Segundo Trabalho Prático: Implementação do Algoritmo de Análise do Valor Médio

Alunos: Gleydiston Bragança, Fernando Lessa
Professor(a): Marta Dias Moreira Noronha

Junho 2025

0.1 Introdução

A gestão eficiente e a otimização do uso dos recursos é crucial nas mais diversas áreas de atuação; portanto, tornam-se necessárias maneiras de garantir uma qualidade de serviço sem desperdícios. Uma das áreas que estuda a otimização de serviços que necessitam de espera é a Teoria das Filas, onde o objetivo central é avaliar o desempenho de sistemas que envolvem a espera de recursos, sendo indispensável para sistemas computacionais, logísticos e de serviços. Neste trabalho iremos abordar o método de **Análise do Valor Médio (MVA - Mean Value Analysis)**, uma técnica amplamente usada e fundamental na teoria das filas, onde diferente das técnicas de simulação, se utiliza de uma maneira mais direta para calcular suas métricas.

0.2 Objetivo do Trabalho

O objetivo principal deste trabalho é implementar e aplicar o algoritmo de Análise do Valor Médio (MVA) para estudar o desempenho de um sistema de filas aplicado a diferentes cenários e configurações, e porfim apresentar os resultados obtidos nas principais métricas.

0.3 Metodologia de Implementação do Algoritmo

A implementação do algoritmo de Análise do Valor Médio (MVA) foi desenvolvida em C++ e o fluxo de execução do programa foi dividido em três etapas principais:

0.3.1 Coleta de Parâmetros

Inicialmente, o programa interage com o usuário por meio de uma interface de console para coletar as variáveis de entrada do modelo: a população de clientes (N), o número de recursos (K) e, para cada um deles, um nome descritivo, seu tempo médio de serviço (S_i) e a contagem de visitas (V_i).

0.3.2 Processamento Iterativo (Núcleo MVA)

O coração do programa reside em um laço `for` que itera de $n = 1$ até N. Este laço implementa a lógica fundamental do MVA, onde o vetor N_i (número médio de clientes por recurso) atua como a variável de estado que conecta uma iteração à seguinte. Dentro de cada iteração, a sequência de cálculo é estritamente seguida:

- Primeiramente, calcula-se o tempo de resposta de cada recurso (R_i), utilizando o estado N_i da iteração $n - 1$.
- Em seguida, o tempo de resposta do sistema (R_0) é determinado.
- A partir dele, a vazão do sistema (X_0) é encontrada.
- Finalmente, o estado N_i é atualizado para a população n, preparando o programa para a próxima iteração.

Algorithm 1 Algoritmo MVA

Entradas:

N: Número total de clientes
K: Número total de recursos
S: Vetor com os tempos de serviço de cada recurso $(S_1, S_2, ..., S_K)$
V: Vetor com o número de visitas a cada recurso $(V_1, V_2, ..., V_K)$

Saídas:

U_{final}: Vetor com a utilização final de cada recurso
R_{final}: Vetor com o tempo de resposta final de cada recurso
L_{final}: Vetor com o número médio de clientes final em cada recurso
$X_{0,final}$: Throughput final do sistema

1: **procedure** MVA(N, K, S, V)
2: $L \leftarrow$ vetor de tamanho K preenchido com zeros ▷ Inicializa $N_i(0) = 0$
3: **for** $n \leftarrow 1$ to N **do** ▷ Laço principal para cada cliente
4: $R_{atual} \leftarrow$ vetor de tamanho K
5: $R_{0,atual} \leftarrow 0$
6: **for** $k \leftarrow 1$ to K **do** ▷ Calcula para cada recurso
7: $R_{atual}[k] \leftarrow S_k \times (1 + L_k)$ ▷ L_k é o $N_i(n-1)$
8: $R_{0,atual} \leftarrow R_{0,atual} + V_k \times R_{atual}[k]$
9: **end for**
10: $X_{0,atual} \leftarrow n / R_{0,atual}$
11: **for** $k \leftarrow 1$ to K **do** ▷ Atualiza o estado para a próxima iteração
12: $L_k \leftarrow R_{atual}[k] \times V_k \times X_{0,atual}$
13: **end for**
14: **end for**
15: **retornar** L, R_{atual}, $X_{0,atual}$ ▷ Retorna as métricas da última iteração
16: **end procedure**

0.3.3 Fórmulas Utilizadas

As métricas são calculadas iterativamente conforme o algoritmo MVA, utilizando as fórmulas da Tabela 1.

Table 1: Fórmulas do Algoritmo MVA

Métrica	**Fórmula**
Tempo de Resposta em i	$R_i(N) = S_i \cdot (1 + L_i(N-1))$
Tempo Total de Resposta	$R(N) = \sum_i V_i \cdot R_i(N)$
Vazão do Sistema	$X(N) = \frac{N}{R(N)}$
Vazão na Recurso i	$X_i(N) = V_i \cdot X(N)$
Utilização do Recurso i	$U_i(N) = X_i(N) \cdot S_i$
Número Médio na Recurso i	$L_i(N) = X_i(N) \cdot R_i(N)$

0.4 Experimentos e Resultados

Apos a implementação do algorítimo foi feito feito uma definição de 3 cenários, onde cada cenário simula um ambiente com diferentes configurações de recursos, tempos de serviço e número de clientes. Em seguida foi feita a conferencia dos valores obtidos pelo sistema e os cálculos matemáticos feito pelos participantes deste trabalho. Os resultados são então exibidos em uma tabela formatada, permitindo uma análise clara do desempenho geral do sistema e de cada componente individualmente. Além disso, os dados foram armazenados em arquivos *output* para serem analisados de forma gráfica utilizando python.

Cenário 1 – Academia

Parâmetros:

- Número de clientes: 4
- Recursos: Musculação, Esteira, Bicicleta

Table 2: Resultados do Cenário 1 – Academia

Recurso	**Utilização**	**Tempo Resp. (R)**	**Nº Médio Clientes (L)**	**Tempo Espera (W)**
Musculação	0.5506	8.9460 min	0.9852	4.9261 min
Esteira	0.8810	21.0632 min	2.3197	18.5575 min
Bicicleta	0.4405	6.3116 min	0.6951	2.7804 min

Métricas Gerais:

- Tempo de Resposta Total: 72.6416 min

- Throughput: 0.0551 clientes/min (≈ 3.30 clientes/hora)

Conclusão: A esteira apresenta o maior tempo de resposta e utilização, tornando-se o principal gargalo. A bicicleta mostra-se subutilizada.

Imagem do Cenário 1 – Academia

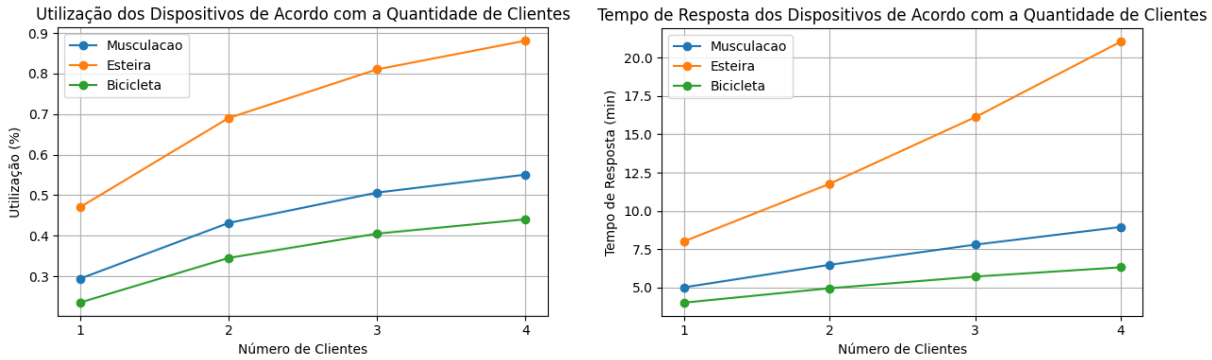

Figure 1: Gráficos comparativos por cenário: Utilização e Tempo de Resposta por Recurso

Cenário 2 – Sistema de Mensageria

Parâmetros:

- Número de clientes: 15
- Recursos: CPU, Disco, Impressora

Table 3: Resultados do Cenário 2 – Sistema de Mensageria

Recurso	Utilização	Tempo Resp. (R)	Nº Médio Clientes (L)	Tempo Espera (W)
CPU	0.9826	71.1827 min	9.9919	69.9432 min
Disco	0.8422	47.1471 min	4.4120	39.7081 min
Impressora	0.3743	6.3699 min	0.5961	2.3844 min

Métricas Gerais:

- Tempo de Resposta Total: 320.5820 min
- Throughput: 0.0468 clientes/min (≈ 2.81 clientes/hora)

Conclusão: A CPU está praticamente saturada, sendo o principal gargalo. O tempo de resposta geral é elevado. A impressora tem pouca utilização.

Imagem do Cenário 2 – Sistema de Mensageria

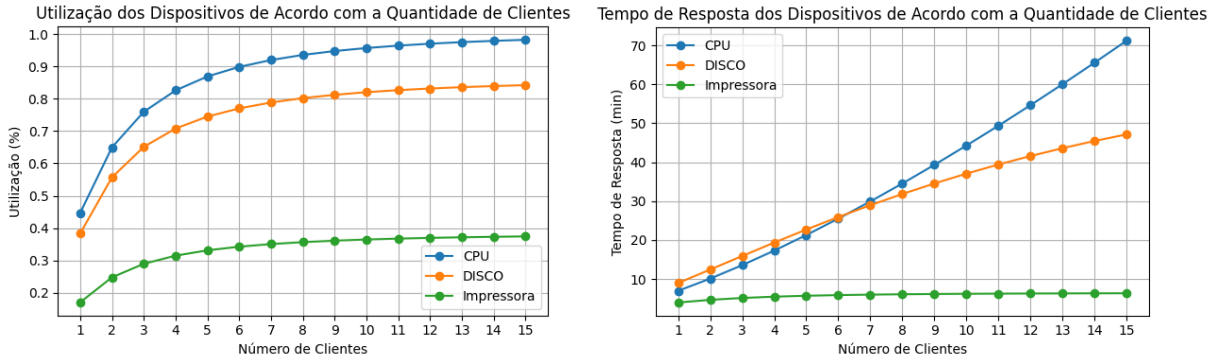

Figure 2: Gráficos comparativos por cenário: Utilização e Tempo de Resposta por Recurso

Cenário 3 – Banco

Parâmetros:

- Número de clientes: 7
- Recursos: Pagamento, Saque, Depósito

Table 4: Resultados do Cenário 3 – Banco

Recurso	Utilização	Tempo Resp. (R)	Nº Médio Clientes (L)	Tempo Espera (W)
Pagamento	0.7564	12.3158 min	2.3289	9.3158 min
Saque	0.1891	2.4520 min	0.2318	0.4637 min
Depósito	0.9455	46.9504 min	4.4392	44.3921 min

Métricas Gerais:

- Tempo de Resposta Total: 74.0341 min
- Throughput: 0.0946 clientes/min ($\approx$ 5.67 clientes/hora)

Conclusão: O recurso de depósito representa o principal gargalo. O recurso de saque apresenta baixa utilização, mostrando-se ocioso.

Imagem do Cenário 3 – Banco

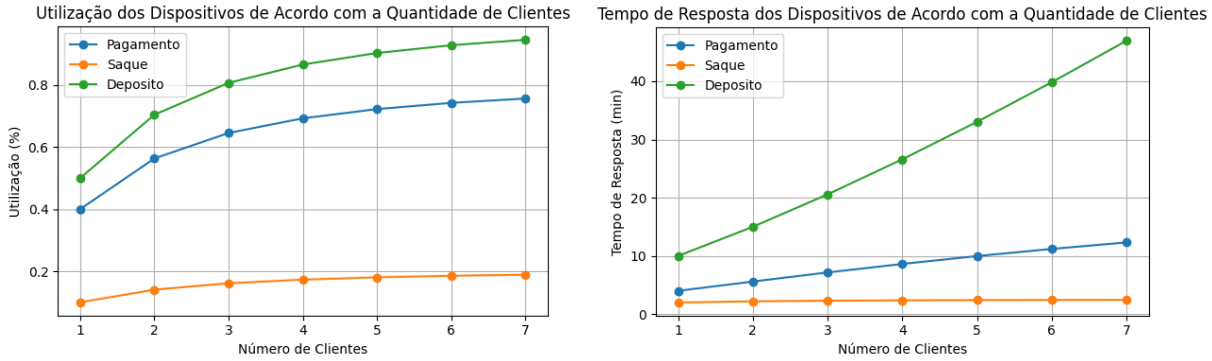

Figure 3: Gráficos comparativos por cenário: Utilização e Tempo de Resposta por Recurso

Como observado nos resultados apresentados, cada cenário possui um perfil de utilização e gargalos distintos. No cenário da **academia**, a esteira destacou-se como o principal ponto de contenção, com alta utilização e tempo de resposta elevado, enquanto a bicicleta permaneceu subutilizada. Já no **sistema de mensageria**, a **CPU** operou próxima do limite, demonstrando saturação e contribuindo significativamente para o alto tempo de resposta global. Por outro lado, no **cenário bancário**, o recurso de **depósito** foi o que mais impactou o desempenho do sistema, com grande tempo de espera e utilização elevada, enquanto o recurso de saque mostrou-se pouco demandado.

Além da validação e cálculo dos valores médios por meio do algoritmo MVA, foram gerados gráficos que ilustram a **evolução da utilização dos dispositivos conforme o número de clientes aumenta** em cada cenário. Essa análise dinâmica permite visualizar a escalabilidade e o comportamento dos recursos à medida que a carga do sistema cresce, fornecendo subsídios importantes para o dimensionamento e otimização do sistema analisado.

0.5 Conclusão

Através da implementação do algoritmo de Análise de Valor Médio (MVA) em linguagem C++, foi possível modelar e analisar com eficiência o desempenho de sistemas representados como redes fechadas de filas. A ferramenta desenvolvida demonstrou-se capaz de calcular as principais métricas de desempenho para diferentes cenários de carga e configuração.

Os experimentos revelaram gargalos distintos em cada cenário, além disso, foram gerados gráficos que ilustram a evolução da utilização dos recursos conforme o número de clientes aumenta. Essa análise visual reforça a importância da modelagem e simulação na identificação de pontos críticos e no apoio a decisões de otimização e dimensionamento de sistemas.

0.5.1 Melhorias Futuras

- Adicionar interface gráfica com gráficos de tempo de resposta e utilização.
- Transformar em uma biblioteca que pode ser utilizada em serviços de fila.