UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Daniel Fernandes Gomes

ANÁLISE DE DESEMPENHO E OTIMIZAÇÃO DO SISTEMA DE CONTROLE PARA REDES DE PETRI ESTOCÁSTICAS

Belo Horizonte

2025

Dissertação de Mestrado.

Daniel Fernandes Gomes

ANÁLISE DE DESEMPENHO E OTIMIZAÇÃO DO SISTEMA DE CONTROLE PARA REDES DE PETRI ESTOCÁSTICAS

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Carlos Andrey Maia

Belo Horizonte

2025

G633a

Gomes, Daniel Fernandes.
Análise de desempenho e otimização do sistema de controle para Redes de Petri Estocásticas [recurso eletrônico] / Daniel Fernandes Gomes. – 2025.
1 recurso online (91 f. : il., color.) : pdf.

Orientador: Carlos Andrey Maia.

Dissertação (mestrado) – Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 73-91.

Bibliografia: f. 71-72.

1. Engenharia elétrica – Teses. 2. Análise estocástica – Teses. 3. Redes de petri – Teses. 4. Otimização – Teses. I. Maia, Carlos Andrey. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Marcio A. A. Gomes CRB/6 2812
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

ESCOLA DE ENGENHARIA

COLEGIADO DO CURSO DE GRADUAÇÃO / PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"Análise de Desempenho e Otimização do Sistema de Controle Para Redes de Petri Estocásticas"

Daniel Fernandes Gomes

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 27 de fevereiro de 2025.

Por:

Prof. Dr. Carlos Andrey Maia
DEE (UFMG) - Orientador

Prof. Dr. Rodney Rezende Saldanha
DEE (UFMG)

Prof. Dr. Vinícius Mariano Gonçalves
DEE (UFMG)

Prof. Dr. Roberto Gomes Ribeiro
Departamento de Computação e Sistemas (UFOP)

Documento assinado eletronicamente por **Carlos Andrey Maia**, **Professor do Magistério Superior**, em 27/02/2025, às 17:09, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rodney Rezende Saldanha**, **Membro de comissão**, em 12/03/2025, às 08:49, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Vinicius Mariano Goncalves**, **Membro**, em 12/03/2025, às 20:33, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Roberto Gomes Ribeiro**, **Usuário Externo**, em 20/03/2025, às 10:08, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3975394** e o código CRC **6C4E6A81**.

Referência: Processo nº 23072.210495/2025-00 SEI nº 3975394

Agradecimentos

Ao concluir este trabalho, expresso minha mais profunda gratidão a Deus, pela oportunidade, força e inspiração concedidas ao longo desta jornada.

Aos meus pais, José e Áurea, pelo amor, ensinamentos e apoio incondicional, que foram fundamentais em cada etapa do meu crescimento pessoal e profissional. Aos meus irmãos e sobrinhos, que sempre me incentivaram e encorajaram nos momentos mais desafiadores, proporcionando-me segurança e motivação para seguir adiante.

Ao meu orientador, Prof. Dr. Carlos Andrey Maia, pela dedicação exemplar e pelo compromisso com os princípios morais e profissionais que elevam sua atuação acadêmica e científica. Sua orientação criteriosa, paciência e incentivo foram essenciais para o desenvolvimento deste trabalho.

Ao Programa de Pós-Graduação em Engenharia Elétrica da UFMG, a todos os professores e servidores que, de alguma forma, contribuíram para o meu crescimento acadêmico, profissional e pessoal.

Ao CNPq, pelo suporte financeiro concedido durante o desenvolvimento desta pesquisa, viabilizando a realização deste estudo e permitindo que ele fosse conduzido com rigor e profundidade.

Aos meus colegas, amigos e familiares, que, de diferentes formas, torceram, apoiaram e estiveram ao meu lado, compartilhando essa jornada com palavras de incentivo e compreensão.

Aos colegas de trabalho do CIT SENAI, IST Automotiva e IST Metal Mecânica, pelo apoio contínuo e por proporcionarem as condições necessárias para que eu pudesse conciliar minhas atividades profissionais e acadêmicas. Um agradecimento especial ao meu coordenador, Júlio Pires, por sua compreensão e suporte, permitindo minha liberação do trabalho nos horários de aula, o que foi essencial para a concretização deste projeto.

Um agradecimento especial à minha esposa, Rachel, e à minha filha, Helena, que são a minha base e a minha maior fonte de inspiração. O amor, a paciência e a compreensão de ambas foram fundamentais para que eu pudesse me dedicar a este projeto com determinação e empenho.

A lista de agradecimentos é extensa, e certamente não consigo nomear todos aqueles que, de alguma forma, contribuíram para que este trabalho se tornasse realidade. No entanto, a todos que me apoiaram ao longo deste percurso, deixo aqui minha mais sincera gratidão.

Resumo

Este trabalho aborda a otimização de sistemas de filas do tipo cliente-servidor com possibilidade de abandono, modelados por meio de Redes de Petri Temporizadas Estocásticas (RPTs). Esses sistemas permitem representar, de forma estruturada, eventos estocásticos como chegada de clientes, atendimento e abandono proporcional ao tamanho da fila, sendo úteis para a análise de desempenho em ambientes dinâmicos.

Com o objetivo de reduzir a taxa de abandono sem comprometer a produtividade, foi introduzido um mecanismo de controle determinístico, baseado na limitação da fila e na temporização da admissão de clientes. A partir dessa estrutura, foram aplicadas técnicas de otimização para encontrar os melhores parâmetros de controle, utilizando Busca Exaustiva (BE) e Algoritmos Genéticos (AG).

Foram formuladas funções objetivo voltadas à maximização do lucro do sistema, considerando a taxa de atendimento e penalizações associadas ao abandono. Os resultados demonstram que, enquanto a BE garante a solução ótima em espaços de busca reduzidos, os AGs se destacam pela capacidade de explorar espaços maiores com menor custo computacional, identificando soluções próximas do ótimo. Assim, os AGs mostram-se uma abordagem viável e eficiente para otimizar o controle de sistemas estocásticos com características de filas e abandono.

Palavras-chave: Redes de Petri estocásticas. Sistemas a Eventos Discretos. Algoritmos genéticos. Sistemas cliente-servidor, Otimização de controle.

Abstract

This work addresses the optimization of client-server queueing systems with the possibility of abandonment, modeled through Stochastic Timed Petri Nets (STPNs). These systems allow for a structured representation of stochastic events such as customer arrival, service, and abandonment proportional to queue length, making them suitable for performance analysis in dynamic environments.

To reduce the abandonment rate without compromising productivity, a deterministic control mechanism was introduced, based on queue limitation and admission timing. From this structure, optimization techniques were applied to determine the best control parameters, using Exhaustive Search (ES) and Genetic Algorithms (GAs).

Objective functions were formulated to maximize system profit, taking into account the service rate and penalties associated with abandonment. The results show that while ES guarantees the optimal solution in small search spaces, GAs stand out for their ability to explore larger spaces at lower computational cost, identifying near-optimal solutions. Therefore, GAs prove to be a viable and efficient approach for optimizing control in stochastic systems with queueing and abandonment characteristics.

Keywords: Stochastic Petri Nets. Discrete Event Systems. Client-server systems. Genetic Algorithms. Control optimization.

Lista de ilustrações

Figura 1 – Sistema de filas 20
Figura 2 – Representação gráfica de uma rede de Petri 22
Figura 3 – Sistema de fila simples 28
Figura 4 – Representação do sistema descrito pela lei de Little. 32
Figura 5 – Modelo de sistema cliente-servidor com abandono. 35
Figura 6 – Modelo inicial da Rede de Petri representando o sistema de fila cliente-servidor com abandono. 45
Figura 7 – Modelo expandido da Rede de Petri para implementação de temporização no sistema de fila cliente-servidor. 46
Figura 8 – Rede de Petri com sistema de controle 52
Figura 9 – Exemplo de uma competição de torneio entre dois pais, onde foi escolhido o pai que representa o menor valor de custo 56
Figura 10 – Exemplo dos filhos com novos valores de M e f gerados dos pais. 56
Figura 11 – Gráfico 3D da FO $L(M, f) = (\text{Tx}_{\text{Atend}} - 10 \cdot \text{Tx}_{\text{Aband}}) \cdot 10$ 62
Figura 12 – Mapa de Calor da FO $L(M, f) = (\text{Tx}_{\text{Atend}} - 10 \cdot \text{Tx}_{\text{Aband}}) \cdot 10$ 62
Figura 13 – Gráfico 3D da Função Objetivo $L(M, f) = (\text{Tx}_{\text{Atend}} - 30 \cdot \text{Tx}_{\text{Aband}}) \cdot 100$ . . 63
Figura 14 – Mapa de Calor da FO $L(M, f) = (\text{Tx}_{\text{Atend}} - 30 \cdot \text{Tx}_{\text{Aband}}) \cdot 100$ 63
Figura 15 – Gráfico 3D da FO $L(M, f) = \text{Tx}_{\text{Atend}}/\text{Tx}_{\text{Aband}}$ 64
Figura 16 – Mapa de Calor da FO $L(M, f) = \text{Tx}_{\text{Atend}}/\text{Tx}_{\text{Aband}}$ 64
Figura 17 – Comparação dos tempos de execução entre as abordagens 68

Lista de tabelas

Tabela 1 – Notação de Kendall para sistemas de filas 29
Tabela 2 – Resumo dos resultados gerais para o sistema $G/G/m$ de fila. 33
Tabela 3 – Tabela detalhada do equacionamento da rede de Petri com abandono - primeira parte 47
Tabela 4 – Tabela detalhada do equacionamento da rede de Petri com abandono - segunda parte (continuação) 47
Tabela 5 – Tabela detalhada do equacionamento da rede de Petri com abandono - terceira parte (continuação) 48
Tabela 6 – Valores temporais das transições. 50
Tabela 7 – Resultados das simulações do sistema de fila em RPT estocástica sem controle 59
Tabela 8 – Resultados das simulações do sistema de fila em RPT estocástica com controle determinístico 60
Tabela 9 – Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica com Controle Otimizado via Busca Exaustiva 61
Tabela 10 – Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica com Controle Otimizado via Algoritmo Genético 65
Tabela 11 – Comparação das Abordagens de Otimização — Parte 1 68
Tabela 12 – Comparação das Abordagens de Otimização — Parte 2 69

Lista de abreviaturas e siglas

AG	Algoritmo Genético
BE	Busca Exaustiva
FIFO	First In, First Out
IID	Independentes e Identicamente Distribuídos
LP	Linear Programming
NLP	Non-Linear Programming
SED	Sistema a Eventos Discretos
RP	Rede de Petri
RPT	Rede de Petri T-temporizada
RPTE	Rede de Petri Temporizada Estocástica
SED	Sistema a Eventos Discretos
FO	Função Objetivo

Lista de símbolos

$\mathcal{N}$	Conjunto que define a rede de Petri
$\mathcal{P}$	Conjunto finito de lugares
$\mathcal{T}$	Conjunto finito de transições
$\mathcal{A}$	Conjunto finito de arcos
w	Representação da função de peso
x_0	Estado inicial da rede de Petri
$\mathcal{D}$	Vetor do tempo de atraso
I	Matriz Identidade
A^+	Matriz de incidência da rede de Petri
A^-	Matriz que representa a retirada de fichas da rede de Petri
L	Matriz de ponderação
x	Vetor de marcação
t	Transição em redes de Petri
p	Lugares na rede de Petri
X_c	Lugar referente ao controle na rede de Petri
γ^T	Vetor de conservação da rede de Petri
Γ	Conjunto de vetores de restrição das redes de Petri
A_c	Matriz com controlador da rede de Petri
A	Distribuição do tempo de chegada
B	Distribuição do tempo de serviço
m	Quantidade de servidores do sistema
K	Capacidade da fila
Z	Disciplina da fila

μ	Taxa de serviço
ρ	Intensidade de tráfego
λ	Taxa média de chegada
W	Tempo de permanência no sistema
$\hat{\theta}$	Estimativa do valor esperado
X_i	Número de amostras
n	Número de amostras geradas
$f(x)$	Função objetivo
$g_i(x)$	Restrição de desigualdade
$h_j(x)$	Restrição de igualdade

Sumário

1 INTRODUÇÃO . . . 16
1.1 Apresentação . . . 16
1.2 Motivação . . . 16
1.3 Objetivos . . . 17
1.3.1 Objetivo Geral . . . 17
1.3.2 Objetivos Específicos . . . 17
1.4 Organização da Dissertação . . . 18

2 CONCEITOS PRELIMINARES . . . 19
2.1 Introdução . . . 19
2.2 Sistemas a Eventos Discretos . . . 19
2.2.1 Ferramentas de Modelagem para Sistemas a Eventos Discretos . . . 20
2.3 Redes de Petri . . . 21
2.3.1 Rede de Petri Temporizada . . . 24
2.3.2 Rede de Petri Controlada . . . 25
2.4 Teoria das Filas . . . 28
2.4.1 Desempenho de um sistema de fila . . . 29
2.4.1.1 Análise do Tempo de Espera e do Estado Estacionário . . . 30
2.4.1.2 Lei de Little . . . 31
2.5 Filas com Clientes Impacientes . . . 32
2.5.1 Recusa (Balking) . . . 33
2.5.2 Abandono (Reneging) . . . 34
2.6 Sistemas do Tipo Cliente-Servidor . . . 34
2.6.1 Modelos Computacionais de Monte Carlo . . . 35
2.7 Otimização . . . 36
2.7.1 Tipos de Otimização . . . 37
2.7.1.1 Otimização Linear . . . 37
2.7.1.2 Otimização Não Linear . . . 38
2.7.1.3 Otimização Discreta . . . 38
2.7.1.4 Otimização Multiobjetivo . . . 38
2.7.2 Métodos de Otimização . . . 39
2.7.3 Otimização por Busca Exaustiva . . . 39
2.7.3.1 Definição Formal . . . 39
2.7.4 Otimização por Algoritmo Genético . . . 41

3 METODOLOGIA . . . 44

3.1 Introdução . . . 44
3.2 Modelagem Rede de Petri . . . 45
3.2.1 Expansão do Modelo para Implementação de Temporização . . . 45
3.2.2 Condição Inicial da rede de Petri temporizada . . . 46
3.2.3 Matriz de Incidência da RPT . . . 47
3.2.4 Modelo computacional de Monte Carlo . . . 49
3.2.5 RPT Controlada . . . 50
3.3 Otimização . . . 52
3.3.1 Formulação da Função Objetivo . . . 52
3.3.1.1 Critérios para a Escolha da Penalização . . . 53
3.3.1.2 Aplicação na Otimização . . . 54
3.3.2 Busca Exaustiva . . . 54
3.3.3 Algoritmo Genético . . . 55

I RESULTADOS 57

4 RESULTADOS . . . 58
4.1 Apresentação dos Resultados . . . 58
4.2 Simulação de RPT com abandono . . . 58
4.3 Simulação de RPT com Controle Determinístico . . . 60
4.4 Otimização por Busca Exaustiva . . . 60
4.4.1 Visualização dos Resultados da FO 1 . . . 61
4.4.2 Visualização dos Resultados da FO 2 . . . 62
4.4.3 Visualização dos Resultados da FO 3 . . . 63
4.5 Otimização por Algoritmo Genético . . . 64
4.5.1 Configuração do Algoritmo Genético . . . 65
4.5.2 Critério de Parada . . . 65
4.5.3 Resultados Obtidos . . . 65
4.6 Análise da Complexidade Computacional . . . 66
4.6.1 Custo Computacional do Controle Determinístico . . . 66
4.6.2 Custo Computacional da Busca Exaustiva . . . 66
4.6.3 Custo Computacional do Algoritmo Genético . . . 67
4.6.4 Comparação das Abordagens . . . 67

5 CONCLUSÃO . . . 70
5.1 Trabalhos Futuros . . . 71

REFERÊNCIAS . . . 72

APÊNDICE A – CÓDIGO PARA SIMULAÇÃO DE REDES DE PETRI COM ABANDONO 74
A.1 Descrição do Código 74

APÊNDICE B – CÓDIGO PARA SIMULAÇÃO DE REDE DE PETRI COM ABANDONO CONTROLADO 78

APÊNDICE C – CÓDIGO DA BUSCA EXAUSTIVA PARA OTIMIZAÇÃO DA REDE DE PETRI 82

APÊNDICE D – CÓDIGO AG PARA OTIMIZAÇÃO DA REDE DE PETRI 86

1 Introdução

1.1 Apresentação

Sistemas a Eventos Discretos (SED) são amplamente utilizados na modelagem e análise de processos em diversas áreas, como manufatura, logística, telecomunicações e sistemas computacionais. Esses sistemas, caracterizados por mudanças de estado discretas em resposta a eventos específicos, frequentemente exibem comportamentos complexos e estocásticos, tornando sua análise e controle desafiadores. Dentre as ferramentas disponíveis para modelagem de SED, as Redes de Petri estocásticas se destacam por sua capacidade de representar, de forma gráfica e analítica, a interação dinâmica entre os elementos do sistema.

O controle eficiente de sistemas modelados por Redes de Petri estocásticas é essencial para garantir um bom desempenho, especialmente em aplicações onde há incerteza e variabilidade. No entanto, o projeto de sistemas de controle enfrenta desafios como a alta complexidade combinatória e a necessidade de equilibrar múltiplos objetivos, como minimizar taxas de abandono, reduzir tempos de espera e otimizar a produtividade. Para lidar com esses desafios, diferentes técnicas de otimização podem ser aplicadas, variando entre métodos exatos e heurísticos.

Entre os métodos exatos, a busca exaustiva consiste em avaliar todas as possíveis configurações do sistema para identificar a solução ótima. Embora essa abordagem garanta o melhor resultado, sua aplicação pode ser inviável em sistemas de grande porte devido ao crescimento fatorial do espaço de busca. Ainda assim, é uma ferramenta valiosa para validar soluções obtidas por métodos aproximados e compreender o comportamento do sistema em cenários específicos.

Neste cenário, técnicas de otimização, como os algoritmos genéticos, oferecem uma abordagem promissora. Esses algoritmos, inspirados nos mecanismos de evolução natural, têm a capacidade de explorar grandes espaços de solução em busca de configurações que atendam a múltiplos critérios de desempenho, mesmo em sistemas altamente complexos e incertos.

1.2 Motivação

A crescente necessidade de sistemas eficientes para lidar com cenários de alta variabilidade e incerteza, como os encontrados em sistemas do tipo cliente-servidor, evidencia a importância de desenvolver estratégias de controle otimizadas. As Redes de Petri estocásticas permitem modelar com precisão sistemas dinâmicos e identificar comportamentos críticos, como gargalos e taxas de abandono. No entanto, a estrutura combinatória desses modelos impõe desafios à definição de estratégias de controle que garantam um desempenho adequado.

Diante disso, técnicas de otimização aplicadas ao controle tornam-se uma abordagem viável para melhorar a eficiência desses sistemas. Neste trabalho, duas estratégias são investigadas: a busca exaustiva e os algoritmos genéticos. A busca exaustiva, embora raramente aplicada ao controle em Redes de Petri Estocásticas, permite uma exploração completa do espaço de soluções, possibilitando identificar configurações ótimas para o sistema dentro de limites computacionais viáveis. Já os algoritmos genéticos oferecem uma alternativa que busca soluções de alto desempenho de forma eficiente, sem a necessidade de testar todas as combinações possíveis.

A motivação deste trabalho é desenvolver uma abordagem que integre essas técnicas de otimização ao controle de Redes de Petri Estocásticas, explorando suas capacidades e limitações no contexto de sistemas cliente-servidor. Embora a formulação original do problema permita considerar múltiplos fatores, a definição da função objetivo levou a uma abordagem mono-objetivo, focada na otimização de um critério específico de desempenho.

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver e implementar um método de otimização para controle de sistemas modelados por Redes de Petri Estocásticas, utilizando busca exaustiva e algoritmos genéticos, com o objetivo de reduzir as taxas de abandono e melhorar a eficiência operacional em sistemas do tipo cliente-servidor.

1.3.2 Objetivos Específicos

Os objetivos específicos deste trabalho incluem:

- ❑ Modelar sistemas do tipo cliente-servidor utilizando Redes de Petri Estocásticas, considerando fatores como abandono e tempos de espera;
- ❑ Aplicar técnicas de otimização, incluindo busca exaustiva e algoritmos genéticos, para explorar o espaço de soluções e identificar configurações que aprimorem o desempenho do sistema;
- ❑ Avaliar e comparar os resultados obtidos com abordagens tradicionais, considerando métricas como taxa de abandono, tempo médio no sistema e produtividade;
- ❑ Investigar as vantagens e limitações de cada método, analisando sua viabilidade computacional e o impacto das soluções obtidas no desempenho do sistema.

1.4 Organização da Dissertação

Esta dissertação está organizada em capítulos que abordam os principais aspectos do desenvolvimento e aplicação do método proposto:

- O Capítulo 2 apresenta os conceitos preliminares necessários para o entendimento do trabalho, incluindo uma introdução aos Sistemas a Eventos Discretos, Redes de Petri e algoritmos genéticos.
- O Capítulo 3 detalha a metodologia desenvolvida para otimizar sistemas de controle utilizando algoritmos genéticos aplicados a Redes de Petri estocásticas.
- No Capítulo 4, são apresentados os resultados obtidos, incluindo uma análise comparativa e uma discussão sobre a eficiência da abordagem proposta.
- Por fim, o Capítulo 5 traz as conclusões do trabalho, destacando as contribuições realizadas e sugerindo possíveis direções para pesquisas futuras.

2 CONCEITOS PRELIMINARES

2.1 Introdução

Neste capítulo, são abordados os conceitos fundamentais necessários para o desenvolvimento do trabalho, incluindo as principais ferramentas utilizadas na modelagem de sistemas a eventos discretos. O objetivo é estabelecer uma base teórica clara, permitindo compreender as abordagens exploradas na dissertação. Para mais detalhes sobre cada tema, recomenda-se a consulta às referências citadas.

2.2 Sistemas a Eventos Discretos

Os Sistemas a Eventos Discretos (SED) representam uma classe específica de sistemas dinâmicos amplamente estudados em diversas áreas da ciência e da engenharia devido à sua relevância em contextos tecnológicos. Esses sistemas caracterizam-se por estados que mudam de maneira discreta em resposta a eventos que ocorrem em pontos específicos no tempo. Exemplos comuns incluem sistemas de manufatura, redes de telecomunicações, entre outros, nos quais eventos discretos, como a chegada de um cliente ou a conclusão de um serviço, definem a dinâmica do sistema.

De forma geral, um sistema é definido como um conjunto de elementos interconectados que trabalham coordenadamente para alcançar um objetivo específico. (CASSANDRAS; LAFORTUNE, 2008) descrevem o conceito de sistema como primitivo, ou seja, algo cuja definição exata pode não ser necessária, pois sua compreensão é frequentemente guiada pela intuição.

O termo “discreto” refere-se a variações ou eventos que assumem valores distintos e específicos, ocorrendo em intervalos separados no tempo ou no espaço. Diferentemente dos sistemas contínuos, onde as variações podem ocorrer de forma suave e ininterrupta, os SEDs operam com mudanças abruptas que ocorrem em instantes bem definidos, conforme (CURY; QUEIROZ, 2002).

A análise desses sistemas é essencial para compreender suas interações e desenvolver modelos que possibilitem prever, controlar e aprimorar seu desempenho. A Figura 1 ilustra um exemplo de sistema de filas, onde eventos como chegada de clientes, espera na fila, atendimento e saída são representados de maneira estruturada.

Os SEDs podem ser modelados por diferentes abordagens, cada uma com características próprias e adequadas a contextos específicos. A próxima seção apresenta as principais ferramentas disponíveis para essa modelagem.

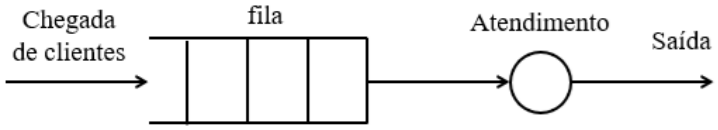

Figura 1 – Sistema de filas

2.2.1 Ferramentas de Modelagem para Sistemas a Eventos Discretos

A modelagem de SEDs pode ser realizada por meio de diversas técnicas, cada uma com suas vantagens e limitações. Entre as principais abordagens utilizadas na literatura, destacam-se:

- **Redes de Petri temporizadas e não temporizadas:** São representações gráficas e matemáticas amplamente utilizadas para modelar concorrência e sincronização. As versões não temporizadas são aplicadas na análise estrutural do sistema, enquanto as Redes de Petri Temporizadas incorporam restrições temporais às transições, permitindo modelar tempos de atraso. No entanto, a adição de temporização pode aumentar a complexidade computacional da análise.

- **Redes de Petri controladas com e sem temporização:** Redes de Petri podem ser estendidas com mecanismos de controle para impor restrições operacionais, garantindo que determinados estados sejam evitados ou que eventos ocorram de maneira ordenada. Quando associadas à temporização, permitem um gerenciamento mais preciso dos tempos de transição e da dinâmica do sistema.

- **Cadeias de Markov:** Modelos baseados em cadeias de Markov são utilizados para representar sistemas onde a transição entre estados ocorre de maneira probabilística, dependendo exclusivamente do estado atual. Essa abordagem é eficiente para análise de confiabilidade e comportamento estacionário, mas apresenta limitação em sistemas de grande porte, devido ao crescimento exponencial do espaço de estados.

- **Teoria das filas:** A teoria das filas fornece modelos matemáticos para analisar sistemas de espera, sendo amplamente aplicada em redes de telecomunicações, serviços e manufatura. Muitos modelos clássicos de filas, são baseados em Cadeias de Markov, permitindo a obtenção de soluções analíticas para métricas como tempo médio de espera e ocupação do sistema. A aplicabilidade da teoria das filas pode ser limitada quando há múltiplas interações concorrentes ou distribuições de tempo complexas que não se encaixam nos modelos analíticos tradicionais.

- **Álgebra tropical Max-Plus:** A álgebra Max-Plus utiliza operações de maximização e soma para modelar sistemas síncronos a eventos discretos, sendo especialmente útil na

análise de tempos críticos e processos cíclicos. Sua formulação clássica é mais adequada para sistemas determinísticos, como redes de transporte e cadeias produtivas com tempos fixos de operação. No entanto, extensões do Max-Plus permitem a incorporação de variações estocásticas, possibilitando a modelagem de incertezas em sistemas temporizados. Ainda assim, o tratamento da estocasticidade nesse contexto exige adaptações adicionais, pois a álgebra Max-Plus tradicional não inclui operações probabilísticas de forma natural.

- **Teoria de linguagem e autômatos:** Modelos baseados em autômatos finitos e gramáticas formais permitem descrever a evolução de sistemas discretos por meio de sequências de eventos. Embora sejam eficientes para a análise de protocolos de comunicação e verificação formal, podem não ser a abordagem mais intuitiva para sistemas altamente concorrentes.

2.3 Redes de Petri

A rede de Petri, é uma ferramenta matemática amplamente utilizada para modelar, analisar e simular sistemas dinâmicos a eventos discretos. Desenvolvida por Carl Adam Petri na década de 1960, essa abordagem combina representações gráficas e analíticas, permitindo descrever o comportamento de sistemas caracterizados por eventos que ocorrem de maneira assíncrona e por estados que mudam de forma discreta (CASSANDRAS; LAFORTUNE, 2008).

Definição 1. *As denominadas redes de Petri (WANG, 1998) são definidas conforme a quíntupla* $\mathcal{N} = (\mathcal{P}, \mathcal{T}, \mathcal{A}, w, x_0)$*, em que:*

- $\mathcal{P}$ *: conjunto finito de lugares;*
- $\mathcal{T}$ *: conjunto finito de transições;*
- $\mathcal{A}$ *: conjunto finito de arcos, dados por* $\mathcal{A} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$*;*
- $w \colon \mathcal{A} \to \mathbb{N}$ *: função de peso dos arcos;*
- $x_0 \in \mathbb{N}^n$ *: estado inicial da rede, onde* n *é o número de lugares.*

O funcionamento de uma rede de Petri baseia-se em fichas (*tokens*), que se movem entre os lugares por meio do disparo (*firing*) das transições. Uma transição $t \in \mathcal{T}$ está habilitada quando todos os seus lugares de entrada contêm pelo menos o número de fichas especificado pelos pesos dos arcos de entrada. Quando uma transição dispara, as fichas são removidas dos lugares de entrada e adicionadas aos lugares de saída, de acordo com os pesos dos arcos de saída, alterando assim a marcação da rede.

A Figura 2 ilustra um exemplo gráfico de uma rede de Petri com transições e lugares definidos. As transições são indicadas por:

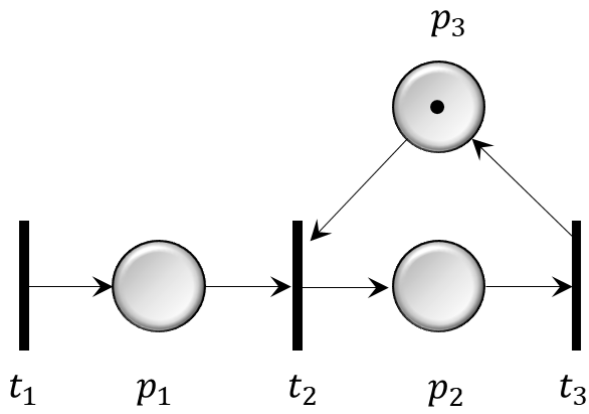

Figura 2 – Representação gráfica de uma rede de Petri

$$\mathcal{P} = \{p_1, p_2, p_3\} \tag{2.1}$$

Enquanto os lugares são designados como:

$$\mathcal{T} = \{t_1, t_2, t_3\} \tag{2.2}$$

Os arcos são definidos por:

$$\mathcal{A} = \{(t_1, p_1)\,(p_1, t_2)\,(t_2, p_2)\,(p_2, t_3)\,(t_3, p_3)\,(p_3, t_2)\} \tag{2.3}$$

Todos os arcos possuem peso $w = 1$, simplificando a análise do modelo. Na configuração inicial, os lugares p_1 e p_2 não possuem fichas, enquanto o lugar p_3 contém uma ficha, Nessas condições a marcação inicial da rede é dada por:

$$x_0 = [0\ 0\ 1] \tag{2.4}$$

Outra definição relevante está relacionada aos lugares de entrada e lugares de saída para as transições, que estão diretamente associados às condições para o disparo (ou ocorrência) de uma dada transição. O conjunto de lugares de entrada para uma transição t_j é denotado por $I(t_j)$. Esses lugares de entrada representam os locais que devem conter fichas suficientes para habilitar o disparo da transição. No exemplo ilustrado na Figura 2, os lugares de entrada das transições são definidos como:

$$I(t_1) = \emptyset; I(t_2) = (p_1) = (p_3); I(t_3 = (p_2)). \tag{2.5}$$

De maneira análoga, podemos definir os lugares de saída para uma transição t_j. Esses lugares de saída estão associados às condições que serão estabelecidas após o disparo de uma

transição, ou seja, são os locais para onde as fichas serão movidas. Para o exemplo dado, os lugares de saída são:

$$O(t_1) = (p_1); O(t_2) = (p_2); O(t_3) = (p_3). \tag{2.6}$$

Com base nesse funcionamento, é possível descrever as mudanças na marcação da rede de forma matemática, utilizando equações recursivas para cada transição. Essas equações são escritas como:

$$x'(p_i) = x(p_i) + k, \quad k \in \mathbb{Z}, \tag{2.7}$$

onde $x(P_1)$ representa o número atual de fichas em um lugar p_i e $x'(p_i)$ indica o número de fichas após o disparo de uma transição.

Exemplo de aplicação: Considere novamente a rede da Figura 2. Denotando o número de fichas nos lugares por $x(p_1), x(p_2), x(p_3)$, temos a marcação inicial:

$$x(p_1) = x(p_2) = 0, x(p_3) = 1 \tag{2.8}$$

Nesse estado, a única transição habilitada é t_1. Quando t_1 dispara, o número de fichas no lugar p_1 é alterado, resultando na nova marcação:

$$x'(p_1) = x(p_1) + 1 \tag{2.9}$$

Dessa forma, as equações recursivas que permitem obter a nova marcação da rede de Petri (RP) após o disparo de qualquer transição t_i podem ser organizadas em uma forma matricial compacta, como segue:

$$x' = x + A \cdot u \tag{2.10}$$

onde:

- x é o vetor de marcações atual da rede, representando o número de fichas em cada lugar;
- x' é o vetor de marcações atualizado após o disparo de uma transição;
- u é o vetor de disparos, onde cada elemento u_i é igual a 1 se a transição t_i foi disparada, e 0 caso contrário;
- A é a matriz de transição da rede, que indica o balanço líquido do número de fichas em cada lugar após o disparo de uma transição.

A matriz A pode ser decomposta em duas partes:

- A^+, que representa a adição de fichas nos lugares de saída após o disparo da transição;
- A^-, que representa a retirada de fichas dos lugares de entrada.

Portanto, temos:

$$A = A^+ - A^- \tag{2.11}$$

Os elementos das matrizes A^+ e A^- são definidos como:

$$\left[A^-\right]_{ij} = w(p_i, t_j) \tag{2.12}$$

$$\left[A^+\right]_{ij} = w(t_j, p_i) \tag{2.13}$$

onde $w(p_i, t_j)$ é o peso do arco que conecta o lugar p_i à transição t_j, enquanto $w(t_j, p_i)$ é o peso do arco que conecta a transição t_j ao lugar p_i.

A matriz A, portanto, fornece o balanço líquido de fichas em cada lugar após o disparo de uma transição. Para exemplificar, considere a matriz de transição para o sistema discutido anteriormente:

$$A = \begin{bmatrix} +1 & -1 & 0 \\ 0 & +1 & -1 \\ 0 & -1 & +1 \end{bmatrix} \tag{2.14}$$

Com isso, o cálculo da marcação futura $u = [t_1, t_2, t_3]^T$ para um dado vetor de disparos é realizado diretamente pela equação matricial.

2.3.1 Rede de Petri Temporizada

As Redes de Petri temporizadas (RPT) representam uma evolução das RPs clássicas, incorporando uma dimensão temporal essencial para capturar o comportamento dinâmico de sistemas complexos. Essa extensão possibilita a modelagem de sistemas em que o tempo desempenha um papel central, permitindo representar tanto eventos determinísticos quanto estocásticos, ampliando o espectro de aplicação das RPTs.

Os tempos associados a essas redes podem ser determinísticos ou estocásticos, dependendo da natureza dos eventos modelados. Enquanto os tempos determinísticos possuem intervalos fixos e previsíveis, os tempos estocásticos são regidos por distribuições de probabilidade, refletindo uma incerteza inerente a muitos sistemas reais. Essa dualidade torna as RPTs altamente versáteis, adequadas para uma ampla variedade de cenários.

A abordagem temporal pode ser aplicada de duas maneiras distintas: aos lugares ou às transições. Nas redes "P-temporizadas", a temporalidade está associada aos lugares, enquanto as redes "T-temporizadas" estão associadas às transições. Essa flexibilidade permite a modelagem das propriedades temporais dos elementos-chave da rede, adaptando-se melhor às características do sistema. Dessa forma, as redes de Petri temporizadas oferecem uma representação mais abrangente e dinâmica, possibilitando a análise do comportamento temporal dos sistemas.

Definição 2. *As denominadas redes de Petri T_temporizadas (WANG, 1998) são definidas conforme a sêxtupla* $N = (\mathcal{P}, \mathcal{T}, \mathcal{A}, w, x_0, \mathcal{D})$*, onde:*

- ❏ $\mathcal{P}$ *é o conjunto finito de lugares;*
- ❏ $\mathcal{T} = \mathcal{T}_D \cup \mathcal{T}_I$ *é o conjunto finito de transições temporizadas* $\mathcal{T}_\mathcal{D}$ *e instantâneas* $\mathcal{T}_I$*;*
- ❏ $\mathcal{A}$ *é o conjunto finito de arcos, dados por* $\mathcal{A} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$*;*
- ❏ $w \colon \mathcal{A} \to \mathbb{N}$ *representa a função de peso;*
- ❏ $x_0 \in \mathbb{N}^n$ *representa o estado inicial da rede, no qual* n *corresponde ao número de lugares na rede;*
- ❏ $\mathcal{D} = \{\mathcal{D}_i : i \in \mathcal{T}_\mathcal{D}\}$ *é o conjunto de tempos de atraso que devem ser respeitados para se disparar as transições temporizadas após habilitadas, sendo* $\mathcal{D}_i$ *um vetor de tempos de atraso para a transição* $\mathcal{T}_i$*.*

2.3.2 Rede de Petri Controlada

O controle em RP é uma área de estudo que busca garantir que a dinâmica da rede respeite condições específicas, assegurando um comportamento desejado do sistema modelado. Essa abordagem se concentra em limitar ou coordenar a movimentação das fichas na rede, de modo que certas restrições ou objetivos sejam alcançados (CASSANDRAS; LAFORTUNE, 2008).

As restrições de controle geralmente estão associadas a condições operacionais do sistema, como evitar sobrecarga de recursos, garantir a sequência correta de operações, evitar deadlocks (bloqueios) e manter a segurança ou estabilidade do sistema. Em um contexto prático, isso pode incluir limitar o número de fichas em um lugar para representar a capacidade máxima de um recurso ou restringir o disparo de transições para impor prioridades entre eventos concorrentes (YAMALIDOU et al., 1996).

Uma RP pode ser controlada impondo restrições sobre o número de fichas em determinados lugares. Essas restrições podem ser expressas matematicamente por desigualdades da forma:

$$Lx \leq b \tag{2.15}$$

onde:

- L é uma matriz de ponderação, que define os pesos associados aos lugares;
- x é o vetor de marcação, representando o estado atual do sistema;
- b é um vetor de limites superiores para o número de fichas.

Para garantir que essas restrições sejam respeitadas, utiliza-se uma abordagem que envolve a criação de lugares de controle (x_c). Esses lugares funcionam como repositórios adicionais de fichas e são projetados para atuar como variáveis de folga, reescrevendo as restrições como:

$$Lx + x_c = b \tag{2.16}$$

onde $x_c \geq 0$. Dessa forma, os lugares de controle garantem que $Lx \leq b$ seja sempre válido. Nessa equação, os lugares de controle ajustam dinamicamente o número de fichas no sistema para assegurar que a desigualdade original seja respeitada. Essa reformulação pode ser expressa na forma matricial como:

$$\begin{bmatrix} L & I \end{bmatrix} \begin{bmatrix} x \\ x_c \end{bmatrix} = b \tag{2.17}$$

onde I é a matriz identidade associada aos lugares de controle. Essa notação unifica a representação dos estados originais da rede (x) e dos lugares de controle (x_c), consolidando o sistema em uma única estrutura de estados ampliada.

Para que o sistema controlado funcione adequadamente, é necessário que ele seja conservativo em relação a certas condições. A conservatividade garante que o comportamento da rede respeite as restrições definidas, independentemente do estado alcançado. Formalmente, uma rede é dita conservativa em relação a um vetor γ se, para qualquer estado y alcançável a partir do estado inicial y_0, a seguinte igualdade for válida:

$$\gamma^T y = \gamma^T y_0 \tag{2.18}$$

Isso significa que o sistema mantém constante uma soma ponderada das fichas, conforme especificado por γ. Para que essa condição seja satisfeita, a matriz de transição A da rede deve obedecer:

$$\gamma^T A = [0] \tag{2.19}$$

No caso de uma rede controlada, a matriz ampliada A_f, que representa tanto o sistema original quanto os lugares de controle, deve garantir que as restrições sejam invariantes. Para isso, definimos um conjunto de vetores Γ, que unifica as condições de restrição da rede:

$$\Gamma = \begin{bmatrix} L \\ I \end{bmatrix} \tag{2.20}$$

A rede controlada será invariante se:

$$\Gamma^T A_f = [0] \tag{2.21}$$

Esse requisito assegura que o sistema controlado nunca viole as condições impostas pelas restrições, mesmo com a evolução dinâmica de suas transições.

A matriz A_f da rede controlada é composta pela matriz de transição da rede original (A) e pela matriz de transição dos lugares de controle (A_c). Para atender às condições de invariância, a matriz A_c deve ser projetada de modo que:

$$\begin{bmatrix} L & I \end{bmatrix} \begin{bmatrix} A \\ A_c \end{bmatrix} = [0] \tag{2.22}$$

o que resulta na seguinte relação:

$$LA + A_c = [0] \tag{2.23}$$

Assim, a matriz A_c do controlador é determinada como:

$$A_c = -LA \tag{2.24}$$

Esse cálculo vincula diretamente o controlador às dinâmicas da rede original, garantindo que ele funcione de forma eficiente e minimamente restritiva. Além disso, é necessário definir a marcação inicial dos lugares de controle ($x_c(0)$) para que as restrições sejam válidas desde o início. Isso é feito usando a equação:

$$x_c(0) = b - Lx(0) \tag{2.25}$$

onde $x(0)$ é a marcação inicial da rede original.

2.4 Teoria das Filas

A teoria das filas foi desenvolvida com o propósito de fornecer modelos que permitam prever o comportamento de sistemas destinados a atender demandas que surgem aleatoriamente (GROSS et al., 2018). Seu objetivo principal é determinar o desempenho desses sistemas em condições operacionais específicas e desenvolver ferramentas descritivas para analisar e estudar filas, em vez de ferramentas prescritivas para controlar o comportamento incerto desses sistemas (CASSANDRAS; LAFORTUNE, 2008).

Segundo (GROSS et al., 2018), a teoria das filas possui inúmeras aplicações valiosas, amplamente documentadas na literatura de probabilidade, pesquisa operacional, ciência de gestão e engenharia industrial. Exemplos incluem o estudo de fluxo de tráfego (veículos, aeronaves, pessoas, comunicações), agendamento (pacientes em hospitais, tarefas em máquinas, programas em computadores) e o projeto de instalações (bancos, agências dos correios, parques de diversões, restaurantes de fast food).

(CASSANDRAS; LAFORTUNE, 2008) destacam que os sistemas de filas contribuíram significativamente para a análise de Sistemas a Eventos Discretos estocásticos, particularmente em problemas relacionados ao compartilhamento eficiente de recursos.

Um sistema de fila pode ser descrito como uma estrutura onde clientes chegam para um serviço, podendo ou não formar uma fila caso o atendimento não seja imediato. Após o serviço, os clientes deixam o sistema. O termo "cliente"é usado de forma genérica e não implica necessariamente em uma pessoa.

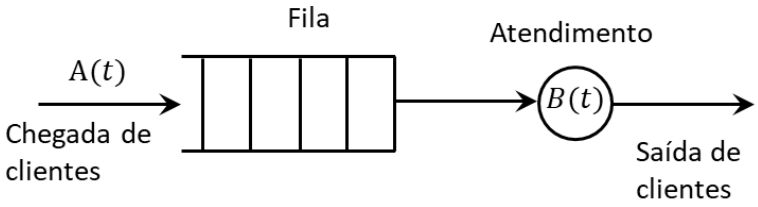

Figura 3 – Sistema de fila simples

Na literatura, (GROSS et al., 2018) destacam seis características fundamentais que definem os processos na teoria de filas. Para padronizar a classificação desses sistemas, David George Kendall propôs, em 1953, a notação de Kendall (KENDALL, 1953), que se tornou amplamente adotada na área. Segundo (RIBEIRO, 2018), essa notação permite descrever um sistema de filas de forma concisa por meio da seguinte representação:

$$A/B/m/k/z \tag{2.26}$$

em que:

- A representa a distribuição do tempo entre chegadas;
- B indica a distribuição do tempo de serviço;
- m é a quantidade de servidores no sistema;
- k define a capacidade do sistema, incluindo a fila e os servidores;
- z especifica a disciplina de atendimento utilizada na fila.

Se $k = \infty$ e $z = FIFO$ (First In, First Out), que é a disciplina em que o primeiro a chegar é o primeiro a ser atendido, os parâmetros k e z podem ser omitidos, reduzindo a notação para $A/B/m$. A Tabela 1 apresenta os símbolos mais comuns utilizados na teoria das filas e suas respectivas definições.

Tabela 1 – Notação de Kendall para sistemas de filas

Característica	**Símbolo**	**Explicação**
Distribuição do tempo entre chegadas (A)	M	Exponencial (Markoviano)
Distribuição do tempo de serviço (B)	D	Determinístico
	E_k	Erlang tipo k ($k = 1, 2, \ldots$)
	H_k	Hiper-exponencial (combinação de k exponenciais)
	G	Geral
Quantidade de servidores (m)	$1, 2, \ldots, \infty$	Número de servidores disponíveis
Capacidade máxima da fila (k)	$1, 2, \ldots, \infty$	Tamanho máximo permitido para a fila
Disciplina de atendimento (z)	FIFO	Primeiro a chegar, primeiro a ser atendido
	LCFS	Último a chegar, primeiro a ser atendido
	RSS	Seleção aleatória para serviço
	PR	Prioritário
	GD	Disciplina Geral

2.4.1 Desempenho de um sistema de fila

Conforme (MAIA, 2022) na maior parte dos sistemas os tempos de chegada, espera e de serviços são estocásticos, apresentando comportamento incerto. A chegada de clientes é modelada por uma sequência de eventos representada por $A(t)$ na Figura 3, que indica o número acumulado de chegadas até o instante t.

Para descrever os intervalos entre as chegadas consecutivas, considera-se a sequência de variáveis aleatórias $\{Y_1, Y_2, \ldots\}$, onde Y_k corresponde ao k-ésimo tempo entre chegadas, definido como:

$$Y_k = \text{Tempo decorrido entre a chegada } (k-1) \text{ e a chegada } k, \text{ para } k = 1, 2, \ldots. \tag{2.27}$$

De acordo com (CASSANDRAS; LAFORTUNE, 2008), na teoria de filas assume-se que os tempos entre chegadas são Independentes e Identicamente Distribuídos (IID), seguindo uma única distribuição de probabilidade que captura o comportamento dos intervalos. Quando Y_k segue uma distribuição exponencial, não é necessário indexar as chegadas, e a taxa média de chegadas (Y) é inversamente proporcional ao valor esperado de Y:

$$E[Y] = \frac{1}{\lambda} \tag{2.28}$$

Analogamente, o tempo de serviço também é modelado como uma sequência IID, com Z representando a variável aleatória associada ao tempo de serviço de cada cliente. Sua taxa média (μ) é dada por:

$$E[Z] = \frac{1}{\mu} \tag{2.29}$$

Com $A(t)$ como base para modelar o comportamento do sistema, outras métricas são definidas para avaliar o desempenho:

- A_k: é o tempo de chegada do k-ésimo cliente.
- D_k: é o tempo de saída do k-ésimo cliente.
- W_k: é o tempo de espera no sistema antes do início do atendimento.
- S_k: é o tempo total no sistema, levando em conta o tempo entre a chegada até a saída do cliente.
- Z_k: é o tempo de serviço do k-ésimo cliente.

Sendo assim, é possível determinar que:

$$S_k = D_k - A_k, \quad \text{ou} \quad S_k = W_k + Z_k. \tag{2.30}$$

2.4.1.1 Análise do Tempo de Espera e do Estado Estacionário

Idealmente, um sistema de filas deveria operar sem tempos de espera, garantindo que os clientes sejam atendidos imediatamente ao chegarem. No entanto, como discutido por (CASSANDRAS; LAFORTUNE, 2008), devido à aleatoriedade das chegadas e tempos de serviço, e à limitação da capacidade de atendimento, a formação de filas é inevitável. Isso leva à necessidade de métricas que permitam avaliar o desempenho do sistema em termos de espera e congestionamento.

Quando um sistema opera por um período suficientemente longo, ele pode atingir um estado estacionário, no qual as estatísticas de desempenho se estabilizam e tornam-se aproximadamente constantes ao longo do tempo. Nessas condições, podem ser calculadas métricas fundamentais, tais como:

- $E[W]$: tempo médio de espera dos clientes no estado estacionário.
- $E[S]$: tempo médio de permanência dos clientes no sistema.
- $E[X]$: comprimento médio da fila no estado estacionário.

O equilíbrio do sistema é determinado pela intensidade de tráfego (ρ), que expressa a relação entre a taxa média de chegada e a capacidade de atendimento do sistema. Esta métrica é dada por:

$$\rho = \frac{\lambda}{m\mu}, \tag{2.31}$$

onde m representa o número de servidores disponíveis no sistema. A condição necessária para garantir que o sistema permaneça estável é que $\rho < 1$, ou seja, a taxa de chegada deve ser menor que a capacidade máxima de atendimento do sistema.

De acordo com (GROSS et al., 2018), se $\rho \geq 1$, o sistema entra em um estado de saturação, onde a fila cresce indefinidamente. Em tais condições, a taxa de serviço não é suficiente para lidar com o volume de chegadas, resultando em congestionamento excessivo e aumento significativo dos tempos de espera e de permanência no sistema. Por essa razão, a otimização de sistemas de filas frequentemente busca ajustar parâmetros como número de servidores (m) e políticas de controle para manter ρ dentro de limites aceitáveis.

2.4.1.2 Lei de Little

A Lei de Little é um princípio fundamental na teoria de filas e sistemas de espera, desenvolvida pelo matemático John D. C. Little e publicada originalmente em 1961. De acordo com (SOUZA; MAIA, 2019) essa lei estabelece uma relação simples e poderosa entre três variáveis principais de um sistema estável:

- ❑ **Número médio de itens no sistema** (L): Representa a quantidade média de entidades (como clientes ou produtos) presentes no sistema, englobando tanto as que estão em fila quanto as que estão em serviço.
- ❑ **Taxa média de chegada** (λ): Refere-se à frequência com que as entidades chegam ao sistema por unidade de tempo.

- ❑ **Tempo médio de permanência no sistema** (W): Corresponde ao tempo médio que cada entidade passa no sistema, desde a chegada até a saída.

A Figura 4 ilustra o conceito geral da Lei de Little em um sistema típico de filas. Nessa representação, observa-se o fluxo de itens que entram no sistema, aguardam em fila, recebem o serviço, e eventualmente deixam o sistema.

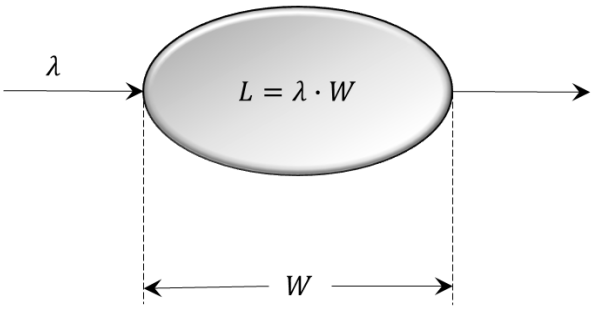

Figura 4 – Representação do sistema descrito pela lei de Little.

Considerando que W_q e W representam, respectivamente, o tempo médio de espera na fila e o tempo médio de permanência no sistema, as fórmulas de Little são definidas por:

$$L = \lambda W \quad \text{e} \quad L_q = \lambda W_q,$$

em que L e L_q correspondem, respectivamente, ao número médio de clientes no sistema e ao número médio de clientes esperando para serem atendidos.

Um resultado interessante que pode ser derivado das fórmulas de Little é a relação entre o número médio de clientes no sistema e o tempo de espera, expressa como:

$$L - L_q = \lambda(W - W_q) = \lambda\left(\frac{1}{\mu}\right) = \frac{\lambda}{\mu},$$

onde μ representa a taxa média de serviço.

A Tabela 2 resume os principais resultados encontrados para um sistema de filas $G/G/m$:

2.5 Filas com Clientes Impacientes

Um aspecto crucial na modelagem de sistemas que envolvem filas é a consideração de clientes impacientes. Conforme destacado por (GROSS et al., 2018), esses clientes tendem a ingressar na fila apenas se a espera estimada for curta e permanecem nela caso o tempo efetivo de espera seja tolerável. A impaciência gerada por períodos excessivos de espera é tão relevante para o desempenho do sistema quanto os processos de chegada e saída de clientes. Embora alguns estudos não levem esse fator em conta na análise de filas tradicionais, a impaciência dos clientes é um elemento indispensável para representar o sistema de maneira realista (WANG; LI; JIANG, 2010).

Tabela 2 – Resumo dos resultados gerais para o sistema $G/G/m$ de fila.

Fórmula	**Descrição**
$\rho = \lambda/m\mu$	Intensidade de tráfego
$L = \lambda W$	Número médio de clientes no sistema
$L_q = \lambda W_q$	Número médio de clientes na fila
$W = W_q + 1/\mu$	Tempo total de espera
$p_b = \lambda/m\mu = \rho$	Probabilidade de um servidor arbitrário estar ocupado
$r = \lambda/\mu$	Número esperado de clientes em serviço (servidores ocupados)
$p_0 = 1 - \rho$	Probabilidade de o sistema estar vazio

Fonte: Adaptado de (GROSS et al., 2018).

A paciência dos clientes é comumente classificada em duas categorias principais, conhecidas na literatura como Recusa (Balking) e Abandono (Reneging). Em muitos sistemas reais, os clientes podem percorrer diferentes etapas antes de receberem o serviço final, o que caracteriza uma rede de fila aberta. Nesse tipo de modelo, os clientes chegam ao sistema a partir de uma fonte externa, podem transitar entre múltiplas filas de atendimento e, eventualmente, sair do sistema após o serviço ou devido à impaciência. Esse conceito é amplamente utilizado para modelar sistemas como call centers multi-etapas, processos hospitalares e redes de comunicação.

Na literatura, diversos estudos sobre a impaciência dos clientes concentram-se em sistemas de call centers. Entre os principais trabalhos estão os de (ROUBOS; JOUINI, 2013), (WHITT, 2004), (YUE; W.; ZHAO, 2014), (AL-SEEDY et al., 2009) e (ABOU-EL-ATA; HARIRI, 1992), que analisam e modelam o comportamento impaciente dos clientes nesses ambientes.

2.5.1 Recusa (Balking)

De acordo com (GROSS et al., 2018), os clientes frequentemente se desmotivam a entrar em uma fila quando esta parece longa, preferindo evitar o sistema. (WANG; LI; JIANG, 2010) definem a recusa como o comportamento em que os clientes tomam suas decisões de entrada com base no tamanho observado da fila. Cada cliente possui um limite N, que representa o número máximo de clientes na fila que ele está disposto a tolerar antes de receber o serviço. Caso o comprimento da fila seja menor que N, o cliente ingressa no sistema; caso contrário, desiste antes mesmo de ingressar.

O comportamento de recusa pode ser modelado seguindo duas regras estabelecidas por (WANG; LI; JIANG, 2010):

- **Regra Tipo I** : O cliente possui um valor limite N de comprimento da fila. Com base nesse valor, ele decide se ingressa ou abandona o sistema.

- ❏ **Regra Tipo II** : A decisão do cliente depende de uma probabilidade p_n, onde N é o número de clientes no sistema no momento da chegada.

2.5.2 Abandono (Reneging)

Diferentemente da recusa, no abandono os clientes inicialmente entram na fila, mesmo que o tempo de espera possa parecer elevado. Contudo, à medida que aguardam, avaliam o tempo restante e podem decidir abandoná-la caso estimem que a espera excederá seu limite de tolerância (GROSS et al., 2018).

Segundo (WANG; LI; JIANG, 2010), as decisões de abandono são regidas por diferentes regras, variando conforme o contexto analisado na literatura. Seja T o tempo máximo de espera tolerado por um cliente:

- ❏ **Regra Tipo III** : O tempo de espera T é fixo e definido previamente.
- ❏ **Regra Tipo IV** : O tempo de espera T segue uma distribuição exponencial com parâmetro i, onde i representa o número de clientes no sistema.
- ❏ **Regra Tipo V** : O tempo de serviço segue uma distribuição lognormal, enquanto os tempos de abandono (espera) não são exponenciais.
- ❏ **Regra Tipo IV** : O cliente abandona o sistema quando a taxa de serviço é lenta ou ocorrem falhas frequentes no atendimento.

Além dessas, (WANG; LI; JIANG, 2010) destacam outras regras de abandono menos comuns, que incluem fatores como preferências pessoais ou percepções subjetivas dos clientes sobre o sistema. Entretanto, as regras mencionadas abrangem os cenários mais recorrentes de abandono.

2.6 Sistemas do Tipo Cliente-Servidor

Os sistemas do tipo cliente-servidor são amplamente observados em diversas áreas e representam cenários em que clientes aguardam em fila para serem atendidos por servidores. Exemplos clássicos incluem sistemas de suporte técnico online, filas de banco ou supermercado, serviços de nuvem para alocação de recursos computacionais e muitos outros. Esses sistemas são caracterizados pela espera dos clientes pelo serviço, o que inevitavelmente resulta, em alguns casos, no abandono da fila antes de serem atendidos. Um dos grandes desafios na modelagem desses sistemas está na representação da paciência dos clientes.

A compreensão do comportamento de clientes impacientes tem se tornado uma área de estudo relevante, especialmente para o aprimoramento de sistemas de call center. Atualmente,

muitas empresas utilizam esses serviços para resolução de problemas, atendimento ao cliente, marketing e outros processos diversos. Para garantir maior eficiência, é essencial estimar e modelar a paciência dos clientes, otimizando assim a operação do sistema e minimizando os impactos do abandono.

(ROUBOS; JOUINI, 2013) propõem, em seu estudo, o uso da distribuição hiperexponencial para modelar a paciência dos clientes. Essa distribuição é definida como uma combinação de duas distribuições exponenciais: com probabilidade p, a taxa associada é γ_1, enquanto, com probabilidade $(1-p)$, a taxa associada é γ_2. Essa abordagem permite representar, de maneira mais precisa, a variabilidade no comportamento dos clientes, abrangendo desde os mais pacientes até os mais impacientes.

A Figura 5 ilustra um exemplo de sistema cliente-servidor com abandono. Nesse sistema, dois tipos de clientes chegam ao sistema por meio dos eventos a_1 e a_2. Os clientes são admitidos no sistema pelos eventos c_1 e c_2, processados pelos servidores, e deixam o sistema após o atendimento, representados pelo evento d. Além disso, o abandono dos clientes é modelado pelos eventos b_1 e b_2.

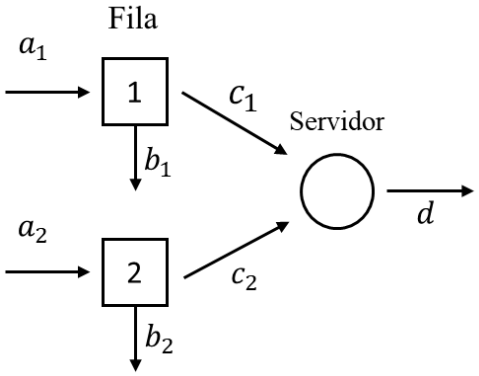

Figura 5 – Modelo de sistema cliente-servidor com abandono.

O modelo apresentado na figura ilustra como eventos de chegada, atendimento, saída e abandono interagem dentro do sistema, oferecendo uma representação clara e concisa da dinâmica de um sistema cliente-servidor. A compreensão e a modelagem adequada desses elementos são fundamentais para a análise e melhoria de sistemas reais.

2.6.1 Modelos Computacionais de Monte Carlo

Os modelos computacionais de Monte Carlo são amplamente utilizados para a análise e simulação de sistemas complexos, especialmente aqueles que apresentam comportamento estocástico. Essa abordagem baseia-se na utilização de geradores de números pseudoaleatórios e distribuições de probabilidade acumulativa que representam o comportamento do sistema em estudo. Frequentemente, essa metodologia também é denominada simulação estática, dada sua natureza independente do tempo.

De acordo com (MAIA, 2022), a simulação de Monte Carlo requer recursos computacionais consideráveis devido à sua abordagem intensiva, que depende de um grande número de amostras para fornecer resultados confiáveis. No entanto, sua implementação é relativamente simples quando aplicada a SED's estocásticas baseados em Markov, uma vez que utiliza a propriedade sem memória da distribuição exponencial e a geração de números pseudoaleatórios uniformes.

Uma das principais aplicações da simulação de Monte Carlo é a estimativa do valor esperado de uma variável aleatória relevante para o sistema modelado. Essa estimativa pode ser calculada utilizando a média aritmética das amostras geradas, como descrito na seguinte equação:

$$\hat{\theta} = \sum_{i=1}^{n} \frac{X_i}{n} \tag{2.32}$$

Nesta equação:

- $\hat{\theta}$: representa a estimativa do valor esperado da variável aleatória X;
- X_i : é a i-ésima amostra de X, gerada durante a simulação;
- n : é o número total de amostras geradas.

A média aritmética das amostras, como descrita, é uma das formas mais simples e eficazes de estimar o valor esperado em processos estocásticos. O uso dessa técnica permite que se obtenham estimativas precisas, desde que o número de amostras n seja suficientemente grande, minimizando o erro estatístico associado.

A eficiência e precisão dos modelos de Monte Carlo dependem diretamente da qualidade do gerador de números pseudoaleatórios e do tamanho da amostra utilizada. Em contrapartida, o alto custo computacional associado à geração e processamento de um grande volume de amostras é um dos principais desafios desta abordagem.

2.7 Otimização

Otimização é um campo de estudo que se dedica à busca das melhores soluções para problemas que envolvem a maximização ou minimização de uma função objetivo, sujeita a restrições impostas pelo sistema ou pelo ambiente. Trata-se de uma área interdisciplinar, com aplicações em engenharia, economia, logística, ciência da computação, entre outras (ARENALES et al., 2011).

Formalmente, um problema de otimização pode ser expresso como:

$$\text{Minimizar ou Maximizar} \quad f(x) \tag{2.33}$$

sujeito a:

$$g_i(x) \leq 0, \quad i = 1, 2, \ldots, m \tag{2.34}$$

$$h_j(x) = 0, \quad j = 1, 2, \ldots, p \tag{2.35}$$

onde:

- $f(x)$ é a função objetivo que se deseja otimizar;
- $x \in \mathbb{R}^n$ representa o vetor de variáveis de decisão;
- $g_i(x)$ são as restrições de desigualdade;
- $h_j(x)$ são as restrições de igualdade.

O objetivo é encontrar o vetor x^* que satisfaz as restrições e otimiza $f(x)$. Dependendo das características do problema, ele pode ser classificado como linear, não linear, contínuo, discreto, entre outros.

2.7.1 Tipos de Otimização

A otimização pode ser classificada em diferentes categorias, dependendo da natureza do problema e das técnicas utilizadas para resolvê-lo. A seguir, apresentam-se as principais abordagens.

2.7.1.1 Otimização Linear

A otimização linear (*Linear Programming* – LP) trata problemas em que tanto a função objetivo quanto as restrições são linearmente dependentes das variáveis de decisão. Um exemplo clássico é o problema de alocação de recursos, no qual se busca maximizar o lucro ou minimizar custos, respeitando restrições como capacidade produtiva e disponibilidade de insumos. Métodos eficientes, como o Simplex e a Programação Linear Inteira Mista (MILP), são amplamente utilizados para resolver esses problemas (VANDERBEI, 2020).

2.7.1.2 Otimização Não Linear

Quando a função objetivo ou pelo menos uma das restrições é não linear, o problema é classificado como otimização não linear (*Nonlinear Programming* – NLP). Esses problemas são significativamente mais complexos, pois podem apresentar múltiplos ótimos locais, regiões não convexas e restrições não diferenciáveis. Métodos iterativos, como gradiente descendente, métodos baseados em segunda ordem (Newton-Raphson) e algoritmos evolutivos, são frequentemente empregados para encontrar soluções aproximadas (BAZARAA; SHERALI; SHETTY, 2013).

2.7.1.3 Otimização Discreta

A otimização discreta envolve problemas em que as variáveis de decisão assumem valores de um conjunto discreto e finito, diferentemente da otimização contínua, onde as variáveis podem assumir qualquer valor dentro de um intervalo real. Essa classe de problemas surge frequentemente em aplicações industriais, logísticas e computacionais, como escalonamento de produção, alocação de recursos e roteamento de veículos (NEMHAUSER; WOLSEY, 1988; PAPADIMITRIOU; STEIGLITZ, 1998).

Muitos problemas de otimização discreta pertencem à classe NP-difícil, o que significa que não se conhece um algoritmo polinomial capaz de resolvê-los exatamente para todas as instâncias em tempo viável (PAPADIMITRIOU; STEIGLITZ, 1998). Exemplos clássicos incluem o Problema do Caixeiro Viajante (TSP), o Job Shop Scheduling (JSSP) e a otimização de controle em Redes de Petri. A dificuldade desses problemas decorre do crescimento exponencial do espaço de soluções à medida que o número de variáveis aumenta.

A alta complexidade desses problemas torna os métodos exatos, como Programação Inteira, Branch-and-Bound e Branch-and-Cut, inviáveis para instâncias grandes (PAPADIMITRIOU; STEIGLITZ, 1998). Assim, recorre-se a heurísticas e metaheurísticas, como a busca tabu (GLOVER; LAGUNA, 1997).

2.7.1.4 Otimização Multiobjetivo

Na otimização multiobjetivo, busca-se otimizar simultaneamente múltiplas funções objetivo, que muitas vezes são conflitantes entre si. Diferentemente da otimização mono-objetivo, onde existe uma única solução ótima, nesses casos define-se um conjunto de soluções de compromisso, conhecidas como soluções de Pareto.

O conceito de fronteira de Pareto é fundamental nesse contexto, pois representa o conjunto de soluções onde não é possível melhorar um objetivo sem piorar outro. Para encontrar essa fronteira, utilizam-se técnicas como Programação Multiobjetivo, Algoritmos Evolutivos Multiobjetivo (MOEA) e o método de ϵ-restrito. Dentre os algoritmos evolutivos mais utilizados, destacam-se o NSGA-II (Non-dominated Sorting Genetic Algorithm II) e o MOEA/D (Multi-Objective Evolutionary Algorithm based on Decomposition), que apresentam diferentes

estratégias para balancear exploração e diversidade na busca por soluções de Pareto (ZHANG; LI, 2009).

2.7.2 Métodos de Otimização

A escolha do método de otimização depende das características do problema, especialmente sua complexidade e estrutura matemática. Diferentes abordagens podem ser utilizadas, entre elas:

- **Métodos Analíticos:** Baseiam-se em derivadas e condições de otimalidade para encontrar soluções exatas, como os multiplicadores de Lagrange (BAZARAA; SHERALI; SHETTY, 2013).
- **Métodos Iterativos:** Aplicam algoritmos numéricos para encontrar soluções aproximadas, incluindo o método do gradiente descendente, Newton-Raphson e gradiente conjugado (NOCEDAL; WRIGHT, 2006).
- **Programação Dinâmica:** Resolve problemas por decomposição em subproblemas menores, sendo aplicada em modelos recursivos (NEMHAUSER; WOLSEY, 1988).
- **Heurísticas e Metaheurísticas:** Buscam soluções aproximadas para problemas complexos, utilizando técnicas como busca tabu (GLOVER; LAGUNA, 1997) e algoritmos genéticos (GOLDBERG, 1989).

2.7.3 Otimização por Busca Exaustiva

Conforme (KNUTH, 2011) a BE, também conhecida como *força bruta*, é um método de solução de problemas de otimização e decisão que consiste em explorar sistematicamente todas as combinações possíveis de soluções. Esse método garante encontrar a solução ótima, porém, muitas vezes, apresenta alta complexidade computacional, especialmente para problemas de grande dimensão.

2.7.3.1 Definição Formal

Dado um conjunto de soluções possíveis S e uma função objetivo $f(x)$, o problema de otimização pode ser definido como:

$$x^* = \arg\min_{x \in S} f(x) \quad \text{ou} \quad \arg\max_{x \in S} f(x). \tag{2.36}$$

onde x^* representa a solução ótima, isto é, o valor de x que minimiza ou maximiza $f(x)$ dentro do conjunto viável S. Em problemas de minimização, busca-se encontrar x^* tal que $f(x^*) \leq f(x)$ para todo $x \in S$. No caso de maximização, deseja-se um x^* que satisfaça $f(x^*) \geq f(x)$ para qualquer $x \in S$.

O algoritmo percorre S, avalia a função objetivo $f(x)$ para cada x, e seleciona a solução que resulta no valor mínimo ou máximo, conforme o objetivo definido.

O algoritmo de busca exaustiva pode ser descrito em alto nível da seguinte forma:

Início
Entrada: Um espaço de busca S finito e uma função objetivo $f : S \to \mathbb{R}$
Inicialize: $s^* \leftarrow$ null, $f^* \leftarrow \infty$
for cada solução $s \in S$ **do**
    Calcule $f(s)$
    if $f(s) > f^*$ **then**
        $s^* \leftarrow s$
        $f^* \leftarrow f(s)$
    end if
end for
Retorne: s^* e f^*
Fim

O algoritmo apresentado realiza uma busca exaustiva em um espaço de soluções S finito para encontrar a melhor solução s^*, conforme definido por uma função objetivo $f : S \to \mathbb{R}$.

Inicialmente, a variável s^* é definida como nula, e o valor f^* é inicializado como infinito, garantindo que qualquer solução válida será considerada na primeira iteração. O algoritmo então percorre todas as soluções possíveis no conjunto S, avaliando a função objetivo $f(s)$ para cada s. Se o valor calculado for maior que f^*, os valores de s^* e f^* são atualizados, armazenando a melhor solução encontrada até aquele momento. Esse processo continua até que todas as soluções tenham sido verificadas, garantindo que, ao final da execução, s^* contenha a melhor solução global e f^* o maior valor da função objetivo.

O espaço de busca S de um problema com n variáveis e d_i possíveis valores para cada variável x_i é definido como o produto cartesiano:

$$S = D_1 \times D_2 \times \cdots \times D_n, \tag{2.37}$$

onde D_i é o domínio da variável x_i. O tamanho total do espaço de busca é dado por:

$$|S| = \prod_{i=1}^{n} |D_i|. \tag{2.38}$$

A complexidade do método é diretamente proporcional ao tamanho do espaço de busca. Para um problema com $|S|$ combinações possíveis e uma função objetivo cujo tempo de avaliação é T_f, o tempo total de execução T é:

$$T = |S| \cdot T_f. \tag{2.39}$$

Embora a busca exaustiva garanta a solução ótima, sua viabilidade é limitada a problemas de pequena escala. Em problemas de grande dimensão, onde $|S|$ cresce fatorialmente, é necessário recorrer a métodos mais eficientes, como algoritmos heurísticos ou meta-heurísticos (NEMHAUSER; WOLSEY, 1988).

2.7.4 Otimização por Algoritmo Genético

Os algoritmos heurísticos inspirados em seres vivos baseiam-se na observação de processos naturais para resolver problemas complexos de otimização e busca. Esses algoritmos imitam comportamentos encontrados na natureza, como a evolução das espécies, o comportamento de enxames e a adaptação ao ambiente, para explorar e encontrar boas soluções em espaços de busca amplos. Entre as abordagens mais conhecidas estão os algoritmos evolutivos, os algoritmos de enxame de partículas e os sistemas imunológicos artificiais, que apresentam aplicações. (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2012)

O funcionamento de um algoritmo genético baseia-se na manutenção de uma população de soluções, que evolui iterativamente por meio da aplicação de operadores genéticos. A cada geração, indivíduos são selecionados com base em sua aptidão, passando por processos de recombinação e mutação para gerar novas soluções. Esse ciclo continua até que um critério de parada seja atingido, garantindo uma busca eficiente no espaço de soluções (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2012).

Dado um espaço de busca S, o objetivo é encontrar uma solução $x^* \in S$ que otimize uma função objetivo $f(x)$, que pode representar um custo ou um valor de aptidão desmonstrados na equação 2.36.

Os algoritmos genéticos são compostos pelos seguintes elementos básicos:

- ❏ Representação (Codificação): Cada solução do espaço de busca é representada por um indivíduo na população, comumente codificado como um vetor (ou cromossomo). A representação pode ser binária, real ou em outras formas dependendo do problema.
- ❏ População Inicial: Um conjunto de N indivíduos é gerado de forma aleatória ou com base em heurísticas, representando as soluções iniciais.
- ❏ Função de Aptidão: Avalia a qualidade de cada indivíduo na população em relação ao problema, determinando sua probabilidade de sobrevivência e reprodução. Geralmente, quanto maior a aptidão, maior a chance do indivíduo ser selecionado.
- ❏ Operadores Genéticos: São aplicados aos indivíduos para criar novos membros da população:

  - Seleção: Escolhe os indivíduos mais aptos para reprodução. Métodos comuns incluem seleção por roleta, torneio e seleção rankeada.
  - Crossover (Recombinação): Combina dois indivíduos (pais) para gerar novos indivíduos (filhos), promovendo a troca de informações genéticas.
  - Mutação: Introduz pequenas alterações em um ou mais indivíduos, explorando novas regiões do espaço de busca.

- ❑ Critérios de Parada: O algoritmo termina quando um critério pré-definido é atingido, como um número máximo de gerações, convergência da população ou qualidade suficiente da solução.

O funcionamento básico de um algoritmo genético pode ser descrito da seguinte forma:

Início
Entrada: População inicial P_0 com N indivíduos, taxa de crossover ρ_c, taxa de mutação ρ_m, função objetivo $f(x)$
Avalie os indivíduos da população inicial usando $f(x)$
for cada geração t **do**
    Selecione os indivíduos para reprodução com base em suas aptidões
    Aplique o crossover nos pares selecionados para gerar filhos
    Aplique mutação nos filhos com probabilidade ρ_m
    Avalie a aptidão dos novos indivíduos
    Atualize a população, mantendo os indivíduos mais aptos (elitismo opcional)
end for
Retorne: O melhor indivíduo da população final
Fim

O AG apresentado inicia-se com a definição de uma população inicial P_0 contendo N indivíduos, que representam soluções potenciais para o problema. Cada indivíduo é avaliado utilizando a função objetivo $f(x)$, que determina sua aptidão. Em seguida, o algoritmo entra em um ciclo de gerações sucessivas, onde são aplicados operadores genéticos para evoluir a população.

A cada geração, indivíduos são selecionados para reprodução com base em sua aptidão. Os pares selecionados passam pelo operador de crossover com uma taxa ρ_c, gerando novos indivíduos. Após o crossover, a mutação é aplicada com uma probabilidade ρ_m, introduzindo pequenas alterações nos filhos para manter a diversidade populacional. Os novos indivíduos gerados são avaliados pela função objetivo e inseridos na população.

Para evitar a perda de boas soluções, o algoritmo pode aplicar um mecanismo de elitismo, garantindo que os indivíduos mais aptos sejam mantidos na próxima geração. O ciclo de seleção, crossover, mutação e atualização da população continua até atingir um critério de parada, como

um número máximo de gerações. Por fim, o melhor indivíduo encontrado ao longo da execução é retornado como solução final do algoritmo.

A complexidade computacional de um AG é determinada principalmente por dois fatores:

- ❑ O tamanho da população (N), que representa o número de indivíduos avaliados a cada geração.
- ❑ O número de gerações (G), que define quantas iterações o algoritmo executa até atingir um critério de parada.

Além disso, há um terceiro fator, o custo computacional para avaliar a função objetivo T_f, que depende da complexidade da simulação utilizada para calcular a aptidão de cada indivíduo. No entanto, em muitos casos, T_f possui complexidade constante $O(1)$, pois o número de rodadas da simulação é fixo e independente de N e G. Assim, o tempo total de execução do algoritmo T pode ser expresso como:

$$T = G \cdot N \cdot T_f \tag{2.40}$$

Como $T_f = O(1)$, a complexidade assintótica do AG reduz-se a $O(G \cdot N)$, ou seja, cresce linearmente com o tamanho da população e o número de gerações.

Essa característica implica que, para problemas onde a simulação da função objetivo tem um custo fixo, o fator determinante para o desempenho do algoritmo genético é o número de indivíduos e gerações escolhidos. Dessa forma, estratégias para reduzir a complexidade computacional incluem ajustar o tamanho da população, limitar o número de gerações ou aplicar técnicas de paralelização para acelerar as avaliações.

3 Metodologia

3.1 Introdução

A metodologia adotada neste trabalho foi projetada para otimizar o desempenho de um sistema de filas do tipo cliente-servidor com abandono, modelado por uma RPT. A modelagem foi baseada em um estudo prévio de (MAIA, 2022), no qual um sistema semelhante foi representado por meio de linguagem de autômatos e analisado com base em processos estocásticos de salto Markoviano. A partir dessa abordagem, desenvolvemos uma modelagem equivalente em RPT, permitindo uma representação estruturada da dinâmica do sistema.

Após a modelagem inicial, foi implementado um controlador determinístico com o objetivo de reduzir a taxa de abandono sem comprometer a produtividade. Esse controle foi calculado a partir dos tempos médios das diferentes etapas da fila e introduziu dois parâmetros principais:

- M: Número de fichas no controlador, responsável por regular a admissão de clientes na fila.
- f: Temporização associada às transições, determinando o intervalo entre admissões no sistema.

Com essa estrutura, foram elaboradas duas funções objetivo (FOs) para avaliar a relação entre a taxa de atendimento e a taxa de abandono, considerando critérios distintos de otimização. Para obter os melhores valores de M e f, utilizamos BE e AG. A BE percorre todas as combinações possíveis dos parâmetros, enquanto o AG aplica operadores evolutivos para selecionar, recombinar e modificar soluções ao longo das iterações.

O modelo de simulação baseado em RPT avaliou o desempenho do sistema para diferentes combinações de M e f, gerando as métricas necessárias para a avaliação das FOs. Essas informações foram utilizadas para a adaptação iterativa dos parâmetros do controle conforme a estratégia de otimização escolhida.

Nesta seção, serão detalhadas as etapas de implementação do modelo, incluindo a formulação das FOs, os métodos de otimização empregados, os parâmetros gerais do AG, os operadores genéticos aplicados e o método utilizado para a seleção dos indivíduos mais promissores.

3.2 Modelagem Rede de Petri

O sistema de fila cliente-servidor com abandono foi inicialmente modelado utilizando uma Rede de Petri básica, que corresponde diretamente à estrutura do sistema de fila original. Nessa modelagem inicial (Figura 6), os lugares e transições foram configurados para representar os estados do sistema, como:

- ❏ Admissão de clientes: Representado pela transição u, que posteriormente insere fichas nos lugares da fila através da transição a.
- ❏ Fila: Representada pelo lugar P_1, que acumulam fichas, indicando clientes em espera.
- ❏ Atendimento: Representado pela transição c, que consome fichas da fila e os processa no servidor.
- ❏ Saída: Fichas processados são removidos do sistema após o atendimento ou abandono.
- ❏ Abandono: Representado pela transição q, que corresponde ao abandono do cliente na fila. Nesse caso, uma ficha retorna para o lugar P_2, permitindo que outra ficha possa ser admitida na fila, sem ser contabilizada como atendimento.

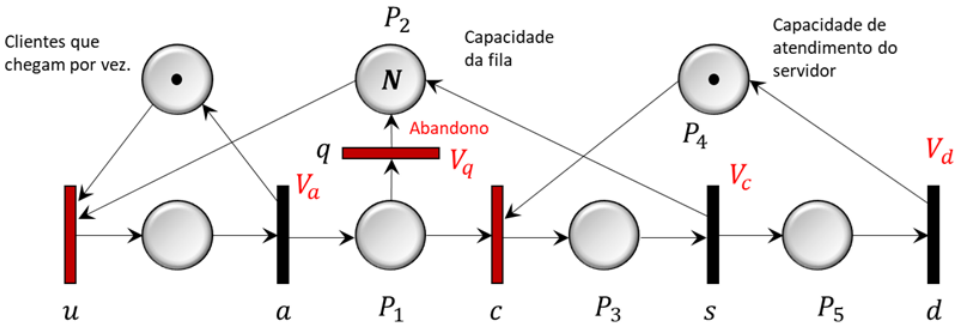

Figura 6 – Modelo inicial da Rede de Petri representando o sistema de fila cliente-servidor com abandono.

Embora funcional, o modelo inicial não capturava aspectos temporais como os tempos de admissão, espera, atendimento ou abandono, limitando sua utilidade para análise de desempenho.

3.2.1 Expansão do Modelo para Implementação de Temporização

Para incorporar a temporização aos eventos do sistema, o modelo inicial foi expandido, como ilustrado na Figura 7. Essa modificação inclui transições e lugares auxiliares, permitindo a representação explícita dos tempos associados às principais atividades do sistema, como tempo de espera, atendimento e abandono.

Na nova representação da RPT, onde havia a transição c foram acrescentados dois lugares e duas transições, representados pelos lugares P_3 e P_5 e pelas transições C_1 e S_d, cujo os tempos associados, representam o deslocamento da ficha, saindo da fila em direção ao servidor. Uma vez que a transição S_d é disparada, consideramos que já se deu início ao atendimento do cliente. A transição P_d também foi expandida para incluir dois novos lugares P_6 e P_7 e as transições S_1 e S_2. Essas adições representam o tempo de atendimento e o deslocamento das fichas em direção à saída do sistema.

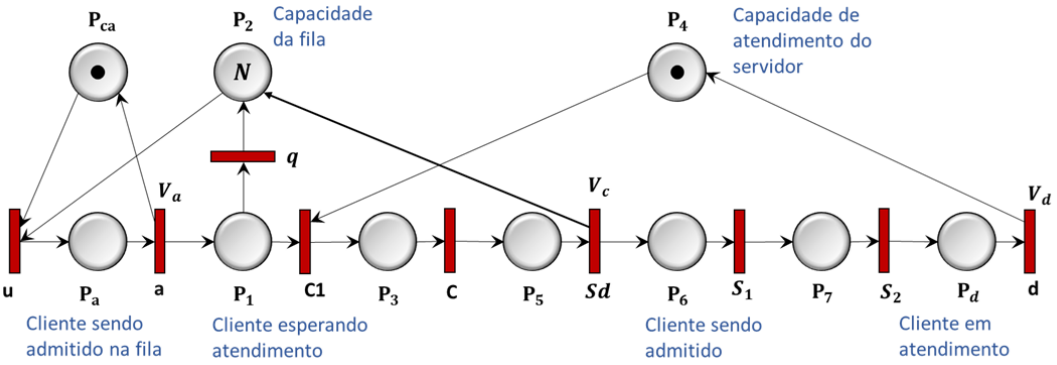

Figura 7 – Modelo expandido da Rede de Petri para implementação de temporização no sistema de fila cliente-servidor.

Essa reformulação foi essencial para permitir a análise de desempenho do sistema com base nos tempos reais de operação, viabilizando estudos mais detalhados sobre o comportamento do sistema em diferentes cenários.

3.2.2 Condição Inicial da rede de Petri temporizada

A condição inicial de uma RPT define o estado inicial dos lugares, ou seja, a quantidade de fichas presentes em cada lugar no início da simulação. Essa condição é fundamental, pois influencia diretamente o comportamento da rede à medida que as transições são disparadas. A condição inicial é representada pelo vetor x_0, onde cada elemento $X_0(P_i)$ corresponde ao número de fichas presentes no lugar P_i no momento inicial. (GROSS et al., 2018)

$$\begin{aligned} x(P_a) = x(P_1) = x(P_3) = x(P_5) = x(P_6) = x(P_7) = x(P_d) = 0, \\ x(P_{ca}) = x(P_4) = 1, \\ x(P_2) = 10 \end{aligned} \tag{3.1}$$

Sendo definido pelo vetor:

$$\begin{aligned} \mathbf{x}_0 &= [P_a\ P_1\ P_3\ P_4\ P_5\ P_6\ P_7\ P_{ca}\ P_d\ P_2]\,, \\ \mathbf{x}_0 &= [0\,0\,0\,1\,0\,0\,0\,1\,0\ N] \end{aligned} \tag{3.2}$$

3.2.3 Matriz de Incidência da RPT

A matriz de incidência A é um elemento fundamental para representar a dinâmica das transições em uma RP. Cada linha dessa matriz corresponde a um lugar P, e cada coluna corresponde a uma transição τ . Os elementos dessa matriz indicam como uma ficha se move de um lugar para outro quando uma transição ocorre. Se a transição τ_i adiciona uma ficha ao lugar P_j , o elemento A_ij será positivo, indicando uma entrada de ficha. Caso a transição remova uma ficha de um lugar, o valor A_ij será negativo, representando a saída de uma ficha. A seguir, detalhamos o equacionamento dos estados da rede a cada transição.

Tabela 3 – Tabela detalhada do equacionamento da rede de Petri com abandono - primeira parte

$\boldsymbol{\tau \times p}$	$\boldsymbol{u}$	$\boldsymbol{a}$	$\boldsymbol{c1}$
P_a	$x'(P_a) = x(P_a) + 1$	$x'(P_a) = x(P_a) - 1$	$x'(P_a) = x(P_a) + 0$
P_1	$x'(P_1) = x(P_1) + 0$	$x'(P_1) = x(P_1) + 1$	$x'(P_1) = x(P_1) - 1$
P_3	$x'(P_3) = x(P_3) + 0$	$x'(P_3) = x(P_3) + 0$	$x'(P_3) = x(P_3) + 1$
P_4	$x'(P_4) = x(P_4) + 0$	$x'(P_4) = x(P_4) + 0$	$x'(P_4) = x(P_4) - 1$
P_5	$x'(P_5) = x(P_5) + 0$	$x'(P_5) = x(P_5) + 0$	$x'(P_5) = x(P_5) + 0$
P_6	$x'(P_6) = x(P_6) + 0$	$x'(P_6) = x(P_6) + 0$	$x'(P_6) = x(P_6) + 0$
P_7	$x'(P_7) = x(P_7) + 0$	$x'(P_7) = x(P_7) + 0$	$x'(P_7) = x(P_7) + 0$
P_{ca}	$x'(P_{ca}) = x(P_{ca}) - 1$	$x'(P_{ca}) = x(P_{ca}) + 1$	$x'(P_{ca}) = x(P_{ca}) + 0$
P_d	$x'(P_d) = x(P_d) + 0$	$x'(P_d) = x(P_d) + 0$	$x'(P_d) = x(P_d) + 0$
P_2	$x'(P_2) = x(P_2) - 1$	$x'(P_2) = x(P_2) + 0$	$x'(P_2) = x(P_2) + 0$

Tabela 4 – Tabela detalhada do equacionamento da rede de Petri com abandono - segunda parte (continuação)

$\boldsymbol{\tau \times p}$	$\boldsymbol{c}$	$\boldsymbol{sd}$	$\boldsymbol{s1}$
P_a	$x'(P_a) = x(P_a) + 0$	$x'(P_a) = x(P_a) + 0$	$x'(P_a) = x(P_a) + 0$
P_1	$x'(P_1) = x(P_1) + 0$	$x'(P_1) = x(P_1) + 0$	$x'(P_1) = x(P_1) + 0$
P_3	$x'(P_3) = x(P_3) - 1$	$x'(P_3) = x(P_3) + 0$	$x'(P_3) = x(P_3) + 0$
P_4	$x'(P_4) = x(P_4) + 0$	$x'(P_4) = x(P_4) + 0$	$x'(P_4) = x(P_4) + 0$
P_5	$x'(P_5) = x(P_5) + 1$	$x'(P_5) = x(P_5) - 1$	$x'(P_5) = x(P_5) + 0$
P_6	$x'(P_6) = x(P_6) + 0$	$x'(P_6) = x(P_6) + 1$	$x'(P_6) = x(P_6) - 1$
P_7	$x'(P_7) = x(P_7) + 0$	$x'(P_7) = x(P_7) + 0$	$x'(P_7) = x(P_7) + 1$
P_{ca}	$x'(P_{ca}) = x(P_{ca}) + 0$	$x'(P_{ca}) = x(P_{ca}) + 0$	$x'(P_{ca}) = x(P_{ca}) + 0$
P_d	$x'(P_d) = x(P_d) + 0$	$x'(P_d) = x(P_d) + 0$	$x'(P_d) = x(P_d) + 0$
P_2	$x'(P_2) = x(P_2) + 0$	$x'(P_2) = x(P_2) + 1$	$x'(P_2) = x(P_2) + 0$

Com base no equacionamento matemático da RPT, apresentado nas Tabelas 3, 4 e 5, que foram divididas em três partes para facilitar a organização e visualização, foi possível derivar a matriz de incidência A. Essa matriz descreve de forma detalhada as interações entre os lugares e as transições do sistema, conforme ilustrado na Equação 3.3.

Tabela 5 – Tabela detalhada do equacionamento da rede de Petri com abandono - terceira parte (continuação)

$\boldsymbol{\tau \times p}$	$\boldsymbol{s2}$	$\boldsymbol{d}$	$\boldsymbol{q}$
P_a	$x'(P_a) = x(P_a) + 0$	$x'(P_a) = x(P_a) + 0$	$x'(P_a) = x(P_a) + 0$
P_1	$x'(P_1) = x(P_1) + 0$	$x'(P_1) = x(P_1) + 0$	$x'(P_1) = x(P_1) - 1$
P_3	$x'(P_3) = x(P_3) + 0$	$x'(P_3) = x(P_3) + 0$	$x'(P_3) = x(P_3) + 0$
P_4	$x'(P_4) = x(P_4) + 0$	$x'(P_4) = x(P_4) + 1$	$x'(P_4) = x(P_4) + 0$
P_5	$x'(P_5) = x(P_5) + 0$	$x'(P_5) = x(P_5) + 0$	$x'(P_5) = x(P_5) + 0$
P_6	$x'(P_6) = x(P_6) + 0$	$x'(P_6) = x(P_6) + 0$	$x'(P_6) = x(P_6) + 0$
P_7	$x'(P_7) = x(P_7) - 1$	$x'(P_7) = x(P_7) + 0$	$x'(P_7) = x(P_7) + 0$
P_{ca}	$x'(P_{ca}) = x(P_{ca}) + 0$	$x'(P_{ca}) = x(P_{ca}) + 0$	$x'(P_{ca}) = x(P_{ca}) + 0$
P_d	$x'(P_d) = x(P_d) + 1$	$x'(P_d) = x(P_d) - 1$	$x'(P_d) = x(P_d) + 0$
P_2	$x'(P_2) = x(P_2) + 0$	$x'(P_2) = x(P_2) + 0$	$x'(P_2) = x(P_2) + 1$

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

Para calcular a matriz A^-, é necessário identificar todos os pontos de saída de fichas, representando as mudanças negativas no estado do sistema. A matriz A^- descreve as retiradas de fichas de cada lugar P_i quando uma transição τ_j ocorre e é dada por:

$$A^- = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

A dinâmica da RPT é regida por duas equações principais:

$$x' = xAu \tag{3.5}$$

$$x - A^{-}u \geq [0] \tag{3.6}$$

Na Equação (3.5), x é o vetor de estado atual, A é a matriz de incidência, e u é o vetor de disparos das transições. A atualização do estado, x', é obtida somando o efeito das transições disparadas ao estado atual.

Já a inequação (3.6), garante que uma transição só será habilitada se houver fichas suficientes nos lugares de entrada. O vetor u ndica quais transições são disparadas, enquanto A^{-} representa as retiradas de fichas associadas a cada transição. Essas equações permitem modelar o comportamento dinâmico da RPT, definindo as transições habilitadas e os estados subsequentes do sistema, de forma a garantir a consistência das operações e a validade das condições de disparo.

3.2.4 Modelo computacional de Monte Carlo

No modelo desenvolvido, os tempos associados às transições foram definidos utilizando valores de λ, que determinam as taxas exponenciais de disparo correspondentes. Esses valores estão diretamente relacionados ao comportamento estocástico do sistema, em que cada transição é disparada após um tempo aleatório governado por uma distribuição exponencial. Nessas condições, o valor esperado pode ser estimado pela fórmula:

$$E[x] \approx \bar{x} = \frac{1}{n}\sum_{n}^{i=1} x_i \tag{3.7}$$

Para simular essa dinâmica, utilizou-se um método baseado em Monte Carlo, permitindo a geração de tempos aleatórios associados às transições habilitadas. Esse método emprega números aleatórios U gerados a partir de uma distribuição uniforme no intervalo $(0, 1]$, garantindo que nunca assumam o valor zero. O tempo até o próximo disparo é então calculado por:

$$T_i = \frac{-1}{\lambda_i} ln(u) \tag{3.8}$$

sendo que λ_i é a taxa de disparo da transição i e u é um número aleatório estritamente positivo, evitando problemas computacionais associados ao cálculo do logaritmo natural.

Para transições desabilitadas, atribui-se um tempo artificialmente elevado, garantindo que essas transições não sejam disparadas enquanto permanecerem inativas. Dessa forma, o sistema considera apenas as transições habilitadas no momento atual, mantendo a coerência da simulação.

Além disso, para determinadas transições, como U e C', adotou-se um valor de $\lambda = 10^{100}$, resultando em um inverso extremamente pequeno (10^{-100}). Essa configuração faz com que o

tempo de disparo gerado pela distribuição exponencial seja próximo de zero, assegurando que tais transições ocorram praticamente de forma imediata assim que forem habilitadas. Essa técnica é amplamente utilizada na modelagem de RP para representar eventos sem latência perceptível, como a ativação condicional de transições ou a movimentação imediata de fichas entre determinados lugares.

Em cada etapa da simulação, o processo segue as seguintes etapas:

- ❏ Identificação das transições habilitadas: Verifica-se a condição $x - A^{-}u \geq [0]$, onde A^{-} é a matriz de incidência negativa, $\bar{x}$ é o vetor de marcação atual e u é o vetor de disparo.
- ❏ Geração dos tempos aleatórios: Calculam-se os valores T_i para as transições habilitadas.
- ❏ Seleção da transição com menor tempo: A transição com o menor tempo associado é escolhida para disparo.
- ❏ Atualização do estado: O estado da Rede de Petri é atualizado por meio da equação de estado previamente definida.

Os valores temporais associados às transições estão detalhados na tabela 6, que apresenta os valores de inverso de λ (*Inv_L*) atribuídos a cada transição do modelo.

Tabela 6 – Valores temporais das transições.

Transição	**Inverso de λ (Inv_L)**
U	10^{-100}
A	1.4
C'	10^{-100}
C	0.1
Sd	0.443
S1	0.443
S2	0.443
D	1.171
q	$14/$n° de fichas na fila

3.2.5 RPT Controlada

Após a modelagem inicial da RP, foi implementado um controlador projetado especificamente para reduzir o tempo de espera no sistema e, consequentemente, minimizar a taxa de abandono dos clientes. Esse controle foi desenvolvido com base na parametrização de dois elementos fundamentais: o número de fichas no controlador (M) e a temporização associada (f). Esses parâmetros foram ajustados para maximizar a eficiência do sistema, garantindo um equilíbrio entre produtividade e qualidade de serviço.

A introdução do controlador exigiu a modificação estrutural da RP original, com a adição de um novo lugar dedicado ao controle. Esse lugar adicional foi integrado ao modelo como um componente que regula a admissão de clientes na fila, limitando o número de clientes que podem ocupar o sistema ao mesmo tempo. Consequentemente, a matriz de incidência da rede também precisou ser expandida, resultando na inclusão de uma nova linha que representa o impacto desse novo lugar nas transições.

O ajuste do controlador considera duas variáveis principais, o número de fichas no controlador (M) e a temporização associada à entrada do sistema (f). Esses parâmetros são definidos com base no maior tempo médio necessário para percorrer uma sequência crítica do sistema, representado por E_{max}. Esse valor é calculado considerando as taxas de chegada, processamento e saída dos clientes, conforme a expressão:

$$E_{max} = max\left(\frac{1}{\lambda_a}, \frac{\frac{1}{\lambda_a}+\frac{1}{\lambda_c}}{2}, \left(\frac{1}{\lambda_{Sd}}+\frac{1}{\lambda_{S1}}+\frac{1}{\lambda_{S2}}+\frac{1}{\lambda_d}\right)\right) \tag{3.9}$$

Essa equação foi obtida a partir da análise dos principais tempos médios do sistema, levando em consideração três fases principais do atendimento:

- ❑ Tempo entre chegadas: Representado por $\frac{1}{\lambda_a}$, corresponde ao tempo médio esperado entre duas chegadas consecutivas de clientes no sistema.
- ❑ Tempo médio de admissão: Considerando que um cliente, ao chegar, pode ser admitido dependendo da disponibilidade do servidor, esse tempo foi estimado pela média entre o tempo de chegada e o tempo médio de processamento inicial, resultando em $\frac{\frac{1}{\lambda_a}+\frac{1}{\lambda_c}}{2}$.
- ❑ Tempo total de atendimento e saída do sistema: Os clientes podem passar por diferentes estágios do atendimento, representados pelas transições associadas ao serviço e à saída. O tempo total dessa etapa é a soma dos tempos médios dos subprocessos individuais:

$$\frac{1}{\lambda_{Sd}}+\frac{1}{\lambda_{S1}}+\frac{1}{\lambda_{S2}}+\frac{1}{\lambda_d} \tag{3.10}$$

A partir do cálculo de E_{max}, os valores de M e f são definidos para garantir um funcionamento eficiente da rede controlada. O número de fichas do controlador é definido pela seguinte relação:

$$M = \left(\frac{\frac{1}{\lambda_a}+\frac{1}{\lambda_{C1}}+\frac{1}{\lambda_C}+\frac{1}{\lambda_{Sd}}+\frac{1}{\lambda_{S1}}+\frac{1}{\lambda_{S2}}+\frac{1}{\lambda_d}}{E_{max}}\right) \tag{3.11}$$

Já a temporização f foi ajustada com base no maior tempo esperado do sistema e na influência dos parâmetros do modelo:

$$f = E_{max} \cdot M - \frac{1}{\lambda_a} - \left(\frac{1}{\lambda_C}+\frac{1}{\lambda_{C1}}\right) - \left(\frac{1}{\lambda_{Sd}}+\frac{1}{\lambda_{S1}}+\frac{1}{\lambda_{S2}}+\frac{1}{\lambda_d}\right) \tag{3.12}$$

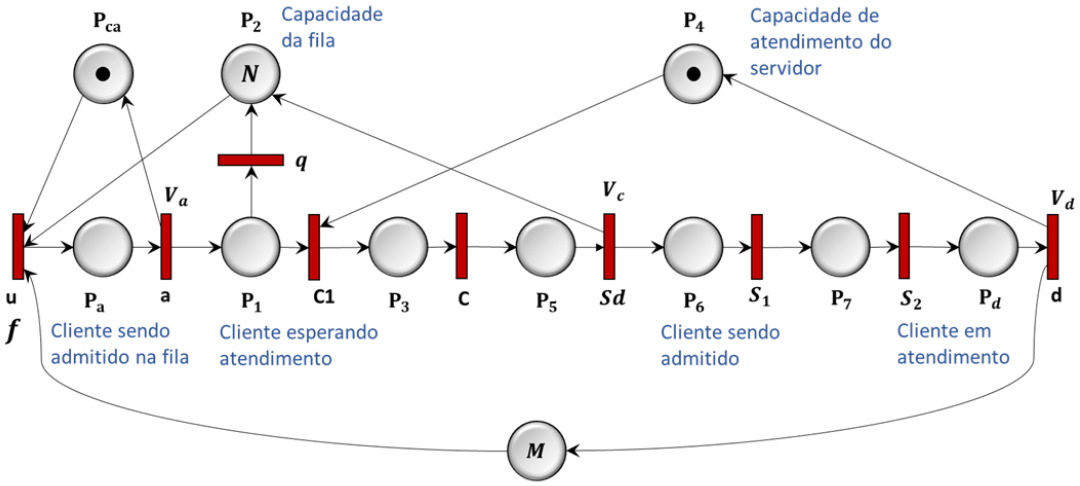

Figura 8 – Rede de Petri com sistema de controle

A lógica por trás dessas fórmulas garante que o sistema opere de forma otimizada, regulando o fluxo de entrada de clientes e ajustando a temporização de maneira que o sistema mantenha um equilíbrio entre produtividade e qualidade de serviço. O parâmetro E_{max} considera os maiores tempos médios do sistema, garantindo que o controlador seja capaz de ajustar o comportamento do modelo para evitar sobrecarga e minimizar a taxa de abandono.

Com essas definições, o sistema de controle foi implementado, utilizando os valores de M e f, de modo a minimizar o tempo de espera e a taxa de abandono, sem comprometer a capacidade produtiva.

A Figura 8 ilustra a Rede de Petri com o controlador integrado, evidenciando as mudanças estruturais implementadas e o impacto dessas alterações no comportamento do sistema. Essa integração foi essencial para alcançar os objetivos de otimização e redução de perdas no sistema.

3.3 Otimização

3.3.1 Formulação da Função Objetivo

Para avaliar o desempenho do sistema, relacionamos a taxa de atendimento (Tx_{Atend}) e a taxa de abandono (Tx_{Aband}). Essas métricas fornecem uma visão sobre a eficiência do sistema, indicando a capacidade de atender demandas e os impactos do abandono de clientes no processo.

Com base nessa relação, expressamos a avaliação do sistema em termos de lucro, definindo funções que capturam o rendimento geral considerando o equilíbrio entre atendimento e abandono. Foram desenvolvidas duas funções objetivo distintas para essa análise.

A primeira função objetivo considera o lucro como a diferença entre a taxa de atendimento e uma versão penalizada da taxa de abandono. Esse modelo reflete diretamente o impacto das desistências sobre o desempenho do sistema:

$$L(M, f) = \text{Tx}_{\text{Atend}} - \alpha \cdot \text{Tx}_{\text{Aband}} \cdot \beta \tag{3.13}$$

Já a segunda função objetivo avalia o lucro com base na razão entre a taxa de atendimento e a taxa de abandono, destacando o equilíbrio relativo entre essas métricas como um fator determinante na otimização do sistema:

$$L(M, f) = \text{Tx}_{\text{Atend}} / \text{Tx}_{\text{Aband}} \tag{3.14}$$

onde:

- M representa o número de fichas no controlador;
- f representa o tempo de admissão no sistema;
- Tx_{Atend} corresponde à taxa de atendimento do sistema;
- Tx_{Aband} representa a taxa de abandono;
- α é um fator de penalização, responsável por atribuir maior peso à minimização da taxa de abandono na primeira função objetivo;
- β é um fator de escala utilizado para ajustar a FO.

Essa abordagem permite comparar diferentes estratégias de avaliação do lucro, possibilitando a análise do impacto do controle sobre o desempenho do sistema sob diferentes critérios. As duas funções foram aplicadas de forma independente no processo de otimização, permitindo identificar configurações de M e f que maximizam os resultados de acordo com cada métrica adotada.

3.3.1.1 Critérios para a Escolha da Penalização

Para determinar um valor adequado para α, foram realizados testes experimentais analisando diferentes valores do parâmetro e observando seu impacto no custo final. A definição da penalização foi guiada pelos seguintes critérios:

- Manutenção da Produtividade – O sistema deve continuar operando com uma taxa de atendimento elevada, evitando restrições excessivas na admissão de clientes.
- Minimização do Abandono – A função objetivo deve priorizar configurações que reduzam a taxa de abandono de maneira significativa.
- Equilíbrio entre as Métricas – Deve haver um balanço adequado entre a taxa de atendimento e a taxa de abandono, assegurando que o controle não comprometa completamente a eficiência do sistema.

Os valores de M e f que maximizam a função objetivo correspondem às configurações que oferecem o melhor equilíbrio entre atendimento e abandono, garantindo um funcionamento eficiente do sistema.

Adicionalmente, a definição de uma segunda função objetivo baseada na razão entre a taxa de atendimento e a taxa de abandono foi motivada pela necessidade de uma métrica alternativa que não dependesse da escolha de um fator de penalização. Essa formulação permite avaliar diretamente o desempenho relativo do sistema, destacando configurações em que a taxa de atendimento se mantém alta mesmo diante de níveis baixos de abandono. Dessa forma, a comparação entre as duas funções possibilita uma análise mais abrangente dos efeitos do controle sobre o comportamento do sistema.

3.3.1.2 Aplicação na Otimização

A escolha das técnicas de otimização foi motivada pela necessidade de avaliar o impacto do controle no desempenho do sistema e pela possibilidade de explorar diferentes abordagens para a obtenção de soluções. A BE foi utilizada por garantir a solução ótima, permitindo uma análise completa do espaço de busca, enquanto o AG possibilitou a aplicação de técnicas metaheurísticas para avaliar o desempenho do sistema, incorporando métodos de otimização em simulações baseadas em RPT. Essas abordagens permitem comparar soluções exatas e heurísticas, analisando suas vantagens e limitações no contexto do problema estudado.

As funções objetivo foram aplicadas nos dois métodos de otimização implementados:

- **Busca Exaustiva (BE)** – Permitiu testar todas as combinações possíveis de M e f dentro de um intervalo previamente definido, garantindo a obtenção da solução ótima para cada função objetivo analisada.
- **Algoritmo Genético (AG)** – Utilizou a função objetivo como critério de seleção, orientando a evolução da população e identificando configurações com maior potencial de desempenho por meio de operadores de seleção, cruzamento e mutação.

3.3.2 Busca Exaustiva

A busca exaustiva é uma técnica de otimização que consiste em avaliar sistematicamente todas as combinações possíveis de parâmetros dentro de um espaço de busca finito e limitado. Por meio dessa abordagem, é possível identificar a solução que maximiza (ou minimiza) a função objetivo, desde que o número de combinações seja viável computacionalmente. Apesar de garantir a obtenção da solução ótima, sua aplicação é restrita a problemas de pequena escala, devido ao crescimento exponencial do número de possibilidades à medida que os parâmetros aumentam.

Neste trabalho, foi desenvolvida uma rotina iterativa no algoritmo de simulação da RPT, permitindo ajustar, a cada rodada, os valores de M (número de fichas no controlador) e f (temporização associada às transições). Para cada combinação desses parâmetros, foi calculado um custo com base em uma função objetivo apresentada na equação 3.13.

Os custos calculados para todas as combinações possíveis de M e f foram armazenados em uma matriz, onde cada elemento correspondia ao custo associado a um par específico de valores de M e f.

No caso específico deste estudo, o método avaliou a matriz de custos, comparando cada elemento para identificar o menor valor de custo e seus respectivos índices, correspondentes a M e f.

3.3.3 Algoritmo Genético

O AG foi implementado com o objetivo de otimizar os valores de M e f na simulação da rede. O processo iniciou-se com a definição dos intervalos de busca para os parâmetros, com $M \in [2, 10]$ e $f \in [0, 4]$. No pseudocódigo visão geral do AG, para realizar as simulações e avaliações, foram utilizadas duas funções principais: `simular_rede(`M, f`)`, que executa a simulação da rede para cada indivíduo, e `funcao_avaliacao(`M, f`)`, responsável por calcular o custo associado, considerando a taxa de atendimento e penalizações relacionadas à taxa de abandono.

A população inicial foi gerada aleatoriamente, com cada indivíduo representando um par (M, f). Os valores de M foram gerados aleatoriamente dentro do intervalo definido, enquanto f foi arredondado para uma casa decimal para manter coerência com os experimentos anteriores. Essa abordagem permitiu uma distribuição uniforme dos valores iniciais, garantindo diversidade no espaço de busca.

A cada iteração, a população foi avaliada para calcular os custos, com base nos quais os melhores indivíduos foram selecionados utilizando o método de torneio. Esse método consiste em escolher aleatoriamente dois indivíduos e comparar seus custos, selecionando aquele que apresenta a melhor avaliação. Esse processo foi repetido até preencher a nova população. A escolha desse método se deve ao fato de que ele equilibra exploração e exploração, reduzindo a probabilidade de convergência prematura para máximos locais.

Após a seleção, os indivíduos foram submetidos a um operador de cruzamento, aplicado com uma taxa de recombinação de $P_c = 0,8$. Para cada par de pais, geraram-se dois descendentes, combinando os valores de M e f. O cruzamento foi projetado utilizando um ponto de corte único, onde parte dos genes de um indivíduo foi trocada com os do outro como no exemplo da figura 9.

Em seguida, foi aplicada a mutação com probabilidade $P_m = 20\%$. A mutação alterou aleatoriamente os genes de f, respeitando os limites definidos para os parâmetros. Essa etapa

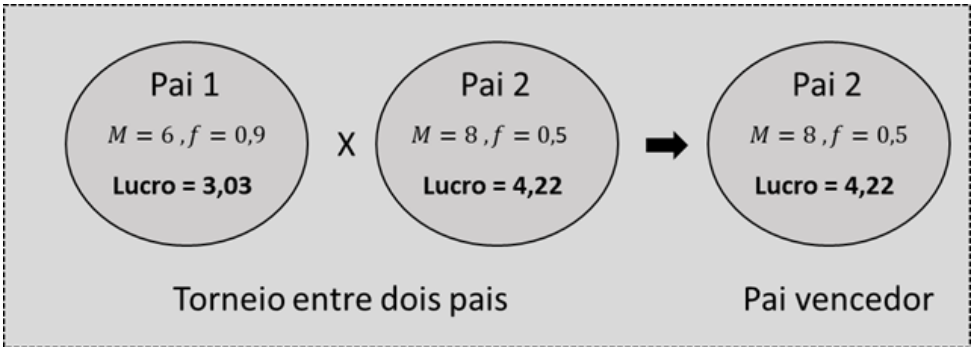

Figura 9 – Exemplo de uma competição de torneio entre dois pais, onde foi escolhido o pai que representa o menor valor de custo

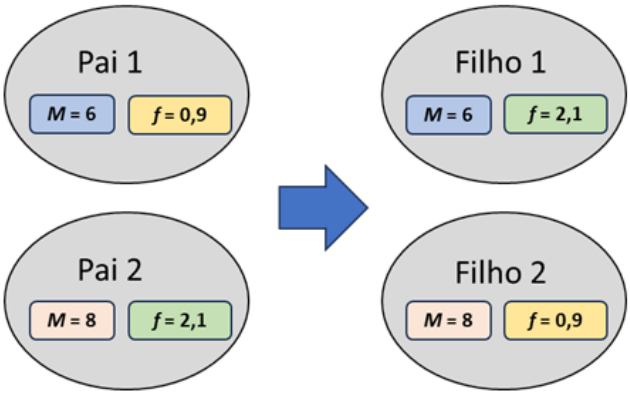

Figura 10 – Exemplo dos filhos com novos valores de M e f gerados dos pais.

contribuiu para a manutenção da diversidade da população e evitou a estagnação em ótimos locais.

A nova população foi composta pelos indivíduos gerados por cruzamento e mutação, juntamente com os melhores da geração anterior, conforme estratégia de elitismo. Esse processo foi repetido até que o critério de parada fosse atingido, definido pelo número máximo de gerações ($N_{\max}$).

Ao final do processo, o algoritmo retornou o par (M^*, f^*) correspondente à melhor avaliação encontrada.

Parte I

Resultados

4 Resultados

4.1 Apresentação dos Resultados

Neste capítulo, são apresentados os principais resultados obtidos a partir da aplicação das técnicas de otimização sobre o sistema modelado por uma RPT com controle determinístico. O objetivo é analisar o impacto da estratégia de controle sobre o desempenho do sistema, em especial na redução da taxa de abandono e na manutenção da produtividade, com base nas duas funções objetivo definidas.

Os parâmetros de controle M e f, foram otimizados utilizando dois métodos, sendo o BE, que garantiu a obtenção da solução ótima dentro do espaço de busca definido, e o AG, que permitiu explorar o espaço de soluções com maior eficiência computacional. As funções objetivo foram aplicadas de forma independente, possibilitando a análise do sistema sob diferentes perspectivas de avaliação de lucro.

As simulações foram realizadas para diferentes combinações de parâmetros, com a função objetivo servindo como critério para seleção das melhores configurações. Os resultados são apresentados de forma comparativa, destacando os valores ótimos encontrados por cada método, bem como o desempenho das soluções em termos de taxa de atendimento, taxa de abandono e custo computacional associado.

A análise também inclui a comparação entre os métodos quanto ao tempo de execução e à qualidade das soluções obtidas, oferecendo uma visão clara sobre os benefícios e limitações de cada abordagem no contexto da otimização de sistemas estocásticos com controle.

4.2 Simulação de RPT com abandono

Neste estudo, utilizou-se uma RPT para modelar e analisar um sistema de fila do tipo cliente-servidor, no qual os clientes podem abandonar a fila dependendo do tempo de espera. O tempo de abandono é proporcional ao tamanho da fila, ou seja, filas maiores resultam em uma maior taxa de abandono.

O sistema possui capacidade máxima de 10 lugares e permite a entrada de um cliente por vez. Os tempos associados aos eventos seguem distribuições probabilísticas, conforme os valores esperados abaixo:

- $E[V_a] = 1{,}40$, $E[V_{qi}] = 14/i$, $E[V_c] = 0{,}10$;
- $E[V_d] = 1{,}171$, modelado por uma distribuição hipoexponencial ($\mu = 2{,}5$, $\sigma = 1{,}4$);

- $E[V_{sd}] = E[V_{s1}] = E[V_{s2}] = 0{,}443$.

A taxa de atendimento ($\mathrm{Tx}_{\mathrm{Atend}}$) e a taxa de abandono ($\mathrm{Tx}_{\mathrm{Aband}}$) foram definidas pelas expressões:

$$\mathrm{Tx}_{\mathrm{Atend}} = \frac{\text{Nº de clientes atendidos}}{\text{tempo total da simulação}} \tag{4.1}$$

$$\mathrm{Tx}_{\mathrm{Aband}} = \frac{\text{Nº de clientes que abandonaram}}{\text{tempo total da simulação}} \tag{4.2}$$

As simulações foram realizadas considerando a movimentação de 10.000 clientes, sendo os primeiros 400 descartados para o período de aquecimento. Cada experimento foi repetido 10 vezes para garantir robustez estatística.

A modelagem foi implementada no software ScicosLab 4.4.2, considerando as características operacionais previamente descritas. A simulação da rede resultou em uma taxa média de atendimento de aproximadamente 0,384, com intervalo de confiança de 95% em [0,380; 0,387], e uma taxa média de abandono de 0,283, com intervalo de confiança de 95% em [0,277; 0,287].

Para avaliar o desempenho do sistema, aplicaram-se duas variações da função objetivo definida na Equação 3.13, diferenciadas pelos valores de α e β utilizados na penalização da taxa de abandono. Além disso, foi considerada a função objetivo da Equação 3.14, baseada na razão entre a taxa de atendimento e a taxa de abandono. A Tabela 7 apresenta os lucros estimados obtidos em cada uma das formulações.

Tabela 7 – Resultados das simulações do sistema de fila em RPT estocástica sem controle

Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica		
Função Objetivo	**Lucro Estimado**	**Tempo de execução [s]**
L(M,f) = (Tx_atendimento - 10·Tx_abandono)·10	-24,489	5,91
L(M,f) = (Tx_atendimento - 30· Tx_abandono)·100	-810,621	5,99
L(M,f) = Tx_atendimento / Tx_abandono	1,359	5,94

Os resultados obtidos demonstram que, na ausência de controle, o sistema apresenta elevado prejuízo operacional, especialmente quando a taxa de abandono é fortemente penalizada nas FOs. Esses dados reforçam a necessidade de estratégias de controle para melhorar o desempenho global do sistema, o que será explorado nas seções seguintes.

4.3 Simulação de RPT com Controle Determinístico

Nesta etapa, foi incorporado ao sistema modelado por RPT um controle determinístico com o objetivo de regular o fluxo de clientes e reduzir a taxa de abandono sem comprometer a produtividade. Esse controle foi baseado na introdução de um lugar adicional na rede, responsável por limitar a admissão de clientes, por meio de dois parâmetros, sendo o número de fichas M e a temporização f associada à entrada no sistema. Os valores desses parâmetros foram definidos com base nas equações 3.11 e 3.12, resultando em $M = 2$ e $f = 1$, valores calculados a partir dos tempos médios das principais transições do sistema.

A aplicação do controle permitiu restringir o número de clientes na fila, atuando diretamente sobre os momentos de entrada e, consequentemente, sobre o comportamento de abandono. Para avaliar o desempenho do sistema sob essa nova configuração, foram aplicadas novamente as três formulações das FOs, conforme definido anteriormente.

A Tabela 8 apresenta os resultados obtidos, com destaque para o aumento no lucro estimado em duas das três formulações. Em particular, a função baseada na razão entre taxas apresentou melhora significativa, indicando que a atuação do controle contribuiu positivamente para o equilíbrio entre atendimento e abandono.

Tabela 8 – Resultados das simulações do sistema de fila em RPT estocástica com controle determinístico

Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica com Controle Determinístico		
Função Objetivo	**Lucro estimado**	**Tempo de execução [s]**
L(M,f) = (Tx_atendimento - 10·Tx_abandono)·10	1,276	6,01
L(M,f) = (Tx_atendimento - 30· Tx_abandono)·100	-26,917	5,96
L(M,f) = Tx_atendimento / Tx_abandono	16,133	5,82

Esses resultados evidenciam que a implementação de um controle simples, fundamentado em parâmetros médios do sistema, é capaz de proporcionar melhorias relevantes no desempenho da rede. Embora a função objetivo com penalização mais severa ainda apresente valor negativo, observa-se uma redução considerável do prejuízo quando comparado ao cenário sem controle, o que será discutido na próxima seção comparativa.

4.4 Otimização por Busca Exaustiva

Para explorar integralmente o comportamento do sistema sob diferentes condições de controle, aplicou-se a técnica de otimização BE, avaliando todas as combinações possíveis de parâmetros M e f dentro de um espaço de busca delimitado. Essa abordagem foi viável devido

à dimensão moderada do problema, permitindo uma análise completa das interações entre os parâmetros.

As simulações foram conduzidas variando-se o número de fichas M no intervalo de 2 a 10 e a temporização f de 0 a 4, com incrementos de 0,1. A escolha de iniciar a varredura a partir de $M = 2$ se justifica pelo fato de que, com somente uma ficha no controlador, não haveria formação de fila, e consequentemente, não ocorreria abandono de clientes, o que inviabilizaria a análise baseada nas funções objetivo propostas. Cada combinação foi avaliada conforme as três formulações distintas das FOs, descritas anteriormente, que relacionam as taxas de atendimento e abandono para estimar o lucro operacional do sistema.

A Tabela 9 apresenta os melhores resultados obtidos em cada FO, com os respectivos valores de lucro estimado, parâmetros de controle e tempos de execução.

Tabela 9 – Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica com Controle Otimizado via Busca Exaustiva

Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica com Controle Otimizado via Busca Exaustiva				
Função Objetivo	**Lucro estimado**	***M***	***f***	**Tempo de execução [s]**
L(M,f) = (Tx_atendimento - 10·Tx_abandono)·10	1,461	2	2,3	1704,7939
L(M,f) = (Tx_atendimento - 30· Tx_abandono)·100	3,300	2	4	1705,689
L(M,f) = Tx_atendimento / Tx_abandono	37,19	2	4	1704,312

Além da tabela de resultados, foram elaboradas visualizações gráficas para facilitar a análise dos efeitos combinados de M e f sobre o lucro. Para cada FO, foram gerados dois gráficos, sendo um mapa de calor e um gráfico 3D, permitindo observar com clareza as regiões de melhor desempenho do sistema. Essas representações auxiliam na compreensão do impacto dos parâmetros de controle e na identificação de padrões relevantes de comportamento do modelo.

4.4.1 Visualização dos Resultados da FO 1

A Figura 11 apresenta o gráfico 3D correspondente à primeira função objetivo $L(M, f) = (\text{Tx}_{\text{Atend}} - 10 \cdot \text{Tx}_{\text{Aband}}) \cdot 10$. Observa-se uma região de lucro máximo nas faixas inferiores de M, com ganhos mais expressivos conforme o aumento de f.

A Figura 12 apresenta o mapa de calor da mesma função. É possível observar com mais clareza as regiões de maior lucratividade, destacadas em tons mais quentes, indicando as melhores combinações de M e f.

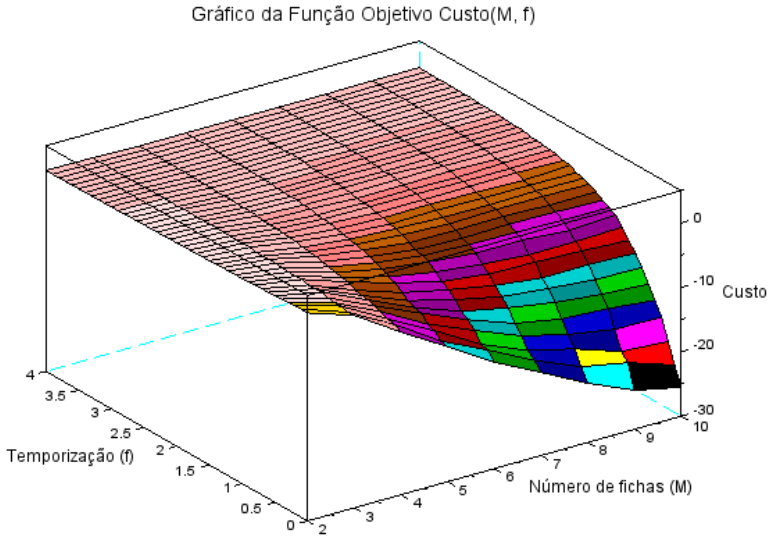

Figura 11 – Gráfico 3D da FO $L(M, f) = (\text{Tx}_{\text{Atend}} - 10 \cdot \text{Tx}_{\text{Aband}}) \cdot 10$

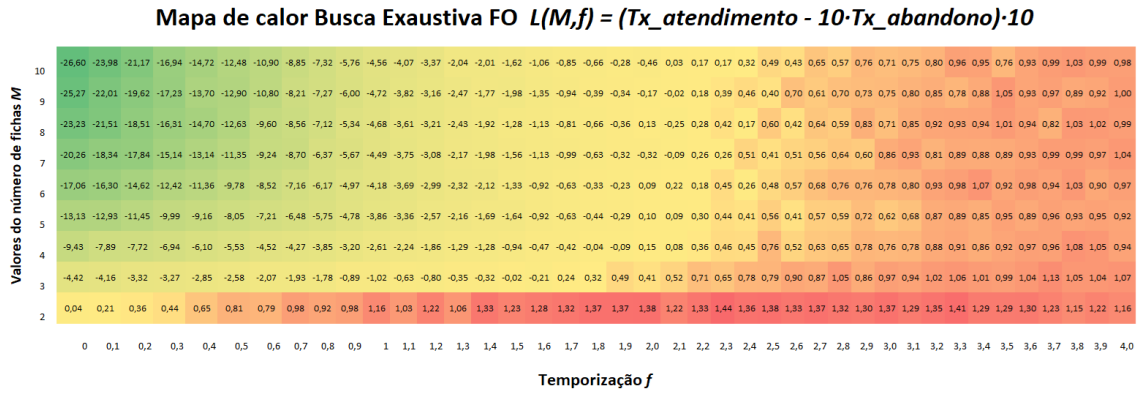

Figura 12 – Mapa de Calor da FO $L(M, f) = (\text{Tx}_{\text{Atend}} - 10 \cdot \text{Tx}_{\text{Aband}}) \cdot 10$

4.4.2 Visualização dos Resultados da FO 2

As visualizações para a segunda função objetivo, $L(M, f) = (\text{Tx}_{\text{Atend}} - 30 \cdot \text{Tx}_{\text{Aband}}) \cdot 100$, revelam o impacto de uma penalização mais severa aplicada à taxa de abandono. Ao atribuir um peso maior a esse componente, a função direciona a otimização para configurações que privilegiam a redução do número de desistências, mesmo que isso ocasione uma leve perda na produtividade. Como resultado, observa-se um deslocamento da região de maior lucro para valores mais elevados de f, o que implica em maior tempo entre admissões, e para valores mais baixos de M, reduzindo a quantidade de clientes que competem simultaneamente por atendimento. Essa configuração tende a diminuir a formação de filas longas, contribuindo diretamente para a redução do abandono e, consequentemente, para a maximização do lucro de acordo com os critérios definidos nesta formulação.

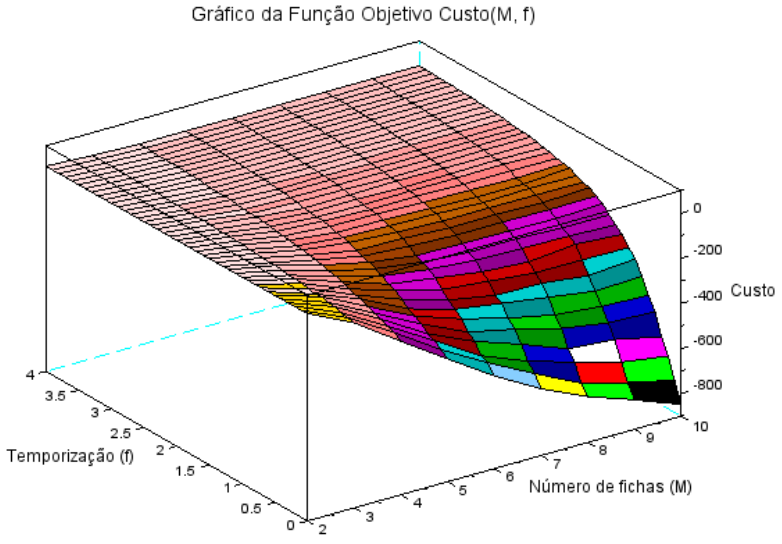

Figura 13 – Gráfico 3D da Função Objetivo $L(M, f) = (\text{Tx}_{\text{Atend}} - 30 \cdot \text{Tx}_{\text{Aband}}) \cdot 100$

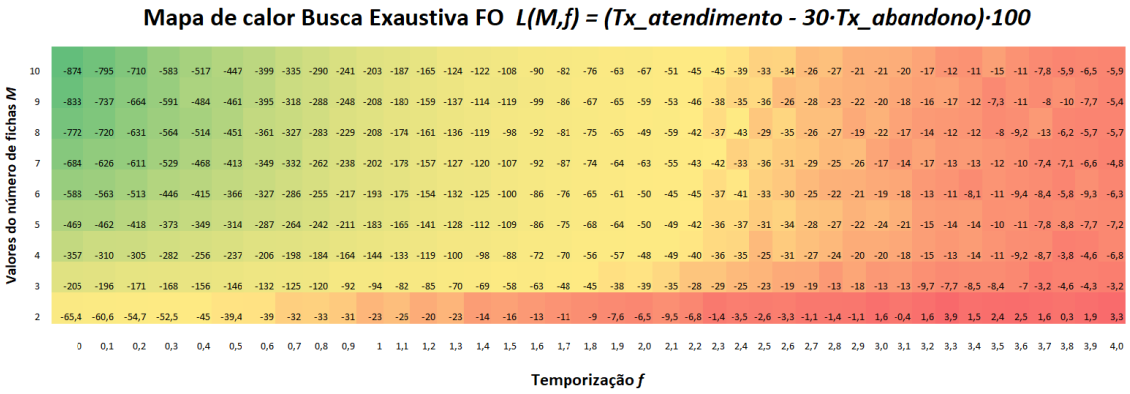

Figura 14 – Mapa de Calor da FO $L(M, f) = (\text{Tx}_{\text{Atend}} - 30 \cdot \text{Tx}_{\text{Aband}}) \cdot 100$

4.4.3 Visualização dos Resultados da FO 3

A terceira função $L(M, f) = \text{Tx}_{\text{Atend}}/\text{Tx}_{\text{Aband}}$ enfatiza a eficiência relativa do sistema. Os resultados indicam um comportamento similar à FO 2, favorecendo também os menores valores de M combinados a valores mais altos de f.

Os resultados obtidos por meio da BE permitiram uma análise abrangente do comportamento do sistema sob diferentes estratégias de controle. As representações gráficas, tanto em forma de superfícies tridimensionais quanto em mapas de calor, forneceram uma visualização clara das regiões mais vantajosas no espaço de busca. A abordagem exaustiva também serviu como base de comparação para os métodos heurísticos, oferecendo uma referência confiável para a avaliação da qualidade das soluções encontradas. A partir das três formulações das funções objetivo consideradas, foi possível identificar configurações que equilibram de forma eficaz

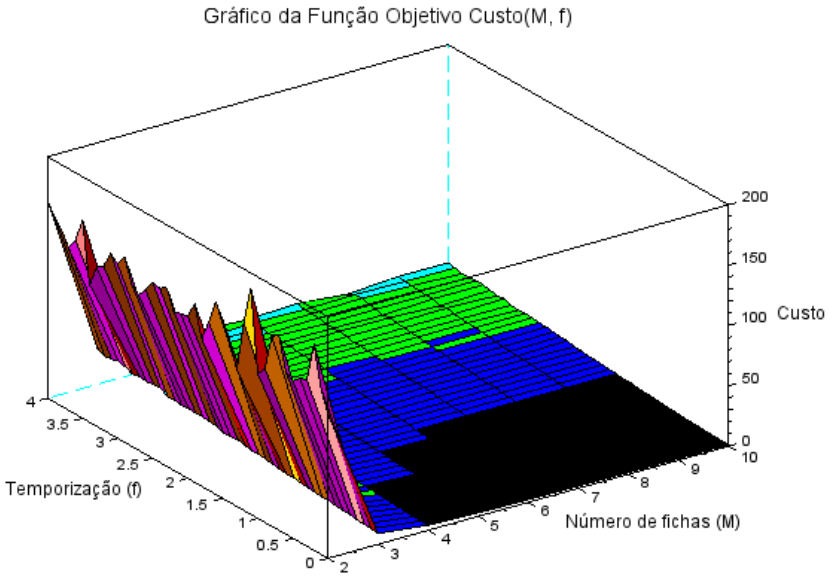

Figura 15 – Gráfico 3D da FO $L(M, f) = \mathrm{Tx}_{\mathrm{Atend}}/\mathrm{Tx}_{\mathrm{Aband}}$

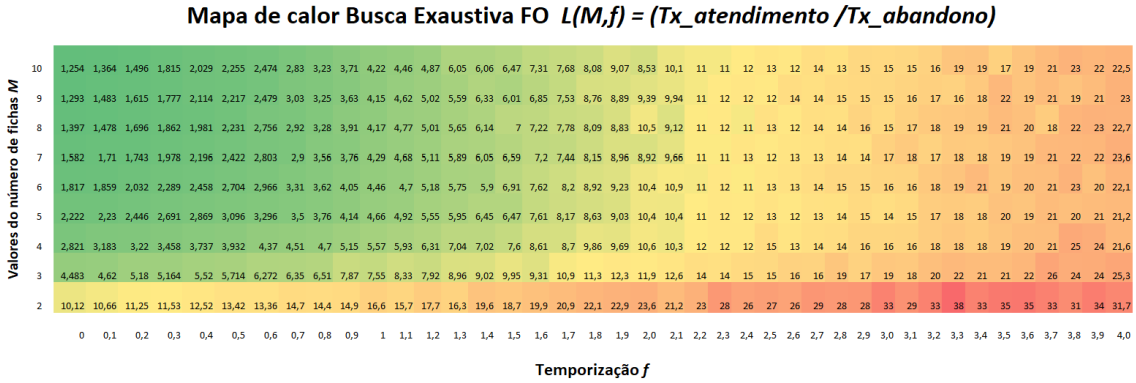

Figura 16 – Mapa de Calor da FO $L(M, f) = \mathrm{Tx}_{\mathrm{Atend}}/\mathrm{Tx}_{\mathrm{Aband}}$

a taxa de atendimento e a taxa de abandono, reforçando a utilidade da BE em cenários de complexidade controlada.

4.5 Otimização por Algoritmo Genético

Dando continuidade à análise do sistema modelado por RPT, aplicou-se o AG como alternativa de otimização para definir os valores mais adequados dos parâmetros de controle M e f. O objetivo foi identificar combinações que maximizassem o lucro do sistema com base nas FOs propostas, considerando o comportamento estocástico do modelo e suas restrições operacionais.

4.5.1 Configuração do Algoritmo Genético

O AG foi implementado com os seguintes parâmetros fixos:

- ❏ População inicial: 25 indivíduos;
- ❏ Taxa de mutação: 0,2;

Cada indivíduo da população é representado por um par (M, f), em que:

- ❏ M varia de 2 a 10 (valores inteiros);
- ❏ f varia de 0 a 4, com incrementos de 0,1.

A população inicial foi gerada aleatoriamente, respeitando os limites definidos para cada parâmetro. A cada geração, o algoritmo avaliava os indivíduos com base na simulação da RPT e nas respectivas FOs.

4.5.2 Critério de Parada

O critério de parada adotado foi baseado no desempenho da RPT com controle determinístico. A execução do AG foi interrompida no momento em que um indivíduo apresentou um lucro superior ao obtido com essa configuração. Assim, não houve necessidade de determinar previamente um número fixo de gerações, pois o algoritmo convergiu naturalmente para uma solução satisfatória dentro de 4 gerações.

4.5.3 Resultados Obtidos

A Tabela 10 apresenta os melhores resultados encontrados pelo AG para cada FO testada, incluindo os valores de M e f associados, bem como os tempos médios de execução da simulação.

Tabela 10 – Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica com Controle Otimizado via Algoritmo Genético

Resultados das Simulações do Sistema de Fila em Rede de Petri Temporizada Estocástica com Controle Otimizado via Algoritmo Genético				
Função Objetivo	**Lucro estimado**	***M***	***f***	**Tempo de execução [s]**
L(M,f) = (Tx_atendimento - 10·Tx_abandono)·10	1,437	2	2,3	672,015
L(M,f) = (Tx_atendimento - 30· Tx_abandono)·100	3,299	2	4	673,256
L(M,f) = Tx_atendimento / Tx_abandono	37,179	2	4	673,268

Como pode ser observado, os valores de $M = 2$ e f entre 2,3 e 4 foram identificados como os mais vantajosos para as funções objetivo consideradas, resultando em lucros superiores aos obtidos com o controle determinístico.

4.6 Análise da Complexidade Computacional

Nesta seção, é realizada uma análise da complexidade computacional das abordagens adotadas neste trabalho com base nos tempos de execução registrados durante as simulações. Considerando que a avaliação do desempenho do sistema é realizada a partir de simulações da RPT já controlada, o custo computacional de cada método está diretamente associado à quantidade de avaliações realizadas e ao tempo necessário para processá-las. Assim, os resultados obtidos permitem comparar a eficiência relativa das diferentes estratégias de otimização utilizadas.

4.6.1 Custo Computacional do Controle Determinístico

Do ponto de vista computacional, a inserção do controle determinístico na RPT não implicou em acréscimos significativos no tempo de simulação. Em comparação à versão sem controle, o tempo total de execução manteve-se praticamente inalterado, mesmo considerando a mesma quantidade de clientes simulados. Esse comportamento sugere que, mesmo com o aumento da complexidade estrutural, como a adição de servidores ou a ampliação da fila, o custo computacional permanece constante, caracterizando uma complexidade $O(1)$.

No modelo proposto, o controle foi implementado por meio da limitação de até dois clientes na fila e da imposição de um atraso fixo de uma unidade de tempo na entrada do sistema. Essa configuração, embora determinística e computacionalmente eficiente, resultou em uma redução na produtividade quando comparada ao sistema sem controle. No entanto, seu papel foi fundamental para estruturar a lógica de controle, servindo como referência para a formulação das FOs e para a aplicação das técnicas de otimização desenvolvidas neste trabalho.

4.6.2 Custo Computacional da Busca Exaustiva

A busca exaustiva consiste em testar todas as combinações possíveis dos parâmetros M e f dentro dos intervalos definidos, garantindo a identificação da melhor solução com base nas funções objetivo. No presente estudo, M foi variado de 2 a 10, enquanto f foi discretizado no intervalo de 0 a 4 com passo de 0,1. Isso resultou em um total de:

$$|M| \times |f| = 9 \times \frac{4}{0,1} = 9 \times 40 = 360 \tag{4.3}$$

Para cada par (M, f), foi executada uma simulação completa da RPT, totalizando um tempo de processamento de aproximadamente 28,5 minutos. A complexidade do processo pode

ser considerada proporcional ao número de combinações, sendo expressa por $O(nm)$, onde n representa o número de valores de M e m o número de valores discretizados de f.

Embora a busca exaustiva forneça uma solução globalmente ótima, seu custo computacional a torna inviável para problemas de maior dimensão, onde o espaço de busca cresce fatorialmente.

4.6.3 Custo Computacional do Algoritmo Genético

Os AGs oferecem uma alternativa eficiente para explorar grandes espaços de busca, reduzindo o custo computacional mesmo sem garantir a otimalidade global. No AG implementado neste estudo, a população inicial foi composta por 25 indivíduos, evoluindo ao longo de 4 gerações. Isso resultou em um total de 100 avaliações da função objetivo, cada uma associada a uma simulação completa do sistema.

A complexidade do algoritmo genético pode ser estimada por $O(G \cdot P)$, onde G representa o número de gerações e P o tamanho da população. Essa complexidade está diretamente relacionada ao número de simulações realizadas, uma vez que cada indivíduo precisa ser avaliado com base no desempenho do sistema modelado.

Com essa configuração, o AG foi capaz de encontrar a melhor solução em um tempo médio de 11,5 minutos, o que representa uma redução significativa em relação à busca exaustiva, cujo tempo total foi de aproximadamente 28,5 minutos. Essa diferença evidencia a eficiência da abordagem evolutiva na otimização do sistema, mesmo considerando o custo adicional de operações como cruzamento e mutação.

Apesar da redução do número de avaliações em relação à BE, o AG manteve um bom desempenho, identificando valores ótimos globais e demonstrando boa capacidade de convergência. A escolha dos parâmetros (tamanho da população, número de gerações e taxa de mutação) foi guiada pelo equilíbrio entre qualidade da solução e tempo de execução, e o critério de parada adotado foi baseado no lucro máximo obtido na simulação com controle determinístico.

4.6.4 Comparação das Abordagens

A análise comparativa entre as abordagens permite avaliar o custo computacional envolvido na aplicação de cada método de controle e otimização, considerando o tempo de execução necessário para obtenção dos resultados e sua viabilidade conforme o tamanho do problema.

Conforme observado, tanto o modelo base quanto o sistema com controle determinístico apresentaram tempos de execução semelhantes, próximos de 6 segundos, o que demonstra que a inclusão do controle direto não compromete a eficiência computacional. Em contraste, a busca exaustiva demandou aproximadamente 1705 segundos para avaliar todas as combinações possíveis de parâmetros, evidenciando sua limitação em problemas de maior escala. O algoritmo

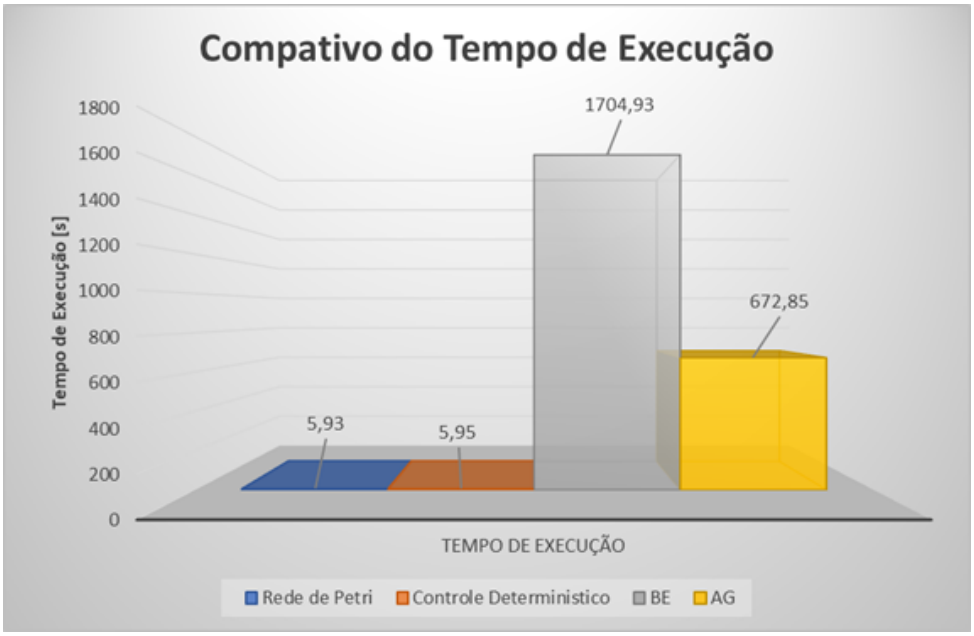

Figura 17 – Comparação dos tempos de execução entre as abordagens

genético, embora tenha realizado 100 avaliações, apresentou um tempo de execução significativamente menor, com média de 673 segundos, o que representa uma redução aproximada de 60% em relação à busca exaustiva.

As Tabelas 11 e 12 apresentam um resumo dos principais aspectos computacionais, reforçando a relação entre complexidade, escalabilidade e flexibilidade para cada abordagem.

Tabela 11 – Comparação das Abordagens de Otimização — Parte 1

Critério	Controle	BE	AG
Complexidade	$O(1)$	$O(nm^2)$	$O(GP)$
Escalabilidade	Alta	Baixa	Média
Garantia de Otimalidade	Não aplicável	Sim	Não

Com base nos resultados obtidos, observa-se que a BE é eficiente apenas quando o espaço de busca é reduzido. Para problemas mais complexos, o algoritmo genético surge como uma alternativa viável, permitindo encontrar boas soluções com menor custo computacional. Já o controle determinístico, embora rápido, apresenta limitações na flexibilidade e adaptabilidade do sistema.

Tabela 12 – Comparação das Abordagens de Otimização — Parte 2

Critério	Controle	BE	AG
Tempo de Execução	≈ 6 s	28,5 min	11,5 min
Complexidade	Nulo	Fatorial	Quadrático
Flexibilidade	Baixa	Alta	Alta
Aplicabilidade	Simples	Problemas de pequeno porte	Problemas de grande e médio porte

5 Conclusão

Este trabalho apresentou uma abordagem para otimização do controle em RPT, aplicada à modelagem de um sistema de filas do tipo cliente-servidor com possibilidade de abandono. A modelagem via RPT permitiu representar com precisão a dinâmica do sistema, incluindo os eventos estocásticos de chegada, atendimento e abandono de clientes. Essa representação viabilizou uma análise detalhada da produtividade e da taxa de abandono sob diferentes configurações de controle.

Inicialmente, foi implementado um controle determinístico, no qual os valores dos parâmetros M e f foram calculados com base nos tempos médios de cada etapa da rede. Essa configuração serviu como referência para a aplicação das técnicas de otimização. A partir da definição de FOs voltadas à maximização do lucro, aplicaram-se a BE e o AG para identificar as combinações de M e f que proporcionassem os melhores resultados de desempenho do sistema.

A BE percorreu todas as combinações possíveis dentro dos limites definidos, garantindo a solução globalmente ótima. No entanto, apresentou custo computacional elevado, com tempo total de simulação de aproximadamente 1704 segundos. Em contrapartida, o AG foi capaz de encontrar soluções muito próximas da ótima em tempo significativamente menor, com cerca de 673 segundos, o que representa uma redução superior a 60%. Mesmo sem garantia formal de otimalidade, o AG demonstrou eficiência, boa convergência e capacidade de adaptação.

A comparação entre as abordagens evidencia que, embora a BE seja útil em problemas com espaço de busca limitado, sua escalabilidade é restrita. Já o AG mostrou-se adequado para problemas maiores, com maior flexibilidade na configuração e ajustes do modelo. A utilização das FOs permitiu avaliar diretamente o lucro do sistema, equilibrando a taxa de atendimento e a penalização da taxa de abandono, o que resultou em uma solução mais aderente aos objetivos operacionais.

Os resultados obtidos evidenciam o potencial da integração entre modelagem estocástica via RPTs e técnicas de otimização metaheurísticas. O AG se destacou por seu desempenho computacional e flexibilidade, oferecendo soluções próximas ao ótimo com tempo de execução reduzido. Já a BE, apesar de garantir otimalidade, mostrou-se inviável para problemas de maior escala. O controle determinístico, embora simples, serviu como base de referência e apresentou desempenho próximo às melhores soluções. Assim, este trabalho oferece uma base sólida para a aplicação de estratégias evolutivas em sistemas a eventos discretos, com potencial para futuras extensões em modelagens mais complexas e com critérios multiobjetivo.

5.1 Trabalhos Futuros

Os resultados deste estudo demonstraram a viabilidade da aplicação de Algoritmos Genéticos para o controle de Redes de Petri Estocásticas, permitindo a obtenção de soluções otimizadas para reduzir a taxa de abandono sem comprometer significativamente a produtividade do sistema. No entanto, diversos aspectos podem ser aprimorados e explorados em trabalhos futuros, tanto do ponto de vista metodológico quanto computacional.

- ❏ Implementar técnicas adaptativas de mutação e cruzamento, ajustando dinamicamente as taxas desses operadores ao longo das gerações.
- ❏ Aplicar a abordagem desenvolvida a sistemas produtivos reais, como redes de manufatura e logística, analisando o impacto da otimização no desempenho operacional.
- ❏ Estender a modelagem para sistemas com múltiplos servidores ou filas paralelas, avaliando a escalabilidade das abordagens de otimização em estruturas mais complexas.

A continuidade deste estudo pode contribuir para o avanço das metodologias de controle em Redes de Petri Estocásticas, oferecendo soluções mais eficientes para sistemas de filas e outros modelos de eventos discretos.

Referências

ABOU-EL-ATA, M.; HARIRI, A. The m/m/c/n queue with balking and reneging. *Computers and Operations Research*, v. 19, p. 713–716, 1992. Citado na página 33.

AL-SEEDY, R. et al. Transient solution of the m/m/c queue with balking and reneging. *Computers and Mathematics with Applications*, v. 57, p. 1280–1285, 2009. Citado na página 33.

ARENALES, M. et al. *Pesquisa Operacional para Cursos de Engenharia*. [S.l.]: Elsevier e ABEPRO, 2011. Citado na página 36.

BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. *Nonlinear Programming: Theory and Algorithms*. 3rd. ed. [S.l.]: John Wiley & Sons, 2013. Citado 2 vezes nas páginas 38 e 39.

CASSANDRAS, C. G.; LAFORTUNE, S. *Introduction to Discrete Event Systems*. [S.l.]: Springer, 2008. Citado 5 vezes nas páginas 19, 21, 25, 28 e 30.

CURY, J. E. R.; QUEIROZ, M. H. Controle modular de sistemas de manufatura. *Revista Brasileira de Controle e Automação*, v. 13, n. 2, p. 123–133, 2002. Citado na página 19.

GASPAR-CUNHA, A.; TAKAHASHI, R.; ANTUNES, C. H. *Manual de Computação Evolutiva e Methaeurística*. [S.l.]: Imprensa da Universidade de Coimbra, 2012. Citado na página 41.

GLOVER, F.; LAGUNA, M. *Tabu Search*. [S.l.]: Springer, 1997. Citado 2 vezes nas páginas 38 e 39.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. [S.l.]: Addison-Wesley, 1989. ISBN 978-0201157673. Citado na página 39.

GROSS, D. et al. *Fundamentals of Queueing Theory*. [S.l.]: John Wiley & Sons, Inc., 2018. (Wiley Series in Probability and Statistics). Citado 6 vezes nas páginas 28, 31, 32, 33, 34 e 46.

KENDALL, D. G. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, Institute of Mathematical Statistics, v. 24, n. 3, p. 338–354, 1953. Citado na página 28.

KNUTH, D. E. *The Art of Computer Programming, Volume 4: Combinatorial Algorithms, Part 1*. [S.l.]: Pearson Education India, 2011. Citado na página 39.

MAIA, C.-A. Stochastic timed discrete-event systems: Modular modeling and performance evaluation through markovian jumps. *IEEE Access*, v. 10, p. 108332–108341, 2022. Citado 3 vezes nas páginas 29, 36 e 44.

NEMHAUSER, G. L.; WOLSEY, L. A. *Integer and Combinatorial Optimization*. [S.l.]: Wiley, 1988. ISBN 978-0471359432. Citado 3 vezes nas páginas 38, 39 e 41.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. 2nd. ed. [S.l.]: Springer, 2006. Citado na página 39.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity*. [S.l.]: Dover Publications, 1998. ISBN 978-0486402581. Citado na página 38.

RIBEIRO, R. G. *Performance evaluation of stochastic DES through analytical models and simulation: an open-pit mine study*. 121 p. Tese (Tese de Doutorado) — Universidade Federal de Minas Gerais, Escola de Engenharia, Belo Horizonte, Brasil, 2018. Bibliografia: f. 106-112. Anexos: f. 113-121. Citado na página 28.

ROUBOS, A.; JOUINI, O. Call centers with hyperexponential patience modeling. *International Journal of Production Economics*, v. 141, n. 1, p. 307–315, 2013. ISSN 0925-5273. Meta-heuristics for manufacturing scheduling and logistics problems. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925527312003428>. Citado 2 vezes nas páginas 33 e 35.

SOUZA, L.; MAIA, C. A. Análise de sistemas a eventos discretos via modelos modulares: Aplicação a sistemas de serviço com abandono de cliente. *Anais do 14° Simpósio Brasileiro de Automação Inteligente (SBAI)*, p. 503–508, 2019. Citado na página 31.

VANDERBEI, R. J. *Linear Programming: Foundations and Extensions*. 5th. ed. [S.l.]: Springer, 2020. Citado na página 37.

WANG, J. *Timed Petri Nets*. [S.l.]: Kluwer Academic Publishers, 1998. Citado 2 vezes nas páginas 21 e 25.

WANG, K.; LI, N.; JIANG, Z. Queueing system with impatient customers: A review. In: *Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2010*. [S.l.: s.n.], 2010. p. 82–87. Citado 3 vezes nas páginas 32, 33 e 34.

WHITT, W. Engineering solution of a basic call-center model. *Department of Industrial Engineering and Operations Research - Columbia University*, 2004. Citado na página 33.

YAMALIDOU, E. et al. Feedback control of petri nets based on place invariants. *Automatica*, v. 32, n. 1, p. 15–28, 1996. Citado na página 25.

YUE, D.; W., Y.; ZHAO, G. Analysis of an m/m/c queueing system with impatient customers and synchronous vacations. *Journal of Applied Mathematics*, v. 2014, p. 1–12, 2014. Citado na página 33.

ZHANG, Q.; LI, H. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 13, n. 2, p. 284–302, 2009. Citado na página 39.

APÊNDICE A – Código para Simulação de Redes de Petri com Abandono

O código a seguir apresenta a implementação de um simulador elementar de Redes de Petri Temporizadas (RPT) com atraso no disparo de transições, obedecendo à distribuição exponencial. Esse simulador foi desenvolvido para modelar um sistema de fila cliente-servidor com abandono e otimizar o controle desse sistema.

Desenvolvimento:

- ❏ Prof. Carlos Andrey Maia – Departamento de Engenharia Elétrica, UFMG
- ❏ Adaptação: Daniel Fernandes Gomes, sob orientação do Prof. Dr. Carlos Andrey Maia

A.1 Descrição do Código

O código foi implementado no **ScicosLab** e modela um sistema cliente-servidor utilizando uma Rede de Petri Temporizada. A simulação permite avaliar a taxa de atendimento e abandono, além de calcular métricas relevantes do sistema.

Entradas:

- ❏ $\mathbf{A}$: Matriz de incidência da rede.
- ❏ $\mathbf{A}^{-}$: Matriz de incidência negativa da rede.
- ❏ $\mathbf{X_0}$: Vetor de marcação inicial.
- ❏ λ: Vetor com taxas de disparo das transições.

Saídas:

- ❏ Índice k: Numeração dos disparos das transições.
- ❏ Vetor U_k: Indica qual transição foi disparada.
- ❏ Vetor X_k: Indica a marcação da rede após o disparo.

O código Scilab está listado abaixo:

```
clear;
clc;

// ---------------- Modelo da Rede de Petri ----------------

A = [
    1  -1   0   0   0   0   0   0   0;
    0   1  -1   0   0   0   0   0  -1;
    0   0   1  -1   0   0   0   0   0;
    0   0  -1   0   0   0   0   1   0;
    0   0   0   1  -1   0   0   0   0;
    0   0   0   0   1  -1   0   0   0;
    0   0   0   0   0   1  -1   0   0;
   -1   1   0   0   0   0   0   0   0;
    0   0   0   0   0   0   1  -1   0;
   -1   0   0   0   1   0   0   0   1
];

Aminus = [
    0   1   0   0   0   0   0   0   0;
    0   0   1   0   0   0   0   0   1;
    0   0   0   1   0   0   0   0   0;
    0   0   1   0   0   0   0   0   0;
    0   0   0   0   1   0   0   0   0;
    0   0   0   0   0   1   0   0   0;
    0   0   0   0   0   0   1   0   0;
    1   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   1   0;
    1   0   0   0   0   0   0   0   0
];

A = A';
Aminus = -Aminus';

// ---------------- Parametros da Simulacao ----------------

Inv_Lu = 0.1;
Inv_La = 1.4;
Inv_Lc1 = 10^(-100);
Inv_Lc = 0.1;
Inv_Lsd = 0.443;
Inv_Ls1 = 0.443;
Inv_Ls2 = 0.443;
Inv_Ld = 1.171;
Inv_Lq = 14;

// Estado inicial da Rede
```

```
X0 = [0 0 0 1 0 0 0 1 0 10];

NR = 10;
contA = 0;
contPA = 0;
contPd = 0;
Taxa = 0;

// ---------------- Simulacao da RPT ----------------

Nmax = 10000;
[n, m] = size(A);
Xk = X0;
Uk = zeros(1, n);
XX = zeros(n, m);
Delta = 10^(-100);
Top = 10^(100);
tempo = 0;
Tx_Ab = zeros(1, NR);
Tx_At = zeros(1, NR);

tic();
for nr = 1:NR
    for k = 1:Nmax
        for i = 1:n
            XX(i, :) = Xk;
        end

        Xk_2 = max((10^(-100)), Xk(2));
        Inv_Lambda = [Inv_Lu; Inv_La; Inv_Lc1; Inv_Lc; Inv_Lsd; Inv_Ls1;
            Inv_Ls2; Inv_Ld; 14/Xk_2];

        R = XX + Aminus;
        D = abs(R) - R;
        Thab = sum(D, 2);

        if prod(Thab) == 0 then
            Tempos_aleat = (-Inv_Lambda) .* log(rand(Thab) + Delta) + Top
                * Thab;
            [Menor_tempo, Trans_menor_tempo] = min(Tempos_aleat);
            Uk = zeros(Uk);
            Uk(Trans_menor_tempo) = 1;
            Xk = Xk + Uk * A;

            if k > 400 then
                tempo = tempo + Menor_tempo;
                if Uk(1) == 1 then contPA = contPA + 1; end
```

```
                if Uk(7) == 1 then contPd = contPd + 1; end
                if Uk(9) == 1 then contA = contA + 1; end
            end
        else
            printf("A rede esta morta. Nao existem transicoes
                habilitadas!");
            abort;
        end
    end

    Tx_atend = contPd / tempo;
    TaxaAban2 = Inv_La^(-1) - Tx_atend;
    Tx_Ab(nr) = TaxaAban2;
    Tx_At(nr) = Tx_atend;
end

toc();
Texec = toc();

// ---------------- Resultados ----------------
disp("Taxa de abandono: " + mean(Tx_Ab));
disp("Taxa de atendimento: " + mean(Tx_At));
disp("Tempo de execucao: " + string(Texec) + " segundos");
```

Listing A.1 – Código da Simulação de RPT com Abandono

APÊNDICE B – Código para Simulação de Rede de Petri com Abandono Controlado

```
// Simulador de Redes de Petri com Atraso no Disparo de Transicoes
// Obedecendo a Distribuicao Exponencial
// Prof. Carlos Andrey Maia
// Departamento de Eng. Eletrica da UFMG
// Data: 18/11/2023
// Modelagem do sistema de abandono para Redes de Petri
// Adaptado por Daniel Fernandes Gomes Sob orientacao do Prof. Dr. Carlos
   Andrey Maia

clear;
clc;

// ---------------------- Definicao da Rede de Petri ----------------------
A = [
    1  -1   0   0   0   0   0   0   0;
    0   1  -1   0   0   0   0   0  -1;
    0   0   1  -1   0   0   0   0   0;
    0   0  -1   0   0   0   0   1   0;
    0   0   0   1  -1   0   0   0   0;
    0   0   0   0   1  -1   0   0   0;
    0   0   0   0   0   1  -1   0   0;
   -1   1   0   0   0   0   0   0   0;
    0   0   0   0   0   0   1  -1   0;
   -1   0   0   0   1   0   0   0   1;
   -1   0   0   0   0   0   0   1   1
];

Aminus = [
    0   1   0   0   0   0   0   0   0;
    0   0   1   0   0   0   0   0   1;
    0   0   0   1   0   0   0   0   0;
    0   0   1   0   0   0   0   0   0;
    0   0   0   0   1   0   0   0   0;
    0   0   0   0   0   1   0   0   0;
    0   0   0   0   0   0   1   0   0;
    1   0   0   0   0   0   0   0   0;
    0   0   0   0   0   0   0   1   0;
    1   0   0   0   0   0   0   0   0;
```

```
    1  0  0  0  0  0  0  0  0
];

A = A';
Aminus = -Aminus';

// ---------------------- Definicao das Taxas de Disparo
    ----------------------
Inv_Lu = 10^(-100);
Inv_La = 1.4;
Inv_Lc1 = 10^(-100);
Inv_Lc = 0.1;
Inv_Lsd = 0.443;
Inv_Ls1 = 0.443;
Inv_Ls2 = 0.443;
Inv_Ld = 1.171;
Inv_Lq = 14;

// Estado inicial da Rede
X0 = [0 0 0 1 0 0 0 1 0 10 1];

NR = 10;
contA = 0;
contPA = 0;
contPd = 0;
tempo = 0;
M = 0;

// ---------------------- Configuracao do Controlador
    ----------------------
Emax = max([Inv_La, (Inv_La + Inv_Lc + Inv_Lc1) / 2, (Inv_Lsd + Inv_Ls1 +
    Inv_Ls2 + Inv_Ld)]);
M = ceil((Inv_La + Inv_Lc1 + Inv_Lc + Inv_Lsd + Inv_Ls1 + Inv_Ls2 +
    Inv_Ld) / Emax);
X0(11) = M;
f = Emax * M - Inv_La - (Inv_Lc1 + Inv_Lc) - (Inv_Lsd + Inv_Ls1 + Inv_Ls2
    + Inv_Ld);
Inv_Lu = f;

// ---------------------- Simulacao da Rede ----------------------
Nmax = 10000;
[n, m] = size(A);
Xk = X0;
Uk = zeros(1, n);
XX = zeros(n, m);
Delta = 10^(-100);
Top = 10^(100);
```

```
Tx_Ab = zeros(1, NR);
Tx_At = zeros(1, NR);

tic();
for nr = 1:NR
    for k = 1:Nmax
        for i = 1:n
            XX(i, :) = Xk;
        end

        Xk_2 = max(10^(-100), Xk(2));
        Inv_Lambda = [Inv_Lu; Inv_La; Inv_Lc1; Inv_Lc; Inv_Lsd; Inv_Ls1;
           Inv_Ls2; Inv_Ld; 14/Xk_2];

        // Verificacao de transicoes habilitadas
        R = XX + Aminus;
        D = abs(R) - R;
        Thab = sum(D, 2);

        if prod(Thab) == 0 then
            Tempos_aleat = (-Inv_Lambda) .* log(rand(size(Thab, 1), 1) +
               Delta) + Top * Thab;
            [Menor_tempo, Trans_menor_tempo] = min(Tempos_aleat);
            Uk = zeros(Uk);
            Uk(Trans_menor_tempo) = 1;
            Xk = Xk + Uk * A;

            if k > 400 then
                tempo = tempo + Menor_tempo;
                if Uk(2) == 1 then contPA = contPA + 1; end
                if Uk(7) == 1 then contPd = contPd + 1; end
                if Uk(9) == 1 then contA = contA + 1; end
            end
        else
            printf("A␣rede␣esta␣morta.␣Nao␣existem␣transicoes␣
               habilitadas!");
            abort;
        end
    end
    Tx_Ab(nr) = Inv_La^(-1) - (contPd / tempo);
    Tx_At(nr) = contPd / tempo;
end
toc();
Texec = toc();
```

```
// ---------------------- Calculo do Intervalo de Confianca
    ----------------------
Z = 1.96;
N = NR;
Taxa_aband = mean(Tx_Ab);
desvio_padrao = stdev(Tx_Ab);
erro_margem = Z * (desvio_padrao / sqrt(N));
IC_inferior = Taxa_aband - erro_margem;
IC_superior = Taxa_aband + erro_margem;

// ---------------------- Impressao dos Resultados ----------------------
disp('Entraram:'), disp(contPA);
disp('Atendidos:'), disp(contPd);
disp('Abandonaram:'), disp(contA);
disp("Taxa de abandono: " + string(Taxa_aband));
disp("Intervalo de confianca 95%: [" + string(IC_inferior) + ", " +
    string(IC_superior) + "]");
disp("Taxa de atendimento: " + string(contPd / tempo));
disp("Tempo de execucao: " + string(Texec) + " segundos");
```

Listing B.1 – Código da Simulação de RPT com Controle

APÊNDICE C – Código da Busca Exaustiva para Otimização da Rede de Petri

```
// Simulador Elementar de Redes de Petri com Atraso no Disparo de
   Transicoes
// Obedecendo a Distribuicao Exponencial
// Prof. Carlos Andrey Maia
// Departamento de Eng. Eletrica da UFMG
// Data: 18/11/2023
// Modelagem do sistema de abandono para Redes de Petri
// Adaptado por Daniel Fernandes Gomes Sob orientacao do Prof. Dr. Carlos
   Andrey Maia

clear;
clc;

// ---------------------- Definicao da Rede de Petri ----------------------
A = [
    1  -1   0   0   0   0   0   0   0;
    0   1  -1   0   0   0   0   0  -1;
    0   0   1  -1   0   0   0   0   0;
    0   0  -1   0   0   0   0   1   0;
    0   0   0   1  -1   0   0   0   0;
    0   0   0   0   1  -1   0   0   0;
    0   0   0   0   0   1  -1   0   0;
   -1   1   0   0   0   0   0   0   0;
    0   0   0   0   0   0   1  -1   0;
   -1   0   0   0   1   0   0   0   1;
   -1   0   0   0   0   0   0   1   1
];

Aminus = [
    0   1   0   0   0   0   0   0   0;
    0   0   1   0   0   0   0   0   1;
    0   0   0   1   0   0   0   0   0;
    0   0   1   0   0   0   0   0   0;
    0   0   0   0   1   0   0   0   0;
    0   0   0   0   0   1   0   0   0;
    0   0   0   0   0   0   1   0   0;
    1   0   0   0   0   0   0   0   0;
```

```
    0  0  0  0  0  0  0  1  0;
    1  0  0  0  0  0  0  0  0;
    1  0  0  0  0  0  0  0  0
];

A = A';
Aminus = -Aminus';

// ---------------------- Definicao das Taxas de Disparo
    ----------------------
Inv_Lu = 10^(-100);
Inv_La = 1.4;
Inv_Lc1 = 0.1;
Inv_Lc = 10^(-100);
Inv_Lsd = 0.443;
Inv_Ls1 = 0.443;
Inv_Ls2 = 0.443;
Inv_Ld = 1.171;
Inv_Lq = 14;

// Estado inicial da Rede
X0 = [0 0 0 1 0 0 0 1 0 10 1];

// Definicao dos intervalos para M e F
num_M = 10;
num_F = (4 - 0.1) / 0.1 + 1;
f_values = 0:0.1:4;
m_values = 1:10;

// Inicializar matriz de custos
custo_matrix = zeros(num_M, num_F);

tic("contador1");
for ContM = 1:num_M
    for contf = f_values
        NR = 10;
        contA = 0;
        contPA = 0;
        contPd = 0;
        tempo = 0;

        Nmax = 10000;
        Xk = X0;
        Uk = zeros(1, size(A, 1));

        M = ContM;
        Xk(11) = M;
```

```
        f = contf;
        Inv_Lu = f;

        for nr = 1:NR
            for k = 1:Nmax
                Xk_2 = max(10^(-100), Xk(2));
                Inv_Lambda = [Inv_Lu; Inv_La; Inv_Lc1; Inv_Lc; Inv_Lsd;
                    Inv_Ls1; Inv_Ls2; Inv_Ld; 14/Xk_2];

                R = Xk + Aminus;
                D = abs(R) - R;
                Thab = sum(D, 2);

                if prod(Thab) == 0 then
                    Tempos_aleat = (-Inv_Lambda) .* log(rand(size(Thab,
                        1), 1) + 10^(-100)) + 10^(100) * Thab;
                    [Menor_tempo, Trans_menor_tempo] = min(Tempos_aleat);
                    Uk = zeros(Uk);
                    Uk(Trans_menor_tempo) = 1;
                    Xk = Xk + Uk * A;

                    if k > 400 then
                        tempo = tempo + Menor_tempo;
                        if Uk(2) == 1 then contPA = contPA + 1; end
                        if Uk(7) == 1 then contPd = contPd + 1; end
                        if Uk(9) == 1 then contA = contA + 1; end
                    end
                else
                    printf("A rede esta morta. Nao existem transicoes
                        habilitadas!");
                    abort;
                end
            end
        end

        TaxaAten = contPd / tempo;
        TaxaAban = Inv_La^(-1) - TaxaAten;
        Custo = TaxaAten - 10 * TaxaAban;

        f_index = round((contf - 0) / 0.1) + 1;
        custo_matrix(ContM, f_index) = Custo;
    end
end

// ---------------------- Resultados da Otimizacao ----------------------
[menor_valor, posicao] = max(custo_matrix);
[l, c] = ind2sub(size(custo_matrix), find(custo_matrix == menor_valor));
```

```
disp("Melhor custo encontrado: " + string(menor_valor) + " para M = " +
    string(l) + " e f = " + string(c / 10));
toc("contador1");
Tempo1 = toc("contador1");
disp("Tempo total de execucao: " + string(Tempo1) + " segundos");

// ---------------------- Grafico da Funcao Objetivo ----------------------
clf();
[xx, yy] = ndgrid(m_values, f_values);
surf(xx, yy, custo_matrix);
xlabel('Numero de fichas (M)');
ylabel('Temporizacao (f)');
zlabel('Custo');
title('Grafico da Funcao Objetivo Custo(M, f)');
```

Listing C.1 – Código da Simulação de RPT com Busca Exaustiva

APÊNDICE D – Código AG para Otimização da Rede de Petri

```
function custo = simular_rede(m, f)

    // Matrizes de conexao
  A =  [1 -1  0  0  0  0 0  0  0;  //PA
          0   1 -1  0  0  0 0  0 -1; //P1
          0   0  1 -1  0  0 0  0  0;  //P3
          0   0 -1  0  0  0 0  1  0;  //P4
          0   0  0  1 -1  0 0  0  0;  //p5
          0   0  0  0  1 -1 0  0  0;  //P6
          0   0  0  0  0  1 -1 0  0;  //P7
         -1   1  0  0  0  0 0  0  0;  //Pca
          0   0  0  0  0  0 1 -1  0;  //Pd
         -1   0  0  0  1  0 0  0  1;  //p2
         -1   0  0  0  0  0 0  1  1]; //F


Aminus =[0 1 0 0 0 0 0 0 0;  //PA
          0 0 1 0 0 0 0 0 1; //P1
          0 0 0 1 0 0 0 0 0;  //P3
          0 0 1 0 0 0 0 0 0;  //P4
          0 0 0 0 1 0 0 0 0;  //p5
          0 0 0 0 0 1 0 0 0;  //P6
          0 0 0 0 0 0 1 0 0;  //P7
          1 0 0 0 0 0 0 0 0;  //Pca
          0 0 0 0 0 0 0 1 0;  //Pd
          1 0 0 0 0 0 0 0 0;  //p2
          1 0 0 0 0 0 0 0 0]; //F
A = A';                          //para trabalhos com vetor coluna
Aminus = -Aminus';               //para trabalhos com vetor coluna

Inv_Lu = f;      //Inverso de Lambda da transicao U
Inv_La = 1.4;            //Inverso de Lambda da transicao A
Inv_Lc1 = 10^(-100);     //Inverso de Lambda da transicao C'
Inv_Lc = 0.1;             //Inverso de Lambda da transicao C
Inv_Lsd = 0.443;         //Inverso de Lambda da transicao Sd
Inv_Ls1 = 0.443;         //Inverso de Lambda da transicao S1
Inv_Ls2 = 0.443;         //Inverso de Lambda da transicao S2
Inv_Ld = 1.171;          //Inverso de Lambda da transicao D
Inv_Lq = 14;              //Inverso de Lambda da transicao Q
```

```
X0 = [0 0 0 1 0 0 0 1 0 10 m];  // 10

NR=10
contA = 0;
contPA = 0;             //Contador 1
contPd = 0;
Taxa=0;

// ---------------------------------------------------------------------


Nmax=10000; // Numero de disparos de transicoes

[n m]=size(A);

Xk=X0;

Uk=zeros(1,n);

XX=zeros(n,m);

Delta=10^(-100);

Top=10^(100);

tempo=0;
Ab=zeros(1,NR)
Tx_At=zeros(1, NR)
Ncli=zeros(1, NR)
Mtemp = zeros(1, NR)

tic("contador1")
for nr = 1:NR

for  k=1:Nmax

       for  i=1:n

    XX(i,:)=Xk;   // Matriz cujas linhas sao os estados Xk

 end

    Xk_2 = max((10^(-100)),Xk(2));
    Inv_Lambda=[Inv_Lu;Inv_La;Inv_Lc1;Inv_Lc;
    Inv_Lsd;Inv_Ls1;Inv_Ls2;Inv_Ld;14/Xk_2];
```

```
    //   Verificacao transices habilitadas

    R= XX+ Aminus; // Matrix que indica retirada de fichas dos lugares de
        entrada das transicoes

    D=abs(R)-R;  // Elemento D(i,j) positivo indica que a transicao "i"
        esta desabilitada.

    Thab= sum(D,2); // Vetor contendo soma valores contidos nas linhas.
        Valor positivo em Thab(i), indica que a  transicao "i" esta
        desabilitada

    if prod(Thab) == 0 then   // Caso uma transicao, digamos "i",  esteja
        habilitada entao Thab(i)=0, e, portanto o produto dos elementos de
        Thab deve ser nulo.

         Tempos_aleat= (-Inv_Lambda).*log(rand(Thab)+ Delta)+ Top*Thab;
            // Artificio: Gera-se um tempo muito grande para as transicoes
            desabilitadas, forcando que elas jamais disparem.

         [Menor_tempo,Trans_menor_tempo]=min(Tempos_aleat);

          k = k ; // k-esimo disparo da transicao indicada em
              "Trans_menor_tempo"

          Uk=zeros(Uk);      // Zera-se o vetor Uk

          Uk(Trans_menor_tempo)=1;   // Escolhe-se a transicao a ser
              disparada como sendo aquela com o menor tempo associado

          Xk=Xk+ Uk*A;  // Atualiza-se o estado


//************************************************************
//Contadores de Entrada, saida e abandono de pecas
//************************************************************

 if  k>400 // Transitorio

         tempo= tempo+Menor_tempo;
   // tempo = 24043
        if Uk(1)==1
             contPA = contPA+1;
         end
         if Uk(7)==1
```

```
            contPd = contPd+1;
        end
      if Uk(9)==1
            contA = contA+1;
      end

   end  // k>1000

        else printf("␣A␣rede␣esta␣morta.␣Nao␣existem␣transicoes␣
           habilitadas␣!"); abort ;

 end
 end

Tx_atend = contPd/tempo;
TaxaAban2= Inv_La^(-1) - Tx_atend;
Tx_Ab(nr)= TaxaAban2;
Tx_At(nr)= Tx_atend;
end
toc("contador1")
Texec=toc("contador1")
temp1=Texec+toc("contador1")
N = NR;                      // Tamanho da amostra
Taxa_aband=mean(Tx_Ab);
desvio_padrao = stdev(Tx_Ab);          // Desvio padrao
Taxa_atend=mean(Tx_At);
Z = 1.96;                               // Valor critico para 95%
erro_margem = Z * (desvio_padrao / sqrt(N)); // Margem de erro
IC_inferior = Taxa_aband - erro_margem;           // Limite inferior
IC_superior = Taxa_aband + erro_margem;           // Limite superior

//disp("Taxa de abandono: " + string(Taxa_aband));
//disp("Intervalo de confianca 95%: [" + string(IC_inferior) + ", " +
   string(IC_superior) + "]");
//disp("Taxa de atendimento: " +string(Taxa_atend));

custo = Taxa_atend - 10*Taxa_aband
//disp("Custo: " + string(custo));
endfunction

function resultado_otimo = algoritmo_genetico(tam_pop, num_geracoes,
   taxa_mutacao, min_m, max_m, min_f, max_f)
tic("contador2")
    disp("Iniciando␣algoritmo␣genetico...");

    // Inicializar populacao inicial
    populacao = [];
```

```
for i = 1:tam_pop
    individuo = [round(min_m + rand() * (max_m - min_m)), ...
                 round((min_f + rand() * (max_f - min_f)) * 10) / 10];
                    // Arredonda para uma casa decimal
    populacao = [populacao; individuo];
end

melhor_custo = -%inf;
melhor_individuo = [];

for geracao = 1:num_geracoes
    disp("Geracao: " + string(geracao));

    // Avaliar a aptidao de cada individuo
    custos = [];
    for i = 1:size(populacao, 1)
        individuo = populacao(i, :);
        m = individuo(1);
        f = individuo(2);
        custo = simular_rede(m, f); // Substitua por sua funcao de
            avaliacao
        custos = [custos; custo];
        if custo > melhor_custo then
            melhor_custo = custo;
            melhor_individuo = individuo;
        end
    end

    // Selecao por torneio
    nova_populacao = [];
    for i = 1:tam_pop
        idx1 = ceil(rand() * tam_pop);
        idx2 = ceil(rand() * tam_pop);
        if custos(idx1) > custos(idx2) then
            nova_populacao = [nova_populacao; populacao(idx1, :)];
        else
            nova_populacao = [nova_populacao; populacao(idx2, :)];
        end
    end

    // Cruzamento
    for i = 1:2:size(nova_populacao, 1) - 1
        if rand() < 0.8  // Taxa de cruzamento fixa em 0.8
            pai1 = nova_populacao(i, :);
            pai2 = nova_populacao(i + 1, :);
            ponto_corte = ceil(rand() * 2);
            filho1 = [pai1(1:ponto_corte), pai2(ponto_corte + 1:$)];
```

```
                filho2 = [pai2(1:ponto_corte), pai1(ponto_corte + 1:$)];
                filho1(2) = round(filho1(2) * 10) / 10; // Garante uma
                    casa decimal
                filho2(2) = round(filho2(2) * 10) / 10; // Garante uma
                    casa decimal
                nova_populacao(i, :) = filho1;
                nova_populacao(i + 1, :) = filho2;
            end
        end

        // Mutacao
        for i = 1:size(nova_populacao, 1)
            if rand() < taxa_mutacao
                nova_populacao(i, :) = [round(min_m + rand() * (max_m -
                    min_m)), ...
                                        round((min_f + rand() * (max_f -
                                            min_f)) * 10) / 10]; // Garante
                                            uma casa decimal
            end
        end

        populacao = nova_populacao;
    end

    disp("Melhor solucao encontrada:");
    disp("Tokens (m): " + string(melhor_individuo(1)));
    disp("Temporizacao (f): " + string(melhor_individuo(2)));
    disp("Custo maximo: " + string(melhor_custo));

    resultado_otimo = melhor_individuo;
    toc();
    tempexec = toc();
    disp("Tempo de execucao: " + string(tempexec) + " segundos");
toc("contador2")
Texec2= toc("contador2")
endfunction

// Parametros do AG
tam_pop = 24;
num_geracoes =8;
taxa_mutacao = 0.2;
min_m = 1; max_m = 10;
min_f = 0.1; max_f = 3.0;

// Executa o AG
algoritmo_genetico(tam_pop, num_geracoes, taxa_mutacao, min_m, max_m,
    min_f, max_f);
```

Listing D.1 – Código da Simulação de RPT com Algoritmo Genético