se mostrou eficiente à medida que adotamos uma estratégia adequada de rejuvenescimento e manipulamos as taxas de transição no que diz respeito em especial a uma permanência menor no estado  $A$  em relação ao estado  $G$  e o número médio de *jobs* não atendidos pelo sistema.

Os casos em que simulamos  $\lambda_A > \mu$  evidenciaram que o sistema pode chegar ao seu ponto crítico mais rapidamente. Neste sentido, surgem alternativas para tentar mitigar o problema: 1) tentar investigar a causa do aumento de  $\lambda$  no sistema, o que pode envolver custos, ou 2) aumentar a capacidade de resposta do sistema de forma que tenhamos  $\mu > \lambda$ . Esta última alternativa envolve altos custos para aumentar a capacidade de pico do sistema.

## 6.1 Trabalhos futuros

### 6.1.1 Validação realista

De forma continuada, seria oportuno analisarmos mais atentamente o comportamento de um sistema simulado em que no contexto de anomalias recebe uma carga de trabalho em *batch* como abordado (MARIN et al., 2022), os múltiplos *background packets* corresponderiam a pacotes ruins injetados por invasores. Em uma primeira análise estaríamos submetendo o sistema a uma carga superior de trabalho, o objetivo é entender o funcionamento do sistema nesta situação anormal a despeito dos conceitos de envelhecimento, recuperação natural e rejuvenescimento. As múltiplas chegadas simultâneas seguiriam um processo MMPP (HARCHOL-BALTER, 2013) (BANKS et al., 2012) onde dois estados seriam considerados, a saber:  $G$ ,  $A$ , correspondendo aos estados bom e anômalo, respectivamente. Todo o processo seria monitorado pela análise de desempenho sequencial (algoritmo de *bucket*).

### 6.1.2 Rejuvenescimento e trabalho cooperativo em rede

Analisar o conceito do MMPP e *bucket* M/M/1 com envelhecimento e rejuvenescimento simulando um controle de malha fechada aplicado a um ambiente de redes de dados que estão sujeitos a envelhecimento e recuperação natural, representando um ambiente *multithread* cooperativo (HOMCHAUDHURI, 2020; SUNDARAM et al., 2007) (MARIN; BALSAMO; FOURNEAU, 2017) de forma escalável e estendendo redes de formulário de produto para esse assunto (BALSAMO; HARRISON; MARIN, 2010; CHAO, 1995; ARTALEJO, 2000).

# REFERÊNCIAS

ACHARYA, S. K.; VILLARREAL-RODRÍGUEZ, C. E. et al. Change point estimation of service rate in an M/M/1/m queue. *Int. J. Math. Oper. Res.*, v. 5, n. 1, p. 110–120, 2013. Citado na página 20.

AHMED, N.; WANG, Q.; OROZCO-BARBOSA, L. Systems approach to modeling the token bucket algorithm in computer networks. *Mathematical Problems in Engineering*, v. 8, 06 2002. Citado na página 67.

ALTMAN, E. et al. Rejuvenation and the spread of epidemics in general topologies. In: IEEE. *2014 IEEE International Symposium on Software Reliability Engineering Workshops*. [S.l.], 2014. p. 414–419. Citado na página 65.

ANDRÉOLETTI, J. et al. The Occurrence Birth–Death Process for Combined-Evidence Analysis in Macroevolution and Epidemiology. *Systematic Biology*, 05 2022. ISSN 1063-5157. Syac037. Disponível em: <https://doi.org/10.1093/sysbio/syac037>. Citado 2 vezes nas páginas 20 e 77.

ARAÚJO, J. et al. Software rejuvenation in eucalyptus cloud computing infrastructure: A method based on time series forecasting and multiple thresholds. In: . [S.l.: s.n.], 2011. Citado 2 vezes nas páginas 62 e 63.

ARTALEJO, J. R. G-networks: A versatile approach for work removal in queueing networks. *European Journal of Operational Research*, Elsevier, v. 126, n. 2, p. 233–249, 2000. Citado na página 70.

Alberto Avritzer. *System and method for multivariate quality-of-service aware dynamic software rejuvenation*. 2013. Citado 2 vezes nas páginas 65 e 67.

AVRITZER, A.; BONDI, A.; WEYUKER, E. J. Ensuring stable performance for systems that degrade. In: *Proceedings of the 5th International Workshop on Software and Performance*. New York, NY, USA: Association for Computing Machinery, 2005. (WOSP '05), p. 43–51. ISBN 1595930876. Disponível em: <https://doi.org/10.1145/1071021.1071026>. Citado na página 66.

AVRITZER, A.; COLE, R. G.; WEYUKER, E. J. Using performance signatures and software rejuvenation for worm mitigation in tactical manets. In: *Proceedings of the 6th International Workshop on Software and Performance*. New York, NY, USA: Association for Computing Machinery, 2007. (WOSP '07), p. 172–180. ISBN 1595932976. Disponível em: <https://doi.org/10.1145/1216993.1217023>. Citado na página 67.

AVRITZER, A. et al. Software aging and rejuvenation for increased resilience: Modeling, analysis and applications. In: \_\_\_\_\_. *Resilience Assessment and Evaluation of Computing Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 167–183. ISBN 978-3-642-29032-9. Disponível em: [https://doi.org/10.1007/978-3-642-29032-9\\_8](https://doi.org/10.1007/978-3-642-29032-9_8). Citado na página 61.

- AVRITZER, A. et al. Monitoring for security intrusion using performance signatures. In: *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*. [S.l.: s.n.], 2010. p. 93–104. Citado na página 65.
- AVRITZER, A.; WEYUKER, E. Monitoring smoothly degrading systems for increased dependability. *Empirical Software Engineering*, v. 2, p. 59–77, 03 1997. Citado na página 61.
- Andre Bondi Alberto Avritzer. *System and method for triggering software rejuvenation using a customer affecting performance metric*. 2007. Citado na página 63.
- Andre B. Bondi Alberto Avritzer. *Dynamic tuning of a software rejuvenation method using a customer affecting performance metric*. 2011. Citado na página 66.
- Rajanikanth Tanikella Alberto Avritzer. *System and Method for Detecting Security Intrusions and Soft Faults Using Performance Signatures*. 2010. Citado na página 22.
- BALSAMO, S.; HARRISON, P. G.; MARIN, A. A unifying approach to product-forms in networks with finite capacity constraints. *ACM SIGMETRICS Performance Evaluation Review*, ACM New York, NY, USA, v. 38, n. 1, p. 25–36, 2010. Citado na página 70.
- BANKS, H. et al. Simulation algorithms for continuous time markov chain models. *Studies in Applied Electromagnetics and Mechanics*, v. 37, p. 3–18, 01 2012. Citado na página 70.
- BÖHM, W. A note on queueing systems exposed to disasters. Department of Statistics and Mathematics, WU Vienna University of Economics . . . , 2008. Citado 2 vezes nas páginas 17 e 44.
- BOVENZI COTRONEO, P.; RUSSO. On the aging effects due to concurrency bugs: A case study on mysql. *2012 IEEE 23rd International Symposium on Software Reliability Engineering.*, 2012. Citado na página 62.
- BROWN, M.; SHAO, Y.-S. Identifying coefficients in the spectral representation for first passage time distributions. *Probability in the Engineering and Informational Sciences*, Cambridge University Press, v. 1, n. 1, p. 69–74, 1987. Citado na página 45.
- BURKATOVSKAYA, Y.; KABANOVA, T.; TOKAREVA, O. Sign CUSUM Algorithm for Change-Point Detection of the MMPP Controlling Chain State. In: SPRINGER. *International Conference on Information Technologies and Mathematical Modelling*. [S.l.], 2016. p. 18–33. Citado 8 vezes nas páginas 17, 20, 21, 22, 25, 37, 38 e 78.
- CAIRNS, B.; POLLETT, P. Extinction times for a general birth, death and catastrophe process. *Journal of Applied Probability*, v. 41, 12 2004. Citado 2 vezes nas páginas 20 e 65.
- CAMPOS, E. de; TEIXEIRA, E. *Controles típicos de equipamentos e processos industriais*. BLUCHER, 2010. ISBN 9788521216957. Disponível em: <https://books.google.com.br/books?id=O2y5DwAAQBAJ>. Citado 6 vezes nas páginas 18, 25, 26, 29, 32 e 68.
- CASADO-VARA, R. et al. Non-linear adaptive closed-loop control system for improved efficiency in iot-blockchain management. *Information Fusion*, v. 49, p. 227–239, 2019. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253518306079>. Citado na página 68.

- CHAO, X. A queueing network model with catastrophes and product form solution. *Operations Research Letters*, Elsevier, v. 18, n. 2, p. 75–79, 1995. Citado na página 70.
- COTRONEO, D. et al. Software aging and rejuvenation: Where we are and where we are going. In: *2011 IEEE Third International Workshop on Software Aging and Rejuvenation*. [S.l.: s.n.], 2011. p. 1–6. Citado na página 61.
- CRAWFORD, F. W.; HO, L. S. T.; SUCHARD, M. A. Computational methods for birth-death processes. *WIREs Computational Statistics*, v. 10, n. 2, p. e1423, 2018. Disponível em: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.1423>. Citado na página 77.
- CRESCENZO, A. D. et al. On the first-visit-time problem for birth and death processes with catastrophes. *arXiv preprint math/0307206*, 2003. Citado na página 45.
- DOHI, T.; TRIVEDI, K. S.; AVRITZER, A. *Handbook of Software Aging and Rejuvenation: Fundamentals, Methods, Applications, and Future Directions*. [S.l.]: World Scientific, 2020. Citado na página 65.
- FILL, J. A. The passage time distribution for a birth-and-death chain: Strong stationary duality gives a first stochastic proof. *Journal of Theoretical Probability*, Springer, v. 22, n. 3, p. 543–557, 2009. Citado na página 45.
- FOURNEAU, J.-M.; MAJHOUB, Y. A. E. Processor sharing g-queues with inert customers and catastrophes: A model for server aging and rejuvenation. *Probability in the Engineering and Informational Sciences*, Cambridge University Press, v. 31, n. 4, p. 420–435, 2017. Citado na página 65.
- GONÇALVES, C. F. et al. A model-based approach to anomaly detection trading detection time and false alarm rate. In: *IEEE. 2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*. [S.l.], 2020. p. 1–8. Citado 7 vezes nas páginas 17, 18, 39, 44, 47, 48 e 64.
- GROTTKE, M. et al. On the efficiency of sampling and countermeasures to critical-infrastructure-targeted malware campaigns. *ACM SIGMETRICS Performance Evaluation Review*, ACM New York, NY, USA, v. 43, n. 4, p. 33–42, 2016. Citado 2 vezes nas páginas 18 e 65.
- GUEPIE, B.; FILLATRE, L.; NIKIFOROV, I. Sequential detection of transient changes. *Sequential Analysis*, v. 31, 10 2012. Citado na página 67.
- HARCHOL-BALTER, M. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. 1st. ed. USA: Cambridge University Press, 2013. ISBN 1107027500. Citado 2 vezes nas páginas 70 e 78.
- HLUCHYJ. Analysis of the leaky bucket algorithm for on off data sources. *Journal of High Speed Networks*, p. 81–98, 1993. Citado na página 66.
- HOMCHAUDHURI, S. Implementation of micro-rejuvenation with cooperative stack leasing. In: *Handbook Of Software Aging And Rejuvenation: Fundamentals, Methods, Applications, And Future Directions*. [S.l.]: World Scientific, 2020. p. 327–351. Citado na página 70.

- HUANG, Y. et al. Software rejuvenation: Analysis, module and applications. In: IEEE. *Twenty-fifth international symposium on fault-tolerant computing. Digest of papers*. [S.l.], 1995. p. 381–390. Citado 3 vezes nas páginas 18, 55 e 61.
- KUMAR, B. K.; ARIVUDAINAMBI, D. Transient solution of an m/m/1 queue with catastrophes. *Computers & Mathematics with applications*, Elsevier, v. 40, n. 10-11, p. 1233–1240, 2000. Citado 2 vezes nas páginas 17 e 44.
- LI, J. et al. Planning optimal rejuvenation policy for aging software systems via a two-layer model. *IEEE Access*, v. 8, p. 136725–136735, 2020. Citado na página 63.
- MARIN, A.; BALSAMO, S.; FOURNEAU, J.-M. Lb-networks: A model for dynamic load balancing in queueing networks. *Performance Evaluation*, v. 115, p. 38–53, 2017. ISSN 0166-5316. Disponível em: <https://www.sciencedirect.com/science/article/pii/S016653161630222X>. Citado na página 70.
- MARIN, A. et al. Modeling service mixes in access links: Product form and oscillations. In: IEEE. *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. [S.l.], 2022. p. 312–318. Citado 3 vezes nas páginas 21, 22 e 70.
- MEYLAHN, J. M.; SABHAPANDIT, S.; TOUCHETTE, H. Large deviations for markov processes with resetting. *Physical Review E*, APS, v. 92, n. 6, p. 062148, 2015. Citado na página 65.
- MIRANDA, L. et al. Sequential performance analysis of systems that age and rejuvenate. In: *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. [S.l.: s.n.], 2022. p. 146–153. Citado 5 vezes nas páginas 18, 19, 32, 40 e 52.
- PISHRO-NIK, H. Introduction to probability, statistics, and random processes. p. 744, 2014. Disponível em: <https://www.probabilitycourse.com,KappaResearchLLC>. Citado na página 78.
- ROMAGNOLI, R.; KROGH, B. H.; SINOPOLI, B. Design of software rejuvenation for cps security using invariant sets. In: IEEE. *2019 American Control Conference (ACC)*. [S.l.], 2019. p. 3740–3745. Citado na página 64.
- SANTOS, A. et al. Assessment of wireless hart networks in closed-loop control system. In: *2015 IEEE International Conference on Industrial Technology (ICIT)*. [S.l.: s.n.], 2015. p. 2172–2177. Citado na página 68.
- SINGH, S. K.; ACHARYA, S. K. A Bayesian inference to estimate change point for traffic intensity in M/M/1 queueing model. *OPSEARCH*, Springer, v. 59, n. 1, p. 166–206, 2022. Citado na página 20.
- SUNDARAM, V. et al. Improving dependability using shared supplementary memory and opportunistic micro rejuvenation in multi-tasking embedded systems. In: IEEE. *13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007)*. [S.l.], 2007. p. 240–247. Citado na página 70.
- WANG, S.; LIU, J. Harrrd: Real-time software rejuvenation decision based on hierarchical analysis under weibull distribution. In: *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*. [S.l.: s.n.], 2020. p. 83–90. Citado na página 63.

---

ZHAO, T. Evaluation of software performance affected by aging. *2010 IEEE Second International Workshop on Software Aging and Rejuvenation*, 2010. Citado 2 vezes nas páginas [24](#) e [62](#).

# Apêndices

# APÊNDICE A – PROCESSOS DE NASCIMENTO E MORTE

Processos de nascimento-morte são processos estocásticos usados para modelar a dinâmica populacional com dois parâmetros principais, a taxa de natalidade e a taxa de mortalidade, que são, respectivamente, a taxa em que novas linhagens aparecem e a taxa em que linhagens são removidas do processo (ANDRÉOLETTI et al., 2022). Os processos de nascimento-morte são uma classe flexível de cadeias de Markov de tempo contínuo que modelam o número de “partículas” em um sistema, onde cada partícula pode “dar à luz” a outra partícula ou “morrer”. A utilidade dos processos de nascimento-morte reside no fato de que “partícula” pode se referir a um membro de qualquer sistema discreto potencialmente interativo no qual apenas se mantém o controle do número de objetos existentes. Os processos de nascimento-morte são ferramentas de modelagem populares em evolução, biologia populacional, genética e ecologia. Por exemplo, se interpretarmos as partículas como espécies em um cenário macroevolutivo, os processos de nascimento-morte podem ser usados para estudar especiação e extinção em escalas de tempo evolutivas. Eles também podem ser usados para estudar a dinâmica de doenças infecciosas em uma população finita, onde o número de indivíduos infectados é a quantidade de interesse. Na evolução molecular, podem modelar nucleotídeos inseridos e deletados em uma sequência de DNA ou RNA como parte de um método de alinhamento probabilístico, elementos genéticos móveis/transponíveis, famílias de genes ou mesmo cromossomos inteiros. Os processos de nascimento-morte podem modelar populações de organismos em um ambiente com recursos limitados. Em populações finitas, eles também são comumente usados para modelar quantidades de interesse em um cenário evolutivo, como mutações, frequências alélicas, seleção ou coalescência (CRAWFORD; HO; SUCHARD, 2018).

# APÊNDICE B – SIMULADOR DE EVENTOS DISCRETOS

A seguir, descrevemos brevemente alguns detalhes sobre o simulador de eventos discretos usado para a análise integrada do Processo MMPP, *bucket* e fila  $M/M/1$  com envelhecimento e rejuvenescimento. Nesta etapa da pesquisa, utilizamos como base a simulação de uma Cadeia de Markov de Tempo Contínuo (HARCHOL-BALTER, 2013).

## B.1 Fundamentos

Como ilustrado na Figura 19, uma vez no estado  $i$ , geramos variáveis aleatórias exponenciais para todos os possíveis estados de transição  $j$ ,  $k$  e  $l$ . Então, uma disputa é estabelecida entre as variáveis aleatórias exponenciais geradas, e a que obtiver o menor tempo ganhará a disputa. Ocorre então uma transição de estado para  $j$ ,  $k$  ou  $l$ . Todos os tempos de permanência em um estado e os tempos de transições são independentes (*memoryless*). Utilizamos também os conceitos de *merging* e *splitting* de processos Poisson, preconizados por (PISHRO-NIK, 2014), descritos no Anexo C e ilustrados nas Figuras 20 e 21, e nos baseamos em (BURKATOVSKAYA; KABANOVA; TOKAREVA, 2016) para detectar instantes de transição de estados da cadeia.

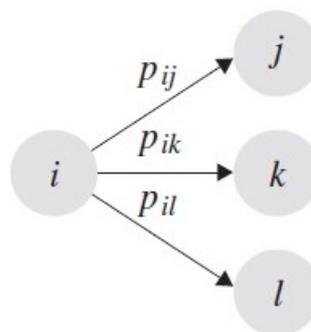


Figura 19 – Dinâmica MMPP utilizando CTMC

## B.2 Métricas de interesse

Com base no fluxo de chegadas nos estados 0 e 1, correspondendo a  $G$  e  $A$ , obtidos em (3.7), podemos caracterizar os instantes de transições  $GA$  e  $AG$ . Sejam  $D_i^A$  e  $D_i^G$  os instantes da  $i$ -ésima entrada no estado  $A$  e no estado  $G$ , respectivamente:

$$D_0^A = \min\{t_i : \tau_i = V_i^A\} \quad (\text{B.1})$$

$$D_k^G = \min\{t_i : \tau_i = V_i^G \text{ e } t_i > D_k^A\} \quad (\text{B.2})$$

$$D_k^A = \min\{t_i : \tau_i = V_i^A \text{ e } t_i > D_{k-1}^G\} \quad (\text{B.3})$$

Temos  $D_0^A < D_0^G < D_1^A < D_1^G < D_2^A < \dots$

Sejam  $N_G$  e  $N_A$  o número de transições de  $A$  para  $G$  e de  $G$  para  $A$ , respectivamente.  $N_G$  e  $N_A$  são os últimos índices em que as expressões (B.2) e (B.3) são bem definidas, respectivamente. Sejam  $T_A$ ,  $T_G$  a duração total dos intervalos em que as cadeias estão nos estados  $A$  e  $G$ :

$$T_A = \sum_{k=0}^{N_G} D_k^G - D_k^A \quad (\text{B.4})$$

$$T_G = \sum_{k=0}^{N_A} D_{k+1}^A - D_k^G \quad (\text{B.5})$$

As taxas de transição entre os estados da cadeia são estimadas seguindo a seguinte equação:

$$\hat{\delta}_{AG} = \frac{N_G}{T_A}, \quad \hat{\delta}_{GA} = \frac{N_A}{T_G} \quad (\text{B.6})$$

# APÊNDICE C – AGREGAÇÃO E DESAGREGAÇÃO DE FLUXOS POISSON

**Agregação de fluxos de Poisson:** Seja, dois processos de Poisson independentes com taxas independentes:

$$N_1(t) \sim \text{Poisson}(\theta_1) \quad (\text{C.1})$$

$$N_2(t) \sim \text{Poisson}(\theta_2) \quad (\text{C.2})$$

$$N(t) = N_1(t) + N_2(t) \quad (\text{C.3})$$

o processo randômico  $N(t)$  é obtido combinando as chegadas em  $N_1(t)$  e  $N_2(t)$ :

$$N(t) \sim \text{Poisson}(\theta = (\theta_1 + \theta_2)) \quad (\text{C.4})$$

Desta forma, como  $N_1(t)$  e  $N_2(t)$  são independentes e ambos possuem incrementos independentes, concluímos que  $N(t)$  também possui incrementos independentes:

$$N_1(t), N_2(t), \dots, N_m(t) \quad (\text{C.5})$$

$$N_j \sim \text{Poisson}(\theta_j), \quad 1 \leq j \leq m \quad (\text{C.6})$$

$$N(t) = N_1(t) + N_2(t) + \dots + N_m(t), t \in (0, \infty) \quad (\text{C.7})$$

$$N(t) \sim \text{Poisson}(\theta_1 + \theta_2 + \dots + \theta_m) \quad (\text{C.8})$$

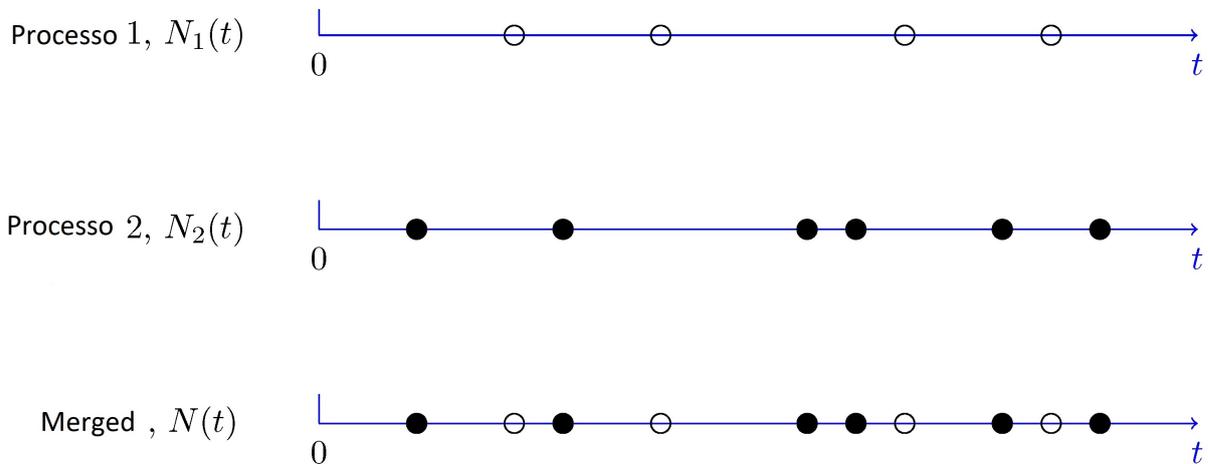


Figura 20 – Agregação de fluxos de Poisson:

**Desagregação de fluxos de Poisson:** Seja, um processo de Poisson com uma taxa:

$$N(t) \sim \text{Poisson}(\theta) \tag{C.9}$$

onde dividimos  $N(t)$  em dois processos:

$$N(t) = N_1(t) + N_2(t) \tag{C.10}$$

Para cada chegada, uma moeda é lançada. Se a moeda cair cara, com probabilidade  $l$ , a chegada é enviada para o primeiro processo, caso contrário, probabilidade  $(1 - l)$ , é enviada para o segundo processo. Os lançamentos da moeda são independentes entre si e independentes de  $N(t)$ . Então:

$$0 = N_1(t) \tag{C.11}$$

$$1 = N_2(t) \tag{C.12}$$

$$N_1(t) \sim \text{Poisson}(\theta l) \tag{C.13}$$

$$N_2(t) \sim \text{Poisson}(\theta(1 - l)) \tag{C.14}$$

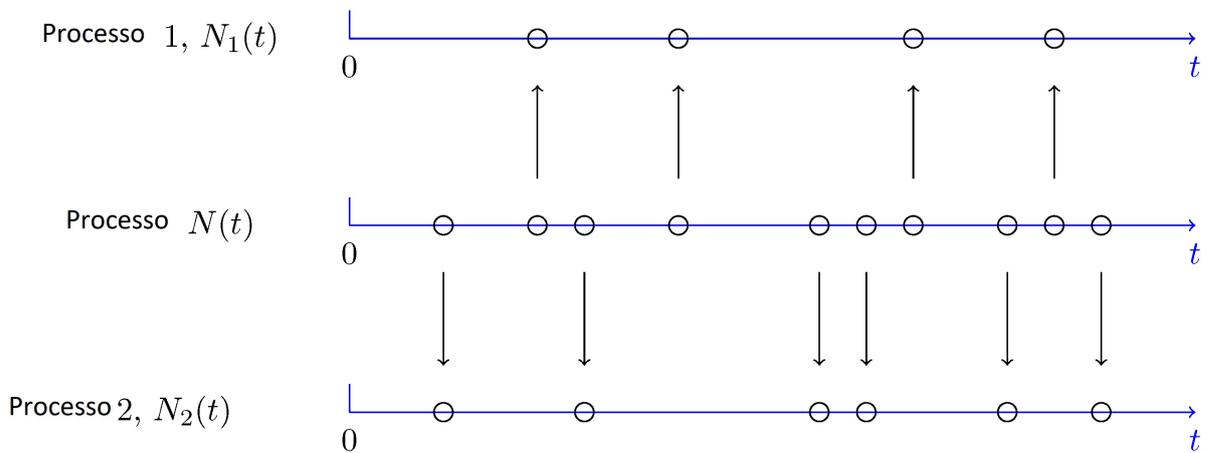


Figura 21 – Desagregação de fluxos de Poisson: