UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Matheus Alves Almeida

FALCOM: Framework Adaptativo baseado em Lógica Fuzzy para Controle de Middleware Orientado a Mensagem

Recife

2025

Matheus Alves Almeida

FALCOM: Framework Adaptativo baseado em Lógica Fuzzy para Controle de Middleware Orientado a Mensagem

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Mestre Profissional em Ciência da Computação.

Área de Concentração: Sistemas Distribuídos

Orientador (a): Nelson Souto Rosa

Recife

2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Almeida, Matheus Alves.
FALCOM: Framework Adaptativo baseado em Lógica Fuzzy para Controle de Middleware Orientado a Mensagem / Matheus Alves Almeida. - Recife, 2025.
95f.: il.

Dissertação (Mestrado) - Universidade Federal de Pernambuco, Centro de Informática, Programa de Pós-graduação em Ciência da Computação, 2025.
Orientação: Nelson Souto Rosa.
Inclui referências.

1. Controladores Fuzzy; 2. Middleware Orientado a Mensagem; 3. Algoritmos Genéticos; 4. Teoria do Controle. I. Rosa, Nelson Souto. II. Título.

UFPE-Biblioteca Central

Matheus Alves Almeida

“FALCOM: Framework Adaptativo baseado em Lógica Fuzzy para Controle de Middleware Orientado a Mensagem”

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Redes de Computadores e Sistemas Distribuídos.

Aprovado em: 24/07/2025.

BANCA EXAMINADORA

_____

Prof. Dr. David Junio Mota Cavalcanti
Centro de Informática / UFPE

_____

Prof. Dr. Admilson de Ribamar Lima Ribeiro
Departamento de Computação / UFS

_____

Prof. Dr. Nelson Souto Rosa
Centro de Informática / UFPE
(**orientador**)

Eu dedico esta dissertação à minha família. Obrigado por tudo!

Agradecimentos

Aos meus pais, Chirlley Núbia e Erenilson Jesus, por todo amor, cuidado e pelos inúmeros sacrifícios feitos ao longo da vida, especialmente no que se refere à minha educação. Sem o exemplo de esforço e dedicação de vocês, nada disso teria sido possível. Eu amo vocês! Ao meu irmão, Érik Alves, pela amizade constante, companheirismo e apoio em todos os momentos importantes da minha trajetória.

À minha tia Raquel (tia Kel), por acreditar em mim incondicionalmente e pelo apoio constante ao longo dessa jornada, mesmo à distância. Obrigado por cada palavra de incentivo, por sempre me lembrar do meu valor e por ser um exemplo de força, carinho e generosidade.

À minha companheira, Lívia Millena, meu amor e meu porto seguro. Agradeço profundamente pelo apoio incondicional não apenas durante o mestrado, mas desde o início da nossa história. Sua presença foi essencial para que eu mantivesse a motivação e a serenidade ao longo dessa caminhada. Eu amo você!

Ao grande professor Dr. Nelson Rosa, sou profundamente grato por tê-lo como orientador. Agradeço por toda a paciência, dedicação e confiança depositadas em mim, bem como por ter proporcionado um ambiente que favoreceu o pleno desenvolvimento desta pesquisa. O senhor tem sido um grande exemplo para mim desde os tempos da graduação.

Ao meu grande amigo Gabriel Nogueira, por compartilhar essa jornada comigo, pelas conversas, trocas de ideias e pelas boas memórias que levarei comigo.

Aos meus amigos do trabalho, do melhor time de dados em linha reta de Pernambuco, agradeço pela compreensão nos momentos de ausência e pelo apoio genuíno ao longo dessa jornada. Obrigado por tornarem os dias difíceis mais leves, com incentivo, companheirismo e bom humor.

À Universidade Federal de Pernambuco, por ter sido o espaço onde pude crescer acadêmica e pessoalmente, e aos colegas do grupo de pesquisa GFADS, pelo ambiente colaborativo e pelas contribuições valiosas.

Por fim, a todos os amigos que, de alguma forma, me apoiaram ou contribuíram para essa trajetória, o meu mais sincero agradecimento.

Resumo

Middleware orientado a Mensagem (MOM) tem sido amplamente adotado para a comunicação entre componentes de sistemas distribuídos nos últimos anos. O modelo produtor/consumidor, usado por MOMs, exerce um papel fundamental na infraestrutura desses sistemas ao oferecer comunicação assíncrona, baixo acoplamento entre componentes e escalabilidade. Devido à natureza distribuída dos MOMs, seus componentes operam sob condições heterogêneas, com diferentes configurações e graus de autonomia. Essa característica, embora traga vantagens importantes em termos de flexibilidade e escalabilidade, introduz desafios, como o controle do fluxo de mensagens entregues aos consumidores. Dependendo da capacidade de processamento dos consumidores, esses elementos podem ficar ociosos ou sobrecarregados com mensagens, comprometendo o desempenho geral do sistema. Para lidar com esses desafios, abordagens baseadas em regras fixas ou controle manual vêm sendo utilizadas, mas frequentemente se mostram limitadas em cenários dinâmicos, impactando negativamente a estabilidade, o *throughput* e a qualidade do serviço. Neste contexto, estudos prévios exploram o uso de técnicas da Teoria do Controle para promover ajustes dinâmicos no fluxo de mensagens. No entanto, persistem limitações relacionadas à dificuldade de adaptação dos controladores às variações do ambiente e à necessidade de sintonização manual dos parâmetros. Diante desse cenário, esta dissertação propõe o framework adaptativo FALCOM, uma solução autoadaptativa para o controle de fluxo em MOMs. O FALCOM combina lógica fuzzy com algoritmos genéticos, explorando os pontos fortes de cada abordagem. A lógica fuzzy oferece uma maneira de lidar com incertezas e relações não lineares. Por sua vez, os algoritmos genéticos são responsáveis por gerar e ajustar automaticamente as regras do mecanismo de inferência fuzzy, viabilizando a adaptação contínua do controlador em tempo de execução. A avaliação experimental do FALCOM utilizou métricas de desempenho relacionadas tanto à estabilidade quanto precisão, em cenários com metas estáticas e dinâmicas. Os resultados desta dissertação demonstraram que a abordagem fuzzy adaptativa, aliada a técnicas evolucionárias, é eficaz na manutenção do fluxo de mensagens desejado mesmo sob condições variáveis de carga, representando uma alternativa promissora para o controle inteligente e autônomo em MOMs.

Palavras-chaves: Controladores Fuzzy. Middleware Orientado a Mensagem. Algoritmos Genéticos. Teoria do Controle.

Abstract

Message-oriented middleware (MOM) has been widely adopted to facilitate communication between components in distributed systems. The publisher/subscriber model employed by MOMs plays a fundamental role in system infrastructure by providing asynchronous communication, loose coupling between components, and enhanced scalability. Due to their distributed nature, MOM components operate under heterogeneous conditions with varying configurations and degrees of autonomy. While this characteristic offers significant advantages in terms of flexibility and scalability, it also introduces challenges, particularly in controlling message flow to consumers. Depending on their processing capacity, consumers may become either idle or overloaded with incoming messages, thereby compromising overall system performance. To address these challenges, fixed-rule and manual control approaches have been employed; however, they often prove inadequate in dynamic scenarios, negatively affecting system stability, throughput, and quality of service. Previous studies have explored Control Theory techniques to enable dynamic message flow adjustments, but limitations persist, particularly regarding controller adaptability to changing environments and the requirement for manual tuning of control parameters. In response to these limitations, this dissertation proposes an adaptive framework FALCOM, for dynamic message flow control in MOMs. The FALCOM combines fuzzy logic with genetic algorithms, leveraging the complementary strengths of both approaches. Fuzzy logic provides a mechanism for handling uncertainties and nonlinear relationships, while genetic algorithms automatically generate and adjust fuzzy inference rules, enabling continuous controller adaptation at runtime. The experimental evaluation of FALCOMwas conducted using performance metrics that assess both stability and accuracy in scenarios with static and dynamic setpoints. The results demonstrate that the adaptive fuzzy approach, combined with evolutionary techniques, effectively maintains desired message flow under variable load conditions, representing a promising solution for intelligent and autonomous control in MOMs.

Keywords: Fuzzy Controllers. Message-Oriented Middleware. Genetic Algorithms. Control Theory.

Lista de Figuras

Figura 1 – Arquitetura de uma aplicação construída sobre um MOM 22
Figura 2 – Arquitetura do MAPE-K 24
Figura 3 – Visão geral de um sistema adaptativo 25
Figura 4 – Visão geral de um *feedback control loop* 26
Figura 5 – Arquitetura de um controlador fuzzy 30
Figura 6 – Arquitetura geral de um algoritmo genético 36
Figura 7 – Visão geral da integração dos controladores com o Consumidor 41
Figura 8 – Diagrama do *feedback loop* 50
Figura 9 – Impacto do valor do *PC* na taxa de chegada de mensagens 52
Figura 10 – Resultado com ajustes de regras específicos 53
Figura 11 – Resultado com ajustes de regras para cada *setpoint* 53
Figura 12 – Diagrama simplificado da configuração usada nos experimentos 55
Figura 13 – NRMSE médio por função de fuzzificação e tipo de carga 61
Figura 14 – GR médio por função de fuzzificação e tipo de carga 62
Figura 15 – GR médio por função de defuzzificação e tipo de carga 63
Figura 16 – NRMSE médio por função de fuzzificação e tipo de carga 64
Figura 17 – GR médio por função de fuzzificação e tipo de carga 65
Figura 18 – GR médio por método de defuzzificação e tipo de carga 66
Figura 19 – NRMSE médio por método de defuzzificação e tipo de carga 67
Figura 20 – Fluxo de mensagens controlados pelos três melhores controladores experimentados 77
Figura 21 – Fluxo de mensagens controlados pelos três piores controladores experimentados 78
Figura 22 – Fluxo de mensagens sob atuação dos controladores Fuzzy 1, 2 e 4 78
Figura 23 – Fluxo de mensagens sob atuação dos controladores Fuzzy 21, 22 e 24 79

Lista de tabelas

Tabela 2 – Variáveis linguísticas dos controladores fuzzy 42
Tabela 3 – Configurações dos controladores fuzzy 45
Tabela 4 – Fatores e níveis 59
Tabela 5 – Desempenho dos controladores para carga de 1000 msgs/s com meta fixa. . . . 70
Tabela 6 – Desempenho dos controladores para carga de 2500 msgs/s com meta fixa. . . . 71
Tabela 7 – Desempenho dos controladores para carga de 5000 msgs/s com meta fixa. . . . 72
Tabela 8 – Desempenho dos controladores para carga de 5000 msgs/s com metas variáveis 73
Tabela 9 – Desempenho dos controladores para carga de 2500 msgs/s com metas variáveis. 75
Tabela 10 – Desempenho dos controladores para carga de 1000 msgs/s com metas variáveis. 76
Tabela 11 – Resumo comparativo entre abordagens de controle de fluxo em MOMs 85

Lista de abreviaturas e siglas

ACID	*Atomicity, Consistency, Isolation, Durability*
BSC	*Bisector*
CARG	*Compound Annual Growth Rate*
CoG	*Center of Gravity*
DOM	*Document Object Model*
ERP	*Enterprise Resource Planning*
FIFO	*First In, First Out*
GR	*Goal Range*
HPA	*Horizontal Pod Autoscaling*
IoT	*Internet of Things*
LoM	*Largest of Maximum*
LQR	*Linear Quadratic Regulator*
MoM	*Message-Oriented Middleware*
MoM	*Mean of Maximum*
MPC	*Model Predictive Control*
NRMSE	*Normalized Root Mean Square Error*
PC	*Prefetch Count*
PID	*Proportional-Integral-Derivative*
QoS	*Quality of Service*
RPC	*Remote Procedure Call*

Sumário

1 INTRODUÇÃO 13
1.1 MOTIVAÇÃO 13
1.2 PROBLEMA 15
1.3 OBJETIVOS 18
1.4 ESTRUTURA DA DISSERTAÇÃO 20
2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 MIDDLEWARE ORIENTADO A MENSAGEM 21
2.2 SISTEMAS ADAPTATIVOS 24
2.3 TEORIA DO CONTROLE 26
2.4 CONTROLADORES FUZZY 29
2.5 ALGORITMOS GENÉTICOS 35
2.6 CONSIDERAÇÕES FINAIS 39
3 FALCOM 40
3.1 VISÃO GERAL 40
3.2 REPOSITÓRIO DE CONTROLADORES 41
3.3 ALGORITMO GENÉTICO 45
3.4 IMPLEMENTAÇÃO 48
3.5 USANDO FALCOM 48
3.6 CONSIDERAÇÕES FINAIS 54
4 EXPERIMENTOS E RESULTADOS 55
4.1 OBJETIVOS 55
4.2 AMBIENTE 55
4.3 MÉTRICAS 56
4.4 FATORES 58
4.5 PROJETO DOS EXPERIMENTOS 59
4.6 RESULTADOS E DISCUSSÕES 60
4.7 CONSIDERAÇÕES FINAIS 79
5 TRABALHOS RELACIONADOS 80

5.1 VISÃO GERAL 80
5.2 TEORIA DO CONTROLE APLICADA A MIDDLEWARE 81
5.3 LÓGICA FUZZY EM SISTEMAS COMPUTACIONAIS 82
5.4 COMPUTAÇÃO EVOLUCIONÁRIA E LÓGICA FUZZY 83
5.5 CONTRIBUIÇÕES 84
5.6 CONSIDERAÇÕES FINAIS 85
6 CONCLUSÃO E TRABALHOS FUTUROS 86
6.1 CONCLUSÕES 86
6.2 CONTRIBUIÇÕES 87
6.3 LIMITAÇÕES 88
6.4 TRABALHOS FUTUROS 89
REFERÊNCIAS 90

1 Introdução

Este capítulo apresenta uma visão geral da pesquisa desenvolvida nesta dissertação. Inicialmente, é apresentada a motivação do trabalho. Em seguida, o problema de pesquisa é detalhado, com ênfase nos desafios técnicos e conceituais que motivaram o desenvolvimento da solução. Por fim, são apresentados os objetivos da dissertação, bem como a estrutura geral do documento.

1.1 Motivação

Aplicações distribuídas são caracterizadas pela sua natureza intrinsecamente heterogênea e descentralizada. Essa heterogeneidade abrange um amplo conjunto de variáveis relacionadas ao ambiente de implantação, tais como a infraestrutura de rede, os recursos de hardware, os sistemas operacionais e até as linguagens de programação utilizadas no desenvolvimento das aplicações (COULOURIS et al., 2012). Considerando que o funcionamento adequado de sistemas distribuídos depende da comunicação eficiente entre seus componentes, torna-se imprescindível a adoção de mecanismos que viabilizem a interoperabilidade entre elementos tão diversos. Nesse contexto, é necessário estabelecer padrões comuns que permitam a troca de mensagens de forma transparente e eficiente. Com o objetivo de abstrair essa complexidade, existem os sistemas de middleware (BERNSTEIN, 1996), que são camadas intermediárias de software responsáveis por fornecer serviços de comunicação padronizados.

Esses sistemas de middleware oferecem interfaces que ocultam a heterogeneidade de aplicações distribuídas, proporcionando uma visão unificada e simplificada para os desenvolvedores das mesmas (TANENBAUM; STEEN, 2014). De modo geral, os sistemas de middleware são classificados em quatro modelos principais (EMMERICH, 2000): *Remote Procedure Call* (RPC), transacionais, *Distributed Object Middleware* (DOM) e sistemas de middleware orientados a mensagens (MOMs), cada um deles adequado a diferentes modelos de interação e requisitos de acoplamento entre os componentes do sistema.

Um middleware baseado em RPC, como o gRPC (GRPC, 2025) e Apache Thrift (APACHE, 2025b), segue um modelo de comunicação síncrona e direta. No qual o cliente envia uma solicita-

ção e aguarda bloqueado até que o servidor processe a requisição e envie uma resposta. Normalmente, esse modelo é utilizado em cenários onde o sistema exige baixa latência e chamadas diretas, como em aplicações web, aplicações baseadas em microsserviços e alguns tipos de sistemas de *backend*. Porém, tais soluções apresentam limitações devido às suas propriedades, especialmente por promoverem forte acoplamento e dependência temporal entre os componentes, o que dificulta a escalabilidade, a tolerância a falhas e a flexibilidade da arquitetura.

Os sistemas de middleware transacionais, por sua vez, são projetados para garantir propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) em operações distribuídas. Eles são amplamente utilizados em sistemas que demandam alta confiabilidade e integridade dos dados, como bancos, sistemas bancários e *Enterprise Resource Planning* (ERP, em português, Sistemas de Planejamento de Recursos Empresariais). Esse tipo de middleware conta com mecanismos de controle de concorrência distribuído e gerenciamento de transações de longa duração. Isso garante que múltiplas operações realizadas em diferentes serviços sejam concluídas com sucesso ou, em caso de falha, completamente desfeitas, mantendo o estado consistente do sistema como um todo. Entre os sistemas de middleware mais recentes com suporte a transações distribuídas, destacam-se o Seata, utilizado em ambientes de microsserviços baseados em Java, e o Styx styxgraph, que oferece suporte a funções transacionais em ambientes de computação *serverless*.

Já os DOMs surgem como uma extensão do paradigma orientado a objetos para ambientes distribuídos, permitindo que objetos localizados em diferentes máquinas interajam como se estivessem no mesmo espaço de memória. Esse modelo oferece transparência de localização e acesso, além de suporte à herança e polimorfismo em ambientes distribuídos.

Os MOMs adotam o modelo produtor/consumidor (*publish/subscriber*), onde produtores publicam mensagens que são posteriormente consumidas por um ou mais consumidores de forma concorrente e independente. Esse modelo se caracteriza por baixo acoplamento entre produtores e consumidores, comunicação assíncrona e escalabilidade na distribuição de mensagens (TANENBAUM; STEEN, 2014; COULOURIS et al., 2012). Além disso, a popularização de sistemas baseados em microsserviços, *big data* e computação em nuvem também contribuiu para a adoção desse modelo. Os MOMs promovem a troca de informações na forma de mensagens através de um componente chamado *Broker*. Soluções como RabbitMQ (RabbitMQ, 2025), Apache Kafka (Apache, 2025) e Mosquitto (Eclipse Mosquitto, 2025) são exemplos de MOMs e passaram a ser amplamente utilizadas em contextos variados, incluindo integrações entre serviços (SOUTO et al., 2004), redes IoT (BAO et al.,

2016), aplicações financeiras (SUBRAMONI et al., 2008) e pipelines de processamento de dados (IM et al., 2024).

Este documento foca no estudo dos MOMs, que apresentou um crescimento expressivo no mercado nos últimos anos. Relatórios recentes indicam que o mercado global de MOMs foi avaliado em aproximadamente 3,8 bilhões de dólares em 2024, com projeções de atingir 8,5 bilhões até 2033, apresentando uma Taxa de Crescimento Anual Composta (CAGR, *Compound Annual Growth Rate*) de 9,5% (Verified Market Reports, 2025). Outra análise aponta um crescimento ainda mais acelerado, com estimativas que variam de 2,78 bilhões em 2024 para 7,66 bilhões até 2033, impulsionado por uma CAGR de 10,5% (Business Research Insights, 2025). Também se observou uma adoção expressiva de plataformas como o RabbitMQ, que registrou um aumento de 50% na criação de clusters na região da África em 2024, em comparação com o ano anterior, conforme apontado no relatório anual da CloudAMQP (CloudAMQP, 2025), reforçando essa tendência de crescimento.

Portanto, compreender o funcionamento dos MOMs e desenvolver mecanismos que garantam seu desempenho e estabilidade torna-se fundamental. À medida que esses sistemas passam a ser cada vez mais incorporados em arquiteturas contemporâneas, como microsserviços, sistemas baseados em eventos e aplicações em nuvem, surgem desafios relacionados ao gerenciamento dinâmico de carga, controle de filas, balanceamento de *throughput* e manutenção da qualidade de serviço (QoS, *Quality of Service*) frente a variações imprevisíveis no ambiente.

1.2 Problema

Apesar das inúmeras vantagens dos MOMs, eles introduzem novos desafios, especialmente relacionados ao controle de fluxo. A comunicação assíncrona, característica desse modelo de middleware, dificulta a coordenação entre os elementos do sistema, tornando mais complexa a tarefa de balancear a carga entre produtores e consumidores de mensagens. Diferenças na capacidade de processamento dos consumidores, variações na taxa de chegada de mensagens e instabilidades nos recursos computacionais, como CPU, memória ou largura de banda, podem gerar sobrecargas nos consumidores, perdas de dados e aumento significativo da latência no sistema.

Em tais cenários, o *broker* pode, por exemplo, inundar um consumidor com mensagens além de sua capacidade de processamento, provocando congestionamento e eventualmente a queda de desempenho ou até a falha do consumidor. Por outro lado, pode ocorrer o subaproveitamento de

recursos, mantendo um consumidor ocioso mesmo diante de uma demanda que poderia ser redistribuída de forma mais eficiente. Ambos os casos comprometem a eficiência e a robustez do sistema como um todo, afetando diretamente os requisitos de QoS. Além disso, é importante destacar que o *broker* geralmente é implantado em uma infraestrutura computacional mais robusta em relação aos produtores e consumidores.

Brokers normalmente oferecem mecanismos de configuração de parâmetros relacionados ao fluxo de mensagens, como limites de fila, políticas de rejeição e priorização de tópicos. Essas configurações, em sua maioria, são estáticas. Elas são definidas previamente a execução do sistema e permanecem inalteradas durante sua operação, mesmo diante de variações nas condições de carga e desempenho. Essa limitação evidencia a necessidade de mecanismos de controle mais flexíveis e adaptativos, capazes de ajustar seu comportamento dinamicamente, em tempo de execução, conforme mudanças nas condições operacionais.

No entanto, o gerenciamento do fluxo em sistemas de MOMs ainda é, predominantemente, realizado de forma manual ou com base em heurísticas específicas, muitas vezes dependentes do conhecimento prévio do domínio de aplicação. No Apache Pulsar (APACHE, 2025a), por exemplo, o consumidor deve explicitar sua capacidade de consumo ao *broker*, que regula a entrega com base nesse limite. Já o RabbitMQ (RabbitMQ, 2025) utiliza uma abordagem baseada em janelas fixas de mensagens *in-flight*, funcionando de forma análoga ao protocolo TCP. Outras soluções adotam modelos *pull-based*, nos quais o consumidor solicita mensagens apenas após o processamento das anteriores, deixando o *broker* em um papel passivo.

Diante desse cenário, diversas estratégias têm sido propostas para gerenciar o fluxo de mensagens em MOMs, com o objetivo de evitar problemas como sobrecarga, aumento da latência e perda de mensagens, além de garantir que o sistema opere de forma estável e responsiva, mesmo em ambientes dinâmicos e sujeitos a incertezas. Essas estratégias podem ser organizadas em três grandes categorias, conforme a abordagem adotada: (i) métodos heurísticos, que aplicam regras empíricas ou baseadas na experiência; (ii) técnicas que utilizam medições diretas das condições operacionais do sistema para a tomada de decisão; e (iii) abordagens fundamentadas na teoria do controle e na inteligência computacional, que aplicam modelos matemáticos e algoritmos adaptativos para oferecer soluções mais robustas e dinâmicas.

No primeiro grupo, encontram-se abordagens que empregam métodos heurísticos, frequentemente baseados na experiência prática, conhecimento prévio do domínio ou regras empíricas esta-

belecidas por administradores de sistemas. Tais soluções são, em geral, mais simples de implementar e possuem menor custo computacional, sendo adequadas para sistemas em que a modelagem formal é inviável ou desnecessária.

Já o segundo conjunto de trabalhos se apoia em medições diretas das condições operacionais, como tempo de resposta da rede, taxa de consumo dos consumidores ou utilização de recursos, e aplica ajustes simples ao fluxo de mensagens com base nessas observações (AMQP, 2023). Essas soluções são geralmente reativas e dependem de limiares previamente definidos para acionar mecanismos de controle.

Por fim, no terceiro grupo, encontram-se abordagens que empregam princípios da teoria do controle (ABDELZAHER et al., 2008). Essas soluções utilizam modelos matemáticos para representar a dinâmica do sistema e usam controladores, como PID ou on/off, com o objetivo de manter o sistema estável. Trabalhos como (ROSA; CAVALCANTI, 2022; ROSA; CAVALCANTI, 2023; ROSA; CAVALCANTI, 2024; ALMEIDA, 2023; PIETZUCH; BHOLA, 2003; JOHANSSON, 2023; TANOMWONG; JAIKAEO, 2018) atuam com a modificação dinâmica de parâmetros do *broker* com o objetivo de adaptar o sistema a variações nas condições operacionais, mantendo o fluxo de mensagens controlado em tempo de execução. Em geral, essas abordagens monitoram determinados indicadores para identificar o ajuste adequado nos parâmetros do consumidor.

Esta dissertação foca no terceiro grupo, visto que esse tipo de abordagem permite a implementação de estratégias dinâmicas e sensíveis ao contexto operacional. Como exemplo, pode-se citar a situação em que um consumidor apresenta melhora em sua condição de sobrecarga, seja por liberação de recursos locais, seja por estabilização no consumo de mensagens. Nessa circunstância, o *broker* pode aumentar gradualmente o número de mensagens encaminhadas a esse consumidor, maximizando a utilização dos recursos disponíveis e melhorando o *throughput* do sistema como um todo. De forma complementar, atrasos observados na rede, como aumento no tempo de entrega das mensagens ou flutuações na latência, podem ser interpretados como indícios de congestionamento ou degradação de desempenho. Em resposta a esses sinais, o controlador pode induzir o *broker* a reduzir temporariamente a taxa de envio de mensagens, contribuindo para a estabilização do sistema e evitando a propagação de falhas em cascata.

Nesse contexto, vários controladores existentes foram experimentados sob diferentes circunstâncias e obtiveram bons resultados. No entanto, controladores como o controlador proporcional integral derivativo (PID, *Proportional–Integral–Derivative*) necessitam de ajuste fino nos seus pa-

râmetros antes de serem integrados a um sistema de controle. Esse processo, conhecido como sintonização, geralmente exige conhecimento especializado sobre a dinâmica do sistema, bem como testes empíricos ou métodos iterativos para calibrar os ganhos proporcional, integral e derivativo. Em sistemas distribuídos e dinâmicos, essa tarefa torna-se especialmente desafiadora, uma vez que os parâmetros ideais podem variar ao longo do tempo em função de alterações na carga, nos padrões de uso ou nas condições da infraestrutura.

Ainda nesse contexto, controladores mais recentes, como o HPA (Kubernetes, 2025) e o AsTAR (YANG et al., 2021), foram experimentados para contornar essas limitações, eliminando a necessidade de ajustes manuais frequentes. Contudo, ainda assumem premissas de linearidade e previsibilidade que nem sempre ocorrem na prática. Em ambientes distribuídos reais, o comportamento do sistema decorre da interação complexa entre múltiplas variáveis, muitas vezes com efeitos não lineares e não determinísticos, como a taxa de chegada de mensagens, o consumo de recursos, a latência da rede e a sobrecarga dos componentes.

Portanto, esta dissertação investiga mecanismos de controle capazes de responder de forma autônoma a flutuações no fluxo de mensagens no contexto de MOMs e que, de alguma forma, superem essas limitações destacadas.

1.3 Objetivos

Com base nas limitações observadas nas abordagens existentes de controle de fluxo, esta dissertação propõe um framework, chamado FALCOM, baseado na combinação entre lógica fuzzy e algoritmos genéticos (AGs). O FALCOM permite construir mecanismos de decisão para controle dinâmico do fluxo de mensagens em MOMs de forma mais flexível, considerando o grau de pertencimento das variáveis a conjuntos linguísticos definidos e permitindo ajustes contínuos com base nas condições do ambiente. Essa abordagem facilita a adaptação em tempo de execução sem a necessidade de parametrizações manuais recorrentes.

A lógica fuzzy foi escolhida por sua capacidade de representar incertezas e lidar com informações imprecisas de forma estruturada, por exemplo, permitindo a formulação de regras do tipo *"se a fila está alta e o consumo está baixo, então reduzir o ritmo de entrega"*, mesmo sem definir limiares exatos. O que a torna adequada para ambientes dinâmicos e sujeitos a variações não determinísticas. Essa abordagem permite construir mecanismos de decisão mais flexíveis, pois, em vez

de classificar uma variável apenas como verdadeira ou falsa, a lógica fuzzy permite atribuir valores intermediários. Por exemplo, uma variável pode assumir valores *moderado* ou *parcialmente alto*, o que representa de forma mais realista o comportamento de sistemas sujeitos a variações contínuas.

No entanto, para implementar esse tipo de controlador é necessário expertise no domínio da aplicação para que as regras usadas no mecanismo de inferência sejam eficientes. Visando reduzir essa necessidade de conhecimento do domínio da aplicação, foram empregados AGs. Os AGs possibilitam o ajuste automático das regras fuzzy por meio de processos que computam de forma autônoma as regras para o sistema em casos de baixa capacidade de controle. A ideia central é que o sistema evolua a geração de regras, observando seu próprio desempenho em tempo de execução, com menos intervenção humana.

Neste contexto, o objetivo principal da proposta é a construção de controladores autoadaptativos baseados em lógica fuzzy, capazes de ajustar dinamicamente os parâmetros do sistema com base em métricas observadas em tempo de execução, como, por exemplo, o tamanho da fila e a taxa de chegada de mensagens no consumidor. Esses controladores não apenas respondem a variações no ambiente, como também reconfiguram sua lógica de operação diante de baixos níveis de desempenho, promovendo maior autonomia e resiliência operacional.

Para alcançar este objetivo principal, foram definidos três objetivos específicos. O primeiro objetivo específico consiste no desenvolvimento de um framework modular que encapsula os controladores fuzzy adaptativos propostos. Este framework foi projetado para ser extensível e integrado facilmente a sistemas de mensageria, permitindo a experimentação com diferentes configurações e estratégias de controle. No framework, é possível decidir que função de pertinência ou método de defuzzificação usar. A arquitetura do framework contempla a separação entre os módulos de controle, monitoramento e ajuste de parâmetros.

O segundo objetivo refere-se à integração de um módulo de otimização baseado em algoritmos genéticos, cuja função é realizar a geração e a atualização automática das regras fuzzy utilizadas pelos controladores. Esse módulo opera de forma contínua ou periódica, identificando situações nas quais o desempenho do controle está aquém do esperado e, nesses casos, adaptando as regras de inferência para melhorar a resposta do sistema. Para o seu funcionamento, um módulo de monitoramento integrado é disparado quando as regras do conjunto atual do mecanismo de inferência do controlador já não geram bons resultados de controle.

O terceiro objetivo é a avaliação experimental do desempenho dos controladores fuzzy adapta-

tivos em cenários simulados de troca de mensagens entre produtores e consumidores, com foco no controle do fluxo entre o *broker* e o consumidor. Os experimentos foram conduzidos com o intuito de verificar a capacidade dos controladores em manter o desempenho do sistema dentro de faixas desejadas, mesmo diante de variações de carga.

Os experimentos conduzidos consideraram dois tipos diferentes de metas: fixas e variáveis. A motivação por trás disso foi tentar realizar uma simulação mais próxima do cenário real, onde há variações na meta por diferentes motivos. Os resultados obtidos demonstram que algumas variações do controlador fuzzy adaptativo superam as abordagens existentes, especialmente em cenários com variabilidade da meta.

1.4 Estrutura da dissertação

A partir deste ponto, o documento está organizado da seguinte forma: o Capítulo 2 apresenta os conceitos fundamentais necessários para a compreensão do trabalho desenvolvido. Em seguida, o Capítulo 3 descreve em detalhes a solução proposta, enquanto o Capítulo 4 aborda o planejamento dos experimentos e a análise dos resultados obtidos. o Capítulo 5 discute os trabalhos relacionados ao tema da dissertação e destaca as contribuições desta pesquisa. Por fim, o Capítulo 6 reúne as principais conclusões do estudo, discute suas limitações e propõe direções para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os principais conceitos teóricos que fundamentam o desenvolvimento deste trabalho. A Seção 2.1 aborda os conceitos básicos relacionados a middleware orientado a mensagem. Em seguida, a Seção 2.2 trata dos aspectos centrais dos sistemas adaptativos, enquanto a Seção 2.3 apresenta os fundamentos da teoria do controle, seguida da Seção 2.4 dedicada aos controladores fuzzy. A Seção 2.5 apresenta os princípios básicos dos algoritmos genéticos. Por fim, a Seção 2.6 traz as considerações finais do capítulo.

2.1 Middleware orientado a mensagem

Um middleware orientado a mensagem (MOM) é uma camada de software que permite que aplicações se comuniquem entre si de forma assíncrona, através da troca de mensagens, usando um modelo de comunicação amplamente usado: o produtor/consumidor (*publisher/subscriber*) (TANENBAUM; STEEN, 2014; EUGSTER et al., 2003). Essa camada é responsável por prover mecanismos para o envio e recebimento de mensagens, além de gerir o envio para os consumidores, garantindo o fraco acoplamento entre as partes comunicantes.

2.1.1 Arquitetura de um MOM

Um MOM é composto por dois elementos principais: o *Broker* e sua respectiva API de acesso, chamada de *Broker Client*. A Figura 1 apresenta uma visão geral de uma arquitetura construída sobre um MOM. O *Produtor* e o *Consumidor* não fazem parte do MOM em si, mas sim da aplicação construída sobre ele. Esses componentes utilizam a interface do *Broker Client* para publicar e consumir mensagens, respectivamente.

O *Broker*, ou provedor de mensagens, atua como intermediário na comunicação entre os componentes da aplicação. Sua função principal é receber, armazenar temporariamente e encaminhar mensagens de forma eficiente. Internamente, o *Broker* utiliza *Filas*, que armazenam as mensagens na ordem de chegada, obedecendo geralmente à política FIFO (*first in, first out*), na qual a primeira mensagem inserida é a primeira a ser entregue.

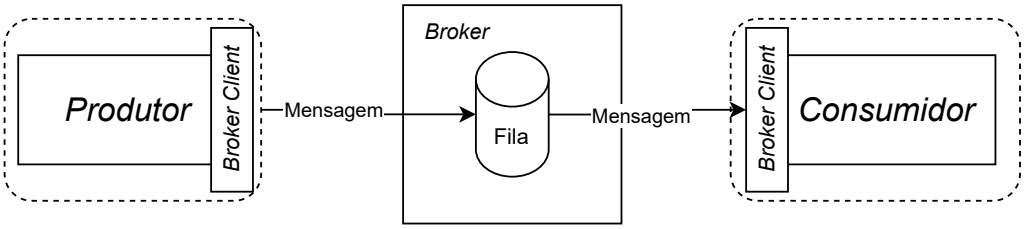

Figura 1 – Arquitetura de uma aplicação construída sobre um MOM

Além da intermediação, o *broker* também é responsável pela aplicação de políticas de roteamento, persistência e confiabilidade da entrega. Dentre as implementações mais amplamente utilizadas, destacam-se o RabbitMQ (RabbitMQ, 2025), o Apache Kafka (Apache, 2025) e o Mosquitto (Eclipse Mosquitto, 2025), cada uma com características específicas que os tornam mais adequados a diferentes demandas em termos de escalabilidade, latência e tolerância a falhas.

O *Produtor* é responsável por gerar e enviar mensagens para o sistema por meio da interface do *Broker Client*. O *Consumidor*, por sua vez, registra-se em filas ou tópicos de interesse e consome as mensagens distribuídas pelo *Broker*.

O fluxo de mensagens em um MOM inicia-se com a subscrição de consumidores a filas ou tópicos específicos, por meio da interface do *broker*. Então, o produtor realiza a publicação de uma mensagem no *broker*. Essa mensagem é então roteada conforme o modelo de mensageria adotado, armazenada em uma fila ou tópico e, subsequentemente, entregue aos consumidores pertinentes. Esses elementos operam em um ambiente sujeito a desafios técnicos recorrentes, como ordenação de mensagens, controle de congestionamento e garantia de entrega (HOHPE; WOOLF, 2015; TANENBAUM; STEEN, 2014; SALEHI; ZHANG; JACOBSEN, 2020).

A forma como as mensagens são entregues aos consumidores pode variar conforme o modelo de comunicação adotado. Nos modelos baseados em *filas*, cada mensagem é consumida por apenas um consumidor, promovendo o balanceamento de carga entre múltiplos consumidores. Já nos modelos baseados em *tópicos*, as mensagens são transmitidas para todos os consumidores interessados naquele tópico, promovendo difusão ou publicação-assinatura. A escolha entre esses modelos afeta diretamente o comportamento do sistema quanto à escalabilidade e redundância.

Adicionalmente, os MOMs usam protocolos de mensagens específicos que determinam a forma como os dados são serializados, transmitidos e recebidos, e.g., AMQP (*Advanced Message Queuing Protocol*) (AMQP Team, 2025), empregado por soluções como o RabbitMQ; o MQTT (*Message*

Queuing Telemetry Transport) (MQTT, 2025), voltado para aplicações como IoT; e o protocolo nativo do Apache Kafka (Apache, 2025). A escolha do protocolo influencia diretamente aspectos como latência, confiabilidade, interoperabilidade e suporte à QoS.

2.1.2 RabbitMQ

O RabbitMQ é uma solução *open-source* amplamente adotada, que implementa o protocolo AMQP (AMQP Team, 2025). Ao contrário de abordagens em que as mensagens são enviadas diretamente às filas, o RabbitMQ usa um componente intermediário denominado *Exchange*.

Há diferentes tipos de *exchanges*, e cada um implementa uma estratégia de roteamento com base nas chaves das mensagens. A chave trata-se de uma cadeia de caracteres associada à mensagem no momento da publicação, usada pelo *exchange* para decidir para quais filas a mensagem deve ser encaminhada. *Exchanges* do tipo *direct* realizam a entrega de mensagens com base na correspondência exata entre a chave da mensagem e a chave de *binding* associada à fila. O *exchange* do tipo *topic* permite correspondências por padrões, sendo apropriado para domínios com múltiplas categorias ou hierarquias de tópicos. Por sua vez, *exchanges* do tipo *fanout* replicam a mensagem para todas as filas vinculadas, enquanto as *exchanges* do tipo *headers* utilizam atributos do cabeçalho da mensagem como critério de roteamento (RabbitMQ, 2025).

As associações entre *exchanges* e filas são realizadas por meio de estruturas chamadas *bindings*, que estabelecem as condições sob as quais uma mensagem será encaminhada para determinada fila. O conjunto formado por *exchanges* e *bindings* oferece flexibilidade na definição da lógica de distribuição das mensagens.

Adicionalmente, o RabbitMQ provê um mecanismo de controle de fluxo a nível de consumidor, utilizando o parâmetro *prefetch count* (PC). Esse parâmetro define o número máximo de mensagens que o *broker* pode enviar a um consumidor sem que haja confirmação de recebimento (*acknowledgment*). Um valor mais restritivo de PC limita o volume de mensagens em trânsito, funcionando como um mecanismo de *backpressure*, o que é particularmente relevante em cenários com variações dinâmicas na taxa de consumo (RABBITMQ, 2025).

2.2 Sistemas Adaptativos

Um sistema adaptativo é capaz de modificar seu comportamento em tempo de execução em resposta a mudanças internas ou externas do sistema, com o objetivo de manter ou otimizar seu desempenho frente a condições variáveis (WEYNS, 2018).

Sistemas Adaptativos são popularmente estruturados usando-se o modelo MAPE-K (KEPHART; CHESS, 2003), como mostrado na Figura 2. No MAPE-K, a lógica de negócio (*Sistema gerenciado*) é gerenciada por um *Sistema gerenciador* organizado em quatro atividades: Monitoramento, Análise, Planejamento e Execução.

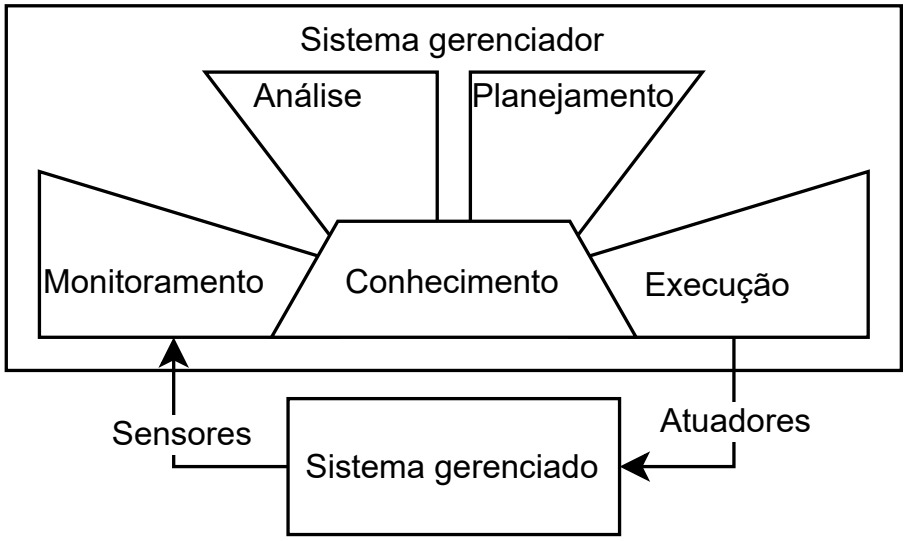

Figura 2 – Arquitetura do MAPE-K

A atividade de *Monitoramento* coleta os dados provenientes do *Sistema gerenciado*, como métricas de desempenho, uso de recursos ou eventos de falha. Em seguida, a *Análise* interpreta os dados coletados, identificando padrões, anomalias ou tendências que possam indicar a necessidade de adaptação. A etapa de *Análise* pode empregar técnicas de inferência, aprendizado de máquina ou métodos heurísticos. A atividade de *Planejamento*, por sua vez, define as ações necessárias para realizar a adaptação, sempre considerando as políticas e objetivos estabelecidos. Finalmente, a etapa de *Execução* implementa as mudanças planejadas, modificando o comportamento do sistema de forma controlada. Todo o ciclo é sustentado por uma base de *Conhecimento* compartilhada, que

armazena modelos, políticas e históricos relevantes, servindo de suporte à tomada de decisão em todas as fases da adaptação.

As estratégias de adaptação podem ser classificadas em dois grupos: reativas e proativas (WEYNS, 2018). A adaptação reativa ocorre quando uma ou mais metas do sistema já foram violadas. Por outro lado, uma estratégia proativa evita a violação de uma ou mais metas atuando preventivamente. A adaptação proativa tenta antecipar mudanças e ajustar o comportamento do sistema de forma preventiva.

A Figura 3 apresenta uma visão geral da organização de um sistema autoadaptativo, conforme proposto por Weyns. Essa estrutura é composta por três elementos principais: o sistema gerenciado, o sistema de gerenciamento e as metas de adaptação, além da interação contínua com o ambiente.

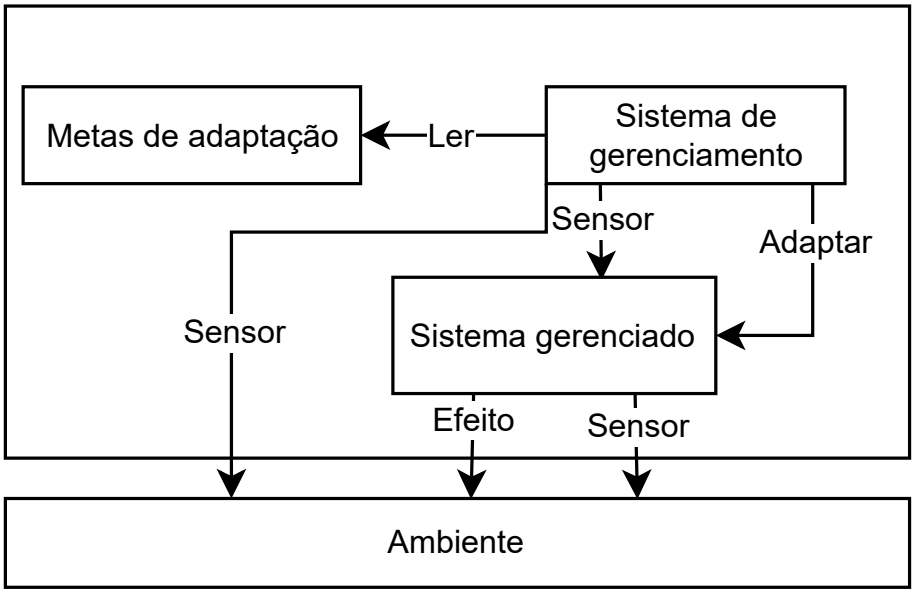

Figura 3 – Visão geral de um sistema adaptativo

O *Sistema gerenciado* corresponde ao componente funcional que realiza as tarefas principais da aplicação, sendo o alvo das adaptações. O *Sistema de gerenciamento* monitora tanto o *Ambiente* quanto o sistema gerenciado, avalia seu estado em relação aos objetivos estabelecidos e executa adaptações conforme necessário. Esse componente encapsula o ciclo MAPE-K anteriormente descrito. As *Metas de adaptação* especificam os critérios de desempenho a serem mantidos durante a operação do sistema, orientando o processo de tomada de decisão. *Sensores* e atuadores são res-

ponsáveis por captar informações contextuais e aplicar mudanças no comportamento do sistema, respectivamente.

Na prática, sistemas adaptativos são empregados em diversos domínios, como arquiteturas em nuvem, sistemas embarcados e aplicações móveis. Em todos esses contextos, a capacidade de observar o ambiente, tomar decisões baseadas em dados contextuais e agir de forma autônoma é essencial para garantir resiliência e eficiência operacional.

Um desafio nesse tipo de arquitetura reside no equilíbrio entre adaptabilidade e previsibilidade. Sistemas excessivamente adaptativos podem se tornar difíceis de validar e suscetíveis a comportamentos não determinísticos ou inconsistentes com seus objetivos. Por isso, existe um esforço contínuo na pesquisa de técnicas que garantam propriedades como estabilidade, segurança e desempenho, mesmo diante de adaptações dinâmicas.

2.3 Teoria do Controle

A Teoria do Controle (JOHAN, 2021) é uma subárea da engenharia voltada ao estudo e desenvolvimento de técnicas para regular o comportamento de sistemas dinâmicos, isto é, sistemas cuja saída evolui ao longo do tempo em resposta a entradas internas e externas. Seu objetivo principal é garantir que o sistema atinja e mantenha um estado desejado, denominado *setpoint*, mesmo na presença de perturbações e incertezas. A Figura 4 ilustra a estrutura básica de um sistema de controle baseado em *feedback*.

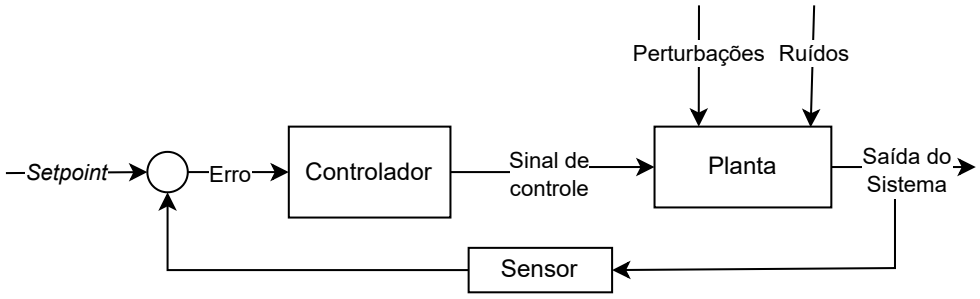

Figura 4 – Visão geral de um *feedback control loop*

Um sistema de controle típico é composto por quatro elementos essenciais: o *setpoint* ou referência, o *Controlador*, a *Planta*, que está sujeita a *Perturbações* e *Ruídos*, e um *Sensor* para

monitoramento (JANERT, 2013). O *setpoint* representa o valor de referência ou objetivo desejado. O *Controlador* calcula e aplica ações corretivas com base no *Erro*, que corresponde à diferença entre o *setpoint* e a *Saída do Sistema*. Esse processo ocorre de forma contínua e iterativa: o controlador ajusta a ação aplicada, buscando minimizar o *Erro* e garantir tanto a estabilidade quanto o desempenho do sistema, mesmo diante de perturbações externas ou variações internas, que podem surgir a partir dos ruídos e perturbações que eventualmente acontecem no *Sistema controlado*.

2.3.1 Erro e Ações Corretivas

O conceito de erro é fundamental para a operação de sistemas de controle. Ele é definido como a diferença entre o *setpoint* (r) e a saída atual do sistema (y).

A análise do erro fornece informações não apenas sobre a magnitude da discrepância, mas também sobre sua direção (SHEVTSOV et al., 2018). Quando o erro é positivo, indica que a saída está abaixo do valor desejado, e o sistema deve agir para aumentá-la; quando é negativo, significa que a saída está acima do *setpoint*, e a ação de controle deve reduzi-la. A magnitude do erro também influencia a intensidade da ação corretiva aplicada.

A resposta a um erro nem sempre é imediata ou proporcional de maneira linear, uma vez que ações corretivas muito bruscas podem gerar instabilidades, como oscilações ou sobrecorreções. Por isso, diferentes estratégias de controle são utilizadas para balancear rapidez, estabilidade e precisão na convergência ao *setpoint*.

2.3.2 *Lags* e *Delays*

Sistemas adaptativos, embora poderosos, possuem algumas limitações naturais devido à necessidade de ajustes dinâmicos e contínuos. Uma das principais restrições é a tendência à oscilação, que pode comprometer tanto a estabilidade quanto o desempenho. Essas oscilações geralmente têm origem em atrasos e características internas dos componentes do sistema. Em sistemas adaptativos, a resposta à entrada raramente é instantânea (JANERT, 2013). A defasagem temporal entre a detecção de um erro e a aplicação da ação corretiva pode comprometer a estabilidade do sistema, especialmente em ambientes altamente dinâmicos.

Os *lags* referem-se a atrasos causados pela dinâmica interna do sistema. Esses atrasos podem

derivar de limitações físicas, como o tempo necessário para que sensores captem informações, ou para que atuadores implementem ajustes no processo. Além disso, atrasos computacionais, como o tempo de processamento necessário para calcular ações corretivas complexas, também contribuem para os *lags*.

Já os *delays* representam atrasos previsíveis e, muitas vezes, inevitáveis entre uma entrada e sua resposta correspondente. Um exemplo clássico são atrasos de transmissão em redes de comunicação, onde os dados levam um tempo fixo para percorrer uma distância física. Em alguns casos, *delays* são deliberadamente introduzidos como estratégia de controle para suavizar respostas muito rápidas e evitar instabilidades.

A correta modelagem e compensação de *lags* e *delays* são fatores levados em consideração para o projeto de controladores, sendo necessário, muitas vezes, o uso de técnicas específicas como predição de estados futuros ou filtros para melhorar a resposta do sistema.

2.3.3 Ruídos e Interferências

Outros conceitos importantes na Teoria de Controle são os de ruídos e interferências (JOHAN, 2021). Devido à sua interação constante com o ambiente, esses sistemas estão sujeitos a variações imprevisíveis e indesejadas em suas entradas. Ruídos podem ser definidos como pequenas flutuações aleatórias nas entradas ou saídas do sistema, geralmente provocadas por fenômenos computacionais, como variações no *clock* do processador, falhas de leitura de sensores digitais, *jitter* em redes de comunicação ou erros de amostragem em sistemas de monitoramento.

Embora muitas vezes de baixa intensidade, o acúmulo de ruído pode prejudicar a precisão do controle, introduzindo erros sutis que degradam o desempenho ao longo do tempo. Interferências, por outro lado, são perturbações mais significativas, podendo ser de origem externa (por exemplo, tráfego de rede concorrente afetando a latência de comunicação) ou interna (como *crosstalk* entre circuitos adjacentes). Elas não apenas adicionam ruído, mas podem distorcer a natureza da informação transmitida, levando a interpretações incorretas pelo controlador e resultando em ações corretivas inadequadas.

Técnicas como filtragem de sinais, uso de sensores redundantes, robustez de projeto e estratégias de controle adaptativo são empregadas para mitigar os efeitos de ruídos e interferências, aumentando a confiabilidade e a estabilidade dos sistemas adaptativos.

2.3.4 Controladores

Controladores são mecanismos projetados para manipular as entradas de um sistema de modo a produzir uma saída desejada (JOHAN, 2021; SHEVTSOV et al., 2018). Em termos gerais, o controlador tem como entrada a diferença entre o *setpoint* e o valor de saída do sistema, e gera um sinal de controle, que é aplicado ao sistema aproximá-lo do *setpoint* (JANERT, 2013).

Existe uma grande diversidade de controladores, tais como: *On/Off*, o controlador PID (*Proporcional-Integral-Derivativo*), o controlador LQR (*Linear Quadratic Regulator*) (JHA; BHASIN, 2020), o controlador MPC (*Model Predictive Control*) (SCHWENZER et al., 2021), o controlador *AsTAR* (*Adaptive Scaling by Threshold-based Auto-Regulation*) (YANG et al., 2021) e o HPA (*Horizontal Pod Autoscaler*) do Kubernetes (Kubernetes, 2025).

2.4 Controladores Fuzzy

Uma característica comum entre diversos controladores é a suposição de uma relação predominantemente linear entre as variáveis de entrada (por exemplo, o erro) e as variáveis de saída (a ação de controle). Contudo, essa suposição de linearidade pode se tornar uma limitação em sistemas dinâmicos complexos e distribuídos, onde a relação entre variáveis é frequentemente não linear, incerta ou sujeita a ruído.

Nesse contexto, existem os controladores baseados em lógica fuzzy. A lógica fuzzy (ZADEH, 1965) introduz a capacidade de modelar sistemas que operam sob informações incertas, imprecisas ou qualitativas. Em vez de assumir que as variáveis pertencem exclusivamente a estados binários (verdadeiro ou falso), a lógica fuzzy permite graus de pertinência a diferentes conjuntos, possibilitando decisões mais flexíveis em ambientes complexos.

A Figura 5 mostra os principais elementos de um controlador fuzzy: Fuzzificação, o Mecanismo de Inferência, a Defuzzificação e a Base de Conhecimento. (BAI; WANG, 2006; DRIANKOV; HELLENDOORN; REINFRANK, 1993).

A *Fuzzificação* é a etapa inicial do processo de controle fuzzy, na qual os valores numéricos das variáveis de entrada são convertidos em representações linguísticas imprecisas, chamadas de valores fuzzy. Essa conversão é feita por meio de funções de pertinência, que associam cada valor a diferentes conjuntos linguísticos, como *Baixo*, *Médio* ou *Alto*. Entre as funções mais utilizadas

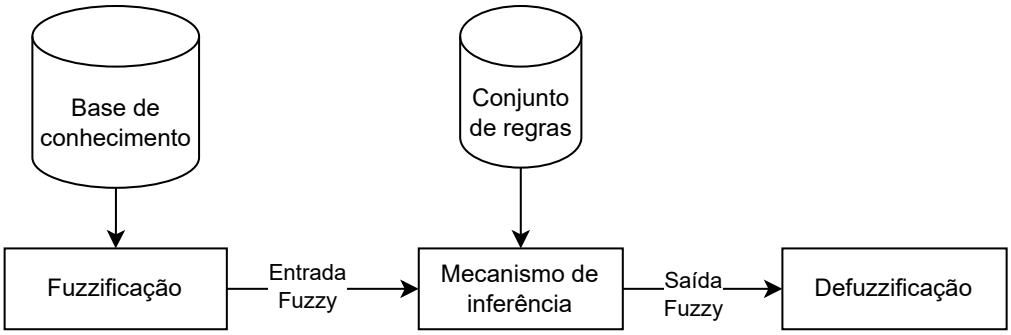

Figura 5 – Arquitetura de um controlador fuzzy

para esse processo estão as de forma triangular, trapezoidal e gaussiana (BAI; WANG, 2006). A definição dos conjuntos e das respectivas funções depende da *Base de conhecimento*, que especifica os domínios fuzzy e suas características.

Depois da *Fuzzificação*, os valores fuzzy passam pelo *Mecanismo de inferência*, que aplica as regras armazenadas no *Conjunto de regras*. Essas regras seguem, em geral, a estrutura lógica do tipo *Se... Então*, permitindo que o sistema relacione diferentes situações de entrada com ações de controle apropriadas. Um exemplo de regra seria: Se o erro é pequeno e a variação é negativa, então a ação é moderada. Cada regra avalia o quanto suas condições são atendidas e os resultados das regras ativadas são combinados em uma *Saída Fuzzy*.

A etapa seguinte é a *Defuzzificação*, que transforma a *Saída Fuzzy* em um valor numérico que pode ser utilizado diretamente no sistema controlado. Como a *Saída Fuzzy* representa diferentes conjuntos com graus variados de pertinência, a *Defuzzificação* aplica critérios específicos para obter um único valor que reflita de forma adequada a decisão do controlador.

2.4.1 Fuzzificação e Funções de Pertinência

A fuzzificação depende de funções de pertinência (*membership function*s), que determinam o grau de pertencimento de uma entrada a um conjunto fuzzy específico. Uma função de pertinência $\mu(x)$ pode ser descrita genericamente como:

$$\mu(x) : X \rightarrow [0,1], \tag{2.1}$$

onde X é o conjunto universo de discurso da variável de entrada, e $[0,1]$ representa o intervalo

de pertinência, no qual 0 indica ausência total de pertencimento e 1 representa pertencimento completo. O formato adotado para a função de pertinência influencia diretamente a sensibilidade do sistema às variações da entrada, afetando a suavidade da resposta do controlador e, consequentemente, sua estabilidade e robustez (BAI; WANG, 2006).

As seções seguintes apresentam algumas das mais utilizadas funções de pertinência.

2.4.1.1 Função Triangular

A função triangular é uma das mais simples e amplamente utilizadas, especialmente em implementações com restrições de recursos computacionais (BAI; WANG, 2006). Ela é definida por três parâmetros (a,b,c), que determinam o início, o pico e o fim do conjunto fuzzy, respectivamente. A forma da função é linear em ambos os lados do ponto central, conforme apresentado na Equação 2.2.

$$\mu(x) = \begin{cases} 0, & x \leq a \ \frac{x-a}{b-a}, \\ a \leq x \leq b \ \frac{c-x}{c-b}, & b \leq x \leq c \ 0, \\ x \geq c \end{cases} \tag{2.2}$$

Sua simplicidade torna a triangular uma escolha adequada para aplicações que exigem respostas rápidas e diretas, embora, por não possuir transições suaves, seja menos apropriada em cenários com alta variabilidade.

2.4.1.2 Função Gaussiana

A função gaussiana, por sua vez, proporciona transições suaves e contínuas entre os conjuntos fuzzy. Ela é definida por dois parâmetros, conforme a Equação 2.3. σ controla a largura da curva gaussiana e c define o centro da curva. Valores menores de σ produzem uma curva em sino mais estreita, levando a transições mais nítidas entre valores baixos e altos.

$$f(x,\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}, \tag{2.3}$$

Essa função é especialmente útil em sistemas que exigem estabilidade e fluidez na resposta, reduzindo a possibilidade de oscilações abruptas decorrentes de pequenas variações na entrada.

2.4.1.3 Função PI-Shaped

A função em forma de PI combina características de funções trapezoidais e sigmoides, apresentando uma região central plana (onde o grau de pertinência é máximo) e transições suaves nas bordas. Essa configuração permite representar zonas de estabilidade ou tolerância com precisão, sendo útil em aplicações onde se deseja uma resposta estável dentro de certos intervalos. Ela é definida pela Equação 2.4.

$$f(x;a,b,c,d) = \begin{cases} 0, & \text{if } x \leq a \\ 2\left(\frac{x-a}{b-a}\right)^2, & \text{if } a < x \leq \frac{a+b}{2} \\ 1-2\left(\frac{x-b}{b-a}\right)^2, & \text{if } \frac{a+b}{2} < x \leq b \\ 1, & \text{if } b < x \leq c \\ 1-2\left(\frac{x-c}{d-c}\right)^2, & \text{if } c < x \leq \frac{c+d}{2} \\ 2\left(\frac{x-d}{d-c}\right)^2, & \text{if } \frac{c+d}{2} < x \leq d \\ 0, & \text{if } x \geq d, \end{cases} \tag{2.4}$$

onde a e b definem as transições de declive à esquerda de 0 a 1. a determina onde a função começa a aumentar a partir de 0, e b controla a rapidez com que ela atinge 1. c e d definem as transições de declive em um intervalo de 1 a 0.

2.4.1.4 Função G-Bell

A função de *G-Bell* oferece flexibilidade de modelagem, pois seus parâmetros a, b e c (conforme definido na Equação 2.5) permitem ajustar a largura, a forma (inclinação) e o centro da curva. Devido à sua versatilidade, é amplamente empregada em sistemas fuzzy mais complexos ou com requisitos específicos de controle.

$$f(x;a,b,c) = \frac{1}{1+\left(\frac{x-c}{a}\right)^{2b}}, \tag{2.5}$$

Onde a representa a largura do sino, quanto maior o a, mais largo o sino. b controla a inclinação no ponto de cruzamento, e c é onde o valor de pertinência é 1, o centro do sino.

2.4.1.5 Função Sigmoidal

A função sigmoidal é caracterizada por uma transição assimétrica e suave entre os extremos, conforme definida na Equação 2.6.

$$\mu(x) = \frac{1}{1+e^{-\alpha(x-c)}} \tag{2.6}$$

O parâmetro α define a inclinação da curva (a taxa de transição), enquanto c determina o ponto de inflexão. Essa função é comumente usada em sistemas fuzzy adaptativos e redes neurais, por sua capacidade de representar crescimento ou decaimento de maneira gradual e contínua.

2.4.1.6 Função Z-shaped

A função *Z-shaped* é utilizada para representar uma transição suave de pertencimento alto para pertencimento baixo, sendo adequada para modelar regiões de decrescimento gradual em variáveis fuzzy. Sua forma é assintótica e suavemente decrescente, conforme ilustrado na Equação 2.7.

$$f(x;a,b) = \begin{cases} 1, & x \leq a \\ 1-2\left(\frac{x-a}{b-a}\right)^2, & a < x \leq \frac{a+b}{2} \\ 2\left(\frac{b-x}{b-a}\right)^2, & \frac{a+b}{2} < x < b \\ 0, & x \geq b \end{cases} \tag{2.7}$$

Onde os parâmetros a e b definem os limites inferior e superior da zona de transição. O ponto médio $\frac{a+b}{2}$ marca o centro da transição entre os estados de alta e baixa pertinência. Essa função apresenta continuidade e suavidade em toda sua extensão, o que a torna adequada para sistemas que demandam transições suaves e não abruptas no processo de fuzzificação.

2.4.2 Defuzzificação

Após a fuzzificação e a aplicação das regras por meio do mecanismo de inferência, a saída do sistema ainda permanece em forma fuzzy, ou seja, representada por conjuntos com graus de pertinência distribuídos ao longo do universo de discurso. Para que essa saída possa ser efetivamente utilizada pelo sistema controlado, que geralmente opera com valores numéricos, é necessário convertê-la em um valor não-fuzzy por meio do processo conhecido como defuzzificação.

Existem diversos métodos de defuzzificação, cuja escolha deve considerar o tipo de resposta desejada: mais estável e refinada, mais rápida e simples, ou um equilíbrio entre essas características (PROPERTIES..., 2010).

Um dos métodos mais utilizados é o Centroide (também chamado de *Center of Gravity* ou *Centroid Method* - CoG), que calcula o centro de massa da área sob a curva de pertinência agregada, conforme a Equação 2.8:

$$z = \frac{\int x\mu(x), dx}{\int \mu(x), dx} \tag{2.8}$$

Esse método fornece uma saída representativa e equilibrada entre os diversos conjuntos fuzzy ativados, sendo particularmente eficaz para manter a estabilidade do controle. Nessa equação, $\mu(x)$ representa o grau de pertinência do valor x ao conjunto fuzzy resultante da inferência, enquanto z corresponde ao valor defuzzificado aplicado ao sistema. A integral no numerador calcula o momento da área em relação à origem, e a integral no denominador determina a área total sob a curva fuzzy.

Outro método de deffuzificação comum é a Média dos Máximos (*Mean of Maximum* - MoM), que calcula a média dos valores de entrada cujo grau de pertinência atinge o valor máximo, conforme a Equação 2.9:

$$z = \frac{\sum_{x \in x | \mu(x) = \mu_{\max}} x}{n} \tag{2.9}$$

Onde, $\mu_{\max}$ é o valor máximo da função de pertinência $\mu(x)$, x representa os pontos onde esse valor máximo ocorre, e n é a quantidade desses pontos. Esse método é especialmente útil quando se deseja uma resposta central entre os valores mais ativados, sendo adequado para distribuições simétricas.

Em contextos onde se busca priorizar decisões mais rápidas ou assertivas, pode-se empregar o método Maior dos Máximos (*Largest of Maximum* - LoM), descrito na Equação 2.10:

$$z = \max x \mid \mu(x) = \mu_{\max} \tag{2.10}$$

Neste método, z corresponde ao maior valor de entrada cujo grau de pertinência é máximo. Essa abordagem é vantajosa em sistemas que requerem respostas mais incisivas ou operam com margens de segurança mais amplas.

Uma alternativa que busca um ponto de equilíbrio, mesmo em distribuições assimétricas, é o método da Bissetriz da Área (*Bisector of Area* - BSC), que determina o valor z capaz de dividir a área total sob a curva de pertinência em duas partes iguais, conforme a Equação 2.11:

$$\int_{x_{\min}}^{z} \mu(x), dx = \int_{z}^{x_{\max}} \mu(x), dx \tag{2.11}$$

Onde $x_{\min}$ e $x_{\max}$ delimitam o intervalo do universo de discurso da variável de saída, e z representa o ponto em que a área sob a curva fuzzy é igualmente dividida. O método da bissetriz é eficaz para evitar vieses em sistemas com distribuições não uniformes.

Os métodos de defuzzificação apresentados foram selecionados por oferecerem diferentes perfis de resposta, considerando variáveis como a complexidade computacional e o comportamento esperado em diferentes condições operacionais. Métodos como o Centroide e o *Bisector* tendem a gerar respostas mais refinadas e estáveis, enquanto abordagens como o LoM são mais simples e adequadas para sistemas que exigem maior reatividade ou operam sob restrições de tempo (PROPERTIES…, 2010).

2.5 Algoritmos Genéticos

Algoritmos evolucionários constituem uma classe de métodos de busca e otimização inspirados em processos naturais, com base na teoria da evolução. Entre as principais meta-heurísticas dessa área, destaca-se o Algoritmo Genético (AG), que simula o mecanismo de seleção natural para explorar o espaço de soluções de problemas complexos, geralmente caracterizados por múltiplos ótimos locais e alta dimensionalidade (ZURADA; MARKS; ROBINSON, 1994; PETROWSKI; BEN-HAMIDA; MICHALEWICZ, 2017). O funcionamento de um AG baseia-se na evolução de uma

população de soluções candidatas ao longo de múltiplas iterações, denominada geração, com o objetivo de aproximar-se da solução global ótima ou de uma solução suficientemente satisfatória para o problema considerado.

Cada indivíduo representa uma possível solução codificada, usualmente na forma de um vetor de parâmetros, frequentemente chamado de cromossomo. As posições desse vetor são conhecidas como genes e armazenam valores que descrevem características específicas da solução. Esses genes podem ser modificados ao longo da execução do algoritmo, promovendo a exploração do espaço de busca. O processo evolutivo é conduzido por três operadores fundamentais: seleção, recombinação e mutação. A Figura 6 apresenta o fluxo de um algoritmo genético (LAMBORA; GUPTA; CHOPRA, 2019; ZURADA; MARKS; ROBINSON, 1994).

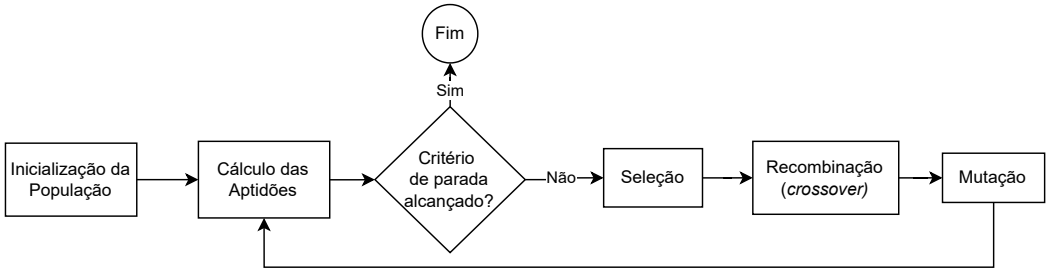

Figura 6 – Arquitetura geral de um algoritmo genético

Inicialmente, a *população* pode ser gerada de maneira totalmente aleatória ou a partir de um conjunto pré-definido de soluções (*Inicialização da População*), eventualmente construído com base em heurísticas ou em conhecimento prévio. Populações iniciais diversificadas favorecem a exploração, enquanto populações especializadas tendem a acelerar a convergência. No entanto, este último cenário aumenta o risco do algoritmo ficar preso em soluções subótimas, ou seja, regiões do espaço de busca que apresentam bom desempenho relativo, mas que estão distantes da solução globalmente ótima.

Cada geração é avaliada com base em uma função de aptidão (*Cálculo das Aptidões*), também chamada de *fitness*, que atribui a cada indivíduo um valor que reflete o quão bem ele resolve o problema. Essa função é responsável por orientar o processo evolutivo, permitindo que as melhores soluções influenciem mais fortemente as gerações subsequentes.

Após a avaliação de cada uma das soluções candidatas, é verificado se o critério de parada já foi alcançado. Essa verificação normalmente é feita sempre que o *fitness* é calculado para cada

indivíduo.

Caso o critério de parada não seja atendido, o fluxo segue para a *Seleção*. O operador de *Seleção* é responsável por definir quais indivíduos da população atual serão escolhidos para participar da geração de descendentes. Esse processo influencia diretamente a pressão seletiva do algoritmo, ou seja, o grau com que as características dos indivíduos mais aptos passam pelas próximas gerações. Entre os métodos mais utilizados, destaca-se a Seleção por Roleta, em que a probabilidade de um indivíduo ser selecionado é proporcional ao seu valor de aptidão. Com isso, soluções de melhor desempenho têm mais chances de contribuir geneticamente para a próxima geração, mas indivíduos menos aptos ainda podem ser escolhidos, o que ajuda a manter a diversidade. Esse método é especialmente útil em problemas com muitas soluções próximas em qualidade, pois preserva certa aleatoriedade na seleção (LAMBORA; GUPTA; CHOPRA, 2019).

Outro método amplamente adotado é a Seleção por Torneio, no qual pequenos grupos de indivíduos são formados aleatoriamente e, dentro de cada grupo, o indivíduo com melhor desempenho é escolhido como pai. A intensidade da pressão seletiva pode ser ajustada pelo tamanho dos torneios: grupos maiores tendem a favorecer indivíduos mais aptos, enquanto grupos menores mantêm maior variabilidade. Essa abordagem é simples de implementar, paralelizável e robusta mesmo em ambientes com ruído na avaliação da aptidão.

Além desses métodos, existe também a Seleção Aleatória, em que todos os indivíduos da população têm chances iguais de serem escolhidos, independentemente de sua aptidão. Embora raramente usada isoladamente em problemas de otimização, essa técnica pode ser útil como componente auxiliar em abordagens híbridas, pois contribui para preservar a diversidade genética e evitar a convergência prematura a soluções subótimas.

Os indivíduos selecionados são combinados por meio do operador de *Recombinação*, também conhecido como *crossover*, cuja finalidade é explorar novas regiões do espaço de busca ao mesclar características dos pais. Entre as estratégias mais utilizadas estão a recombinação de ponto único (*one-point crossover*), de dois pontos (*two-point crossover*) e a recombinação uniforme (*uniform crossover*). A taxa de recombinação influencia diretamente o grau de diversidade estrutural introduzida na nova geração (LAMBORA; GUPTA; CHOPRA, 2019).

A recombinação de ponto único consiste em uma técnica na qual um ponto de corte é escolhido aleatoriamente ao longo do cromossomo dos pais. A partir desse ponto, os segmentos finais dos indivíduos são trocados, gerando dois descendentes com combinações distintas das informações

originais. Esse método é simples e eficiente, sendo especialmente útil quando a posição dos genes possui correlação com a qualidade da solução.

A recombinação de dois pontos é uma técnica semelhante à de ponto único, a diferença está em que dois pontos de corte são selecionados aleatoriamente, delimitando uma região intermediária no cromossomo. Essa região é trocada entre os indivíduos, enquanto os segmentos externos permanecem inalterados. Essa técnica oferece maior flexibilidade na recombinação e tende a preservar melhor blocos de genes relevantes, favorecendo a intensificação da busca em regiões promissoras do espaço de soluções.

Já a recombinação uniforme opera de maneira diferente: para cada posição do cromossomo, decide-se aleatoriamente, com uma probabilidade fixa, qual dos dois pais fornecerá o gene correspondente ao descendente. Essa estratégia rompe totalmente a estrutura sequencial dos cromossomos e promove uma mistura mais granular das características parentais. Embora aumente a diversidade genética, ela pode dificultar a preservação de combinações favoráveis de genes adjacentes, exigindo calibração mais cuidadosa nos demais parâmetros do algoritmo.

A taxa de recombinação define a proporção da população que será submetida ao operador de *crossover* a cada geração, influenciando diretamente o grau de diversidade estrutural introduzida. Taxas elevadas promovem maior exploração, enquanto taxas mais conservadoras favorecem a estabilidade e a preservação de soluções já boas.

Após a recombinação, o operador de *Mutação* atua introduzindo pequenas variações aleatórias nos genes de alguns indivíduos (ZURADA; MARKS; ROBINSON, 1994), promovendo a diversidade genética e evitando a estagnação prematura do algoritmo. A mutação é utilizada para escapar de ótimos locais e ampliar a capacidade de busca global. Sua taxa deve ser cuidadosamente ajustada: valores muito altos podem comprometer a convergência ao dispersar demais a população, enquanto valores muito baixos reduzem a variabilidade, dificultando a superação de barreiras no espaço de busca.

Após a *Mutação*, aplica-se o processo de substituição, que define quais indivíduos farão parte da nova população. Estratégias elitistas garantem que os melhores indivíduos da geração anterior sejam preservados, enquanto abordagens não elitistas podem substituir parte ou toda a população com base em sorte, aptidão relativa ou outras heurísticas.

O ciclo evolutivo prossegue até que um *Critério de Parada seja Alcançado*. Esse critério pode ser o número máximo de gerações, a ausência de melhora na função de aptidão após certo número

de iterações ou a obtenção de uma solução com desempenho considerado satisfatório.

Algoritmos genéticos são amplamente aplicados em problemas nos quais métodos exatos são inviáveis devido ao elevado custo computacional, como em otimização combinatória, ajuste de parâmetros em sistemas complexos, aprendizado de máquina e engenharia evolutiva (LINGARAJ, 2016; MIRJALILI, 2019).

2.6 Considerações finais

Neste capítulo, foram apresentados os principais conceitos teóricos que foram utilizados na proposta desta dissertação. Iniciou-se com o contexto dos serviços de mensageria, apontando a necessidade de mecanismos de controle adaptativo. Também foram apresentados conceitos da Teoria do Controle, com foco na lógica fuzzy, destacando seu papel na modelagem de incertezas e na construção de controladores mais flexíveis. Em seguida, explorou-se o processo de fuzzificação, o mecanismo de inferência fuzzy e os métodos de defuzzificação, apontando suas vantagens e implicações no comportamento do sistema controlado. Por fim, foram apresentados os conceitos relacionados aos algoritmos genéticos, enfatizando sua capacidade de explorar espaços de busca complexos e gerar soluções boas para problemas desconhecidos ou inexplorados suficientes.

3 FALCOM

Este capítulo apresenta o FALCOM (***F**ramework **A**daptativo baseado na **L**ógica Fuzzy para **C**ontrole de Middleware **O**rientado a **M**ensagem*) proposto para adaptação de serviços de mensageria. Ele apresenta inicialmente uma visão geral da solução na Seção 3.1. Em seguida, na Seção 3.2, é detalhado o conjunto de controladores que compõem o repositório do framework. Então, na Seção 3.3 é apresentado como o algoritmo genético foi usado no mecanismo de inferência dos controladores. Na Seção 3.4 são apresentados detalhes da implementação, seguida da descrição de como usar o framework (Seção 3.5).

3.1 Visão Geral

FALCOM é um framework de controle adaptativo capaz de regular dinamicamente o fluxo de mensagens em MOMs, por meio de controladores fuzzy (ALMEIDA; ROSA, 2025). A abordagem permite que os consumidores ajustem seu comportamento de forma autônoma, respondendo às variações da carga de trabalho e preservando a estabilidade e o desempenho do sistema. A Figura 7 ilustra a arquitetura geral da solução, destacando o processo de integração dos controladores ao consumidor.

O fluxo operacional inicia-se com o desenvolvedor, que define os parâmetros necessários para a configuração do sistema no *Orquestrador de Execução*. A partir dessas informações, o orquestrador instancia um controlador fuzzy, selecionado a partir de um *Repositório de Controladores* que agrupa as diferentes variações implementadas. Esse repositório contém um conjunto de controladores fuzzy pré-configurados, cada um com características específicas de fuzzificação, defuzzificação e regras de controle.

O *Orquestrador de Execução* não apenas seleciona e instancia o controlador, mas também é responsável pela geração e ajuste das regras de controle durante a execução. Esse processo adaptativo é viabilizado pela integração de um *Algoritmo Genético*, que avalia continuamente o desempenho do controlador e promove ajustes automáticos nas regras, garantindo assim uma resposta mais eficiente às variações no ambiente.

Uma vez configurado, o controlador fuzzy (*Fuzzy N*) é acoplado ao *Consumidor*, que passa a ter

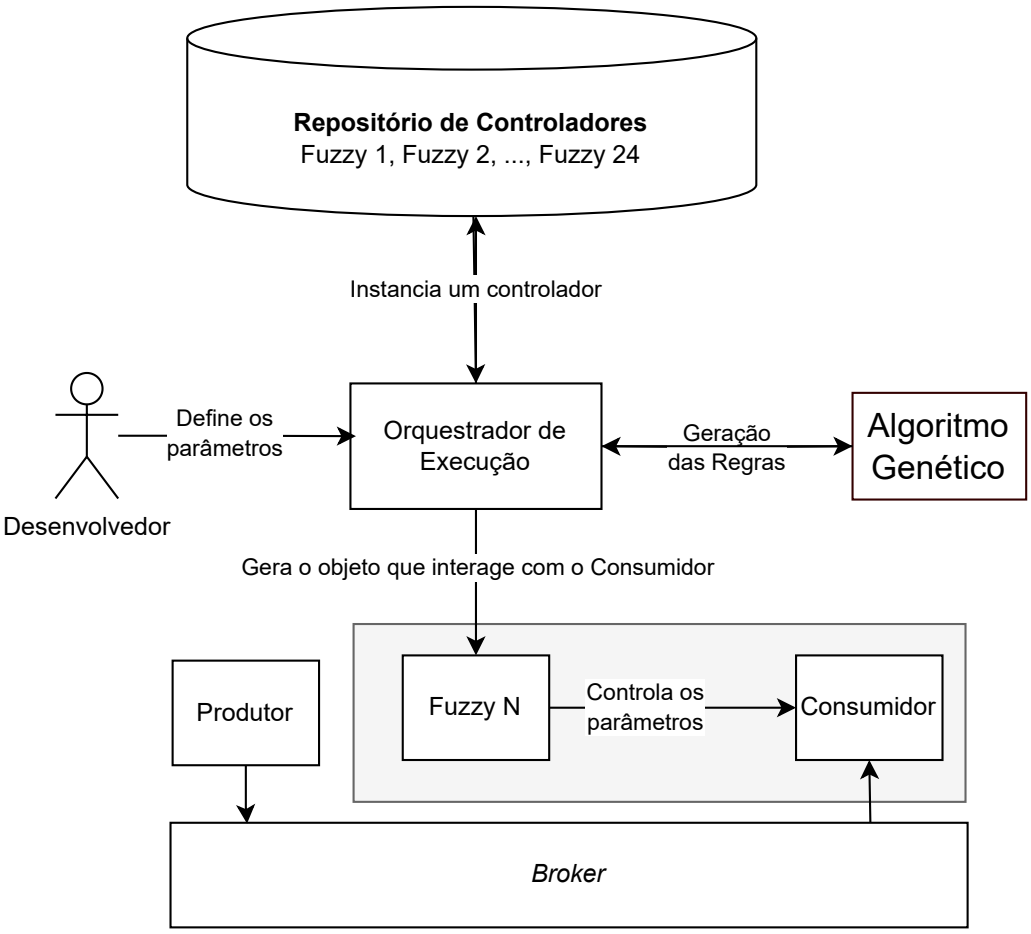

Figura 7 – Visão geral da integração dos controladores com o Consumidor

os parâmetros controlados por ele. O controlador atua ajustando dinamicamente os parâmetros de consumo de mensagens, de modo a regular o fluxo de mensagens que chega ao *Consumidor*. Essa interação ocorre nos parâmetros do *Consumidor* que acabam interferindo na ação do *Broker*. Por fim, o *Produtor* é responsável por enviar as mensagens, enquanto o *Broker* encaminha os dados para o *Consumidor*, que os consome de acordo com as restrições impostas pelo controlador selecionado.

3.2 Repositório de Controladores

Como apresentado no Capítulo 2, um controlador fuzzy é composto por quatro elementos fundamentais: fuzzificação, base de regras, mecanismo de inferência e defuzzificação. A construção dos controladores desenvolvidos neste trabalho seguiu as seguintes etapas: (i) definição dos con-

juntos linguísticos; (ii) escolha dos métodos de fuzzificação; (iii) escolha dos métodos de defuzzificação; e (iv) definição das regras.

3.2.1 Fuzzificação

A definição dos conjuntos fuzzy requer a identificação de variáveis linguísticas de entrada e saída, bem como suas respectivas categorias. A Tabela 2 apresenta as categorias utilizadas neste trabalho, organizadas em ordem decrescente de influência. A definição das categorias linguísticas seguiu uma abordagem tradicional em sistemas de controle fuzzy, na qual se busca representar de forma qualitativa os diferentes níveis de variação das variáveis de entrada e saída. A escolha das sete categorias, compostas por três negativas, uma neutra e três positivas, permitiu uma representação equilibrada e suficientemente granular do comportamento do sistema, capturando desde desvios pequenos até variações mais acentuadas.

Tabela 2 – Variáveis linguísticas dos controladores fuzzy

Erro (variável E)	**Ajuste da saída (variável AS)**
PG: Positivo grande	RA: Redução alta
PM: Positivo médio	RM: Redução média
PP: Positivo pequeno	RP: Redução pequena
ZE: Zero	Z: Nenhum ajuste
NP: Negativo pequeno	AP: Aumento pequeno
NM: Negativo médio	AM: Aumento médio
NG: Negativo grande	AA: Aumento alto

A variável de entrada *E* representa o erro, definido como a diferença entre o valor de referência (*setpoint*) e o valor medido. A polaridade do erro (*Positivo* ou *Negativo*) indica a direção do desvio, enquanto sua magnitude (*grande*, *médio*, *pequeno* ou *zero*) representa a intensidade do desvio. A variável de saída *AS* refere-se ao ajuste aplicado no parâmetro de controle, podendo indicar aumentos ou reduções com diferentes intensidades.

Definidas as variáveis linguísticas, a etapa de fuzzificação é responsável por transformar valores numéricos destas variáveis em graus de pertinência aos conjuntos fuzzy, permitindo que o sistema opere com informações imprecisas ou ambíguas. Neste trabalho, foram avaliadas e implementadas sete funções de pertinência com diferentes características: Gaussiana (Equação 2.3), Triangular

(Equação 2.2), *G-Bell* (Equação 2.5), Sigmoidal (Equação 2.6), *PI-shaped* (Equação 2.4), *Z-shaped* (Equação 2.7). A escolha dessas funções teve como objetivo proporcionar uma diversidade de formas de fuzzificação, abrangendo desde transições suaves e simétricas até comportamentos assimétricos e abruptos.

As funções do tipo Gaussiana e *G-Bell* foram escolhidas por oferecerem transições suaves entre os conjuntos fuzzy, sendo adequadas para situações em que se deseja evitar oscilações abruptas nas decisões do controlador. Já a função Triangular foi escolhida por ser amplamente utilizada em sistemas fuzzy devido a sua simplicidade computacional. Além disso, ela permite uma representação eficiente de transições lineares, sendo recomendada quando se busca equilíbrio entre desempenho e custo computacional.

A função Sigmoidal foi escolhida porque é útil em contextos onde a transição entre regiões é assimétrica ou não linear, permitindo maior flexibilidade na modelagem do comportamento do sistema. A função *PI-shaped* foi escolhida por apresentar um perfil suave com regiões planas nas extremidades, o que a torna adequada para aplicações que requerem uma zona de estabilidade em torno de valores extremos. A função *Z-shaped*, por sua vez, foi escolhida por representar comportamentos decrescentes com saturação, típica de situações onde há uma resposta negativa progressiva a determinados estímulos.

3.2.2 Defuzzificação

A defuzzificação é a etapa final do processo de inferência fuzzy, responsável por converter os graus de pertinência resultantes da inferência em valores numéricos concretos, compatíveis com os sistemas convencionais de controle. Diversos métodos foram considerados neste trabalho: Centroide ou Centro de Gravidade (CoG - *Center of Gravity*) (Equação 2.8), Média dos Máximos (MoM - *Mean of Maximum*) (Equação 2.9), Máxima Adesão (LoM - *Largest of Maximum*) (Equação 2.10), também conhecido como Método da Altura ou Maior dos Máximos, Método Bisector (BSC - *Bisector of Area*) (Equação 2.11).

O método CoG foi escolhido por levar em conta toda a distribuição das pertinências ao longo do domínio da variável de saída. Este método tende a gerar respostas mais precisas e equilibradas, especialmente em sistemas que exigem decisões graduais com sensibilidade a múltiplos fatores de entrada.

O método MoM, que considera apenas os pontos com o maior grau de pertinência, foi incluído como alternativa eficiente em termos computacionais. Embora apresente menor precisão, sua simplicidade o torna útil em aplicações com restrições de processamento ou onde a decisão pode tolerar uma margem de erro mais ampla.

O método LoM seleciona o maior valor associado ao grau de pertinência máximo. Esse método é apropriado em sistemas onde se deseja uma resposta direta e determinística, priorizando a categoria de saída mais ativada.

O método BSC busca o ponto que divide a área total sob a curva de pertinência em duas metades iguais. Trata-se de uma alternativa ao CoG que, embora não leve em conta o momento das distribuições, oferece uma resposta mais robusta à presença de assimetrias ou valores extremos.

3.2.3 Conjunto de Controladores

Ao compor o framework com todas as funções de fuzzificação e defuzzificação, além do algoritmo genético, há um total de vinte e quatro tipos diferentes de controladores possíveis, com combinações diferentes de métodos de fuzzificação e defuzzificação. Todas as possibilidades estão descritas na Tabela 3.

Tabela 3 – Configurações dos controladores fuzzy

Controlador	Função de Pertinência	Método de Desfuzzificação
Fuzzy 1	Triangular	CoG
Fuzzy 2	Triangular	MoM
Fuzzy 3	Triangular	LoM
Fuzzy 4	Triangular	BSC
Fuzzy 5	Gaussiana	CoG
Fuzzy 6	Gaussiana	MoM
Fuzzy 7	Gaussiana	LoM
Fuzzy 8	Gaussiana	BSC
Fuzzy 9	*G-Bell*	CoG
Fuzzy 10	*G-Bell*	MoM
Fuzzy 11	*G-Bell*	LoM
Fuzzy 12	*G-Bell*	BSC
Fuzzy 13	Sigmoidal	CoG
Fuzzy 14	Sigmoidal	MoM
Fuzzy 15	Sigmoidal	LoM
Fuzzy 16	Sigmoidal	BSC
Fuzzy 17	*PI-shaped*	CoG
Fuzzy 18	*PI-shaped*	MoM
Fuzzy 19	*PI-shaped*	LoM
Fuzzy 20	*PI-shaped*	BSC
Fuzzy 21	*Z-shaped*	CoG
Fuzzy 22	*Z-shaped*	MoM
Fuzzy 23	*Z-shaped*	LoM
Fuzzy 24	*Z-shaped*	BSC

3.3 Algoritmo Genético

Para a geração e aplicação das regras no controlador fuzzy, foi desenvolvido um algoritmo genético com o intuito de reduzir a necessidade de *expertise* sobre o escopo do problema. Ou seja, evitando o conhecimento que seria exigido de um desenvolvedor para definir manualmente as regras fuzzy adequadas.

Em sistemas fuzzy convencionais, essa tarefa requer que o desenvolvedor compreenda com detalhes as variáveis envolvidas, suas interações e o comportamento desejado do sistema. Por exem-

plo, em um cenário de controle de tráfego em redes, um desenvolvedor teria de estabelecer manualmente regras como *"se a latência é alta e o tráfego é intenso, então reduzir a taxa de envio"*. O algoritmo genético automatiza esse processo ao explorar combinatoriamente diferentes conjuntos de regras, avaliando seu desempenho e selecionando as mais eficazes, o que permite que o sistema seja ajustado mesmo com conhecimento limitado sobre o domínio específico.

3.3.1 Representação dos indivíduos

Cada indivíduo da população é representado por um conjunto de sete regras do tipo *SE ERRO = entrada ENTÃO AJUSTE = saída*, assegurando a cobertura completa de todas as categorias linguísticas da variável de entrada, conforme definido na Tabela 2. A saída de cada regra pode se repetir dentro do conjunto de regras de um mesmo indivíduo, permitindo a existência de múltiplas regras com o mesmo valor de ajuste, o que reflete a flexibilidade necessária para acomodar comportamentos semelhantes em diferentes faixas de erro. Por exemplo, considerando as variáveis da Tabela 2, poderia estabelecer-se a seguinte regra: *SE ERRO = Positivo grande ENTÃO AJUSTE = Redução média*.

A população inicial constituiu-se em um conjunto contendo 10 indivíduos inicializados aleatoriamente. A escolha de iniciar com esse número foi baseada na busca por um equilíbrio entre diversidade e eficiência computacional. Populações menores tendem a convergir mais rapidamente, ao passo que mantêm a viabilidade de gerar soluções diversas o suficiente para evitar estagnação precoce. A inicialização aleatória dos indivíduos assegura a cobertura de diferentes combinações possíveis de regras desde o início, ampliando o espaço amostral inicial da busca evolutiva e aumentando as chances de descoberta de soluções eficazes ao longo das gerações.

3.3.2 Cruzamento e mutação

Para o cruzamento, foi adotado o Método da Roleta (GOLDBERG, 2013), que favorece indivíduos com melhor desempenho sem eliminar completamente a chance de indivíduos menos aptos contribuírem geneticamente. Esta abordagem mantém a pressão seletiva, isto é, faz com que indivíduos com melhor *fitness* tenham mais chances de passarem para a próxima geração, mas preserva a diversidade populacional. O cruzamento é realizado por ponto único, escolhendo-se aleatoriamente

uma posição de divisão entre os componentes dos indivíduos selecionados. Esse método assegura que os descendentes herdem blocos coerentes de regras de ambos os pais, o que tende a preservar estruturas parciais de comportamento eficaz já identificadas ao longo da evolução.

A mutação ocorre com uma probabilidade de 30%. Não há um consenso para a escolha da taxa de mutação, porém taxas para populações menores variam de 5% a 30%, sendo maior para aplicações em que deseja-se aumentar o espaço de busca (HAUPT; HAUPT, 2004). O valor foi escolhido com base em experimentos preliminares que indicaram ser eficaz para manter a variabilidade da população sem comprometer a convergência. O processo de mutação envolve a substituição do valor de saída de uma regra por outro valor aleatório dentre os possíveis. Essa abordagem de mutação direta sobre a saída permite explorar soluções distintas sem alterar a estrutura de entrada do indivíduo, o que mantém a cobertura do domínio de entrada e evita a geração de indivíduos inválidos.

A substituição dos dois piores indivíduos da população pelos descendentes gerados garante a renovação contínua da população e promove o aprimoramento gradual do conjunto de regras. Essa política de substituição elimina soluções consistentemente inferiores e evita que o processo evolutivo fique preso em soluções subótimas.

3.3.3 Avaliação e critério de parada

A função de avaliação (*fitness* - ver Seção 2.5) foi definida com base na capacidade do conjunto de regras em minimizar o erro entre o valor de referência e o valor de saída simulado. O erro é calculado utilizando o *Normalized Root Mean Square Error* (NRMSE), aplicado a diferentes cenários com múltiplos pontos de referência. Esta abordagem amplia a generalização dos indivíduos avaliados, evitando a especialização em casos isolados.

A escolha do NRMSE como métrica se justifica pela sua sensibilidade a desvios maiores e pela possibilidade de normalização, permitindo a comparação entre diferentes escalas de operação. Valores mais baixos de NRMSE indicam um melhor controle, onde o valor monitorado está próximo do valor de referência, sendo, portanto, diretamente utilizado como uma forma de avaliar o desempenho dos indivíduos.

O algoritmo genético encerra a execução caso uma das seguintes condições seja atendida: a) o melhor indivíduo da população atinge um valor de NRMSE inferior a 0,2; ou b) o valor do melhor *fitness* não apresenta melhoria nas últimas 10 gerações. O valor de 0,2 foi escolhido por

ser um limiar considerado baixo, indicando, portanto, uma melhor precisão e controle. A segunda condição garante a interrupção do processo em caso de estagnação, prevenindo o uso desnecessário de recursos computacionais (GOLDBERG, 2013).

Além disso, foi estipulado um limite superior de 100 gerações, servindo como salvaguarda para cenários onde as condições anteriores não sejam atingidas. Esse critério foi definido com base em testes empíricos que demonstraram que iterações acima de 100 já não melhoravam de forma significativa o valor do *fitness*. Ao final do processo evolutivo, o indivíduo com menor valor de NRMSE é selecionado como resultado do algoritmo, representando o conjunto de regras mais eficiente identificado ao longo da execução.

3.4 Implementação

Essa seção descreve como foram implementados os controladores que compõem o FALCOM. Todas as implementações, bem como os dados usados para gerar os gráficos apresentados neste documento, podem ser encontrados em um repositório público [1].

A implementação dos controladores fuzzy foi realizada utilizando a biblioteca Scikit-Fuzzy [2], Versão 0.4.2. Essa biblioteca oferece uma API de alto nível para desenvolvimento de sistemas de controle baseados na lógica fuzzy. Além disso, é compatível com a linguagem de programação Python [3], que foi a linguagem utilizada para a implementação de todo o projeto, usando sua Versão 3.12. A escolha dessa linguagem se deu pela sua facilidade de uso, bem como pelas bibliotecas disponíveis que auxiliam na manipulação de dados e operações matemáticas vetoriais, como a NumPy [4] e SciPy [5].

3.5 Usando FALCOM

Para utilizar o framework, quatro passos são necessários: (1) definir o *broker* de mensagens a ser utilizado, (2) escolher o parâmetro configurável relacionado ao controle de fluxo de mensagens,

1 https://github.com/gfads/FALCOM
2 https://pypi.org/project/scikit-fuzzy/
3 https://docs.python.org/3/
4 https://numpy.org/
5 https://scipy.org/

(3) integrar o controlador à uma arquitetura funcional para experimentação, (4) e, por fim, atualizar a variável de saída do controlador para o parâmetro definido.

A primeira etapa consiste na escolha do *broker* de mensagens que serve como base para a aplicação dos controladores fuzzy. Consideraram-se algumas opções amplamente utilizadas, como Apache Kafka[6], Mosquitto[7] e RabbitMQ[8]. Optou-se pelo RabbitMQ, por se tratar de uma solução *open-source* com ampla aceitação na indústria, suporte a múltiplos protocolos de mensagens (como *Advanced Message Queuing Protocol* (AMQP Team, 2025) e *Message Queuing Telemetry Transport* (MQTT, 2025)) e uma arquitetura flexível, que facilita o monitoramento e a manipulação de parâmetros internos de operação.

Na segunda etapa, é necessário selecionar o parâmetro configurável do *broker* a ser controlado dinamicamente pelos controladores fuzzy. No RabbitMQ, o único parâmetro que permite esse tipo de ajuste no consumidor é o *Prefetch Count* (PC) (RABBITMQ, 2025). Esse parâmetro determina quantas mensagens um consumidor pode receber sem enviar uma confirmação de processamento, afetando diretamente o desempenho, a latência e o acúmulo de mensagens na fila. O controle dinâmico do PC permite, portanto, ajustar o sistema às variações de carga e comportamento dos consumidores. Um valor alto para este parâmetro pode inundar o consumidor de mensagens sem que ele tenha capacidade para processá-las. Em contrapartida, um valor muito baixo pode manter o consumidor ocioso enquanto espera por novas mensagens.

A terceira etapa consiste na integração das instâncias dos controladores fuzzy à arquitetura do sistema proposto, conforme ilustrado na Figura 8.

O funcionamento do sistema de controle tem início com a publicação de mensagens pelo Produtor, que alimenta a *Fila* do *Broker*, o RabbitMQ. Essas mensagens são então consumidas pelo Consumidor, cuja taxa de processamento é influenciada pelo parâmetro PC.

A *Taxa de chegada de mensagens* no Consumidor é monitorada continuamente e enviada como *feedback* para o *Controlador Fuzzy*. Este valor é comparado ao valor de referência (*Setpoint*), resultando em um *Erro*. Esse erro é processado pelo *Controlador Fuzzy*, que retorna o ajuste necessário na variável PC para o Consumidor. Essa retroalimentação contínua permite que o sistema se adapte dinamicamente às variações na carga de mensagens e no desempenho dos consumidores.

[6] <https://kafka.apache.org/>
[7] <https://mosquitto.org/>
[8] <https://www.rabbitmq.com/>

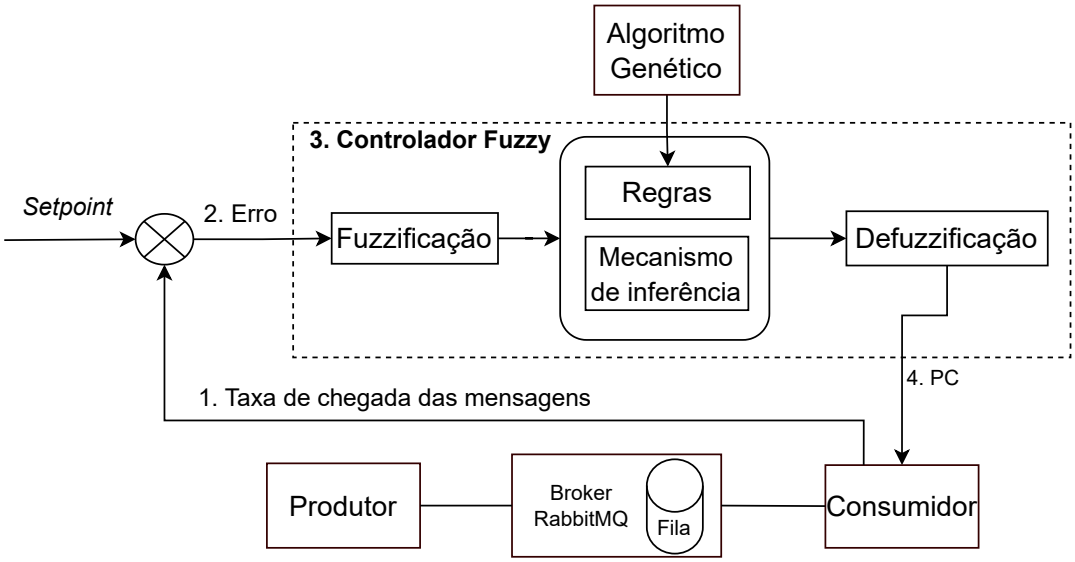

Figura 8 – Diagrama do *feedback loop*

A quarta etapa consiste em atualizar o controlador para a variável de saída estabelecida. O controlador fuzzy foi projetado com uma variável de entrada, denominada *Erro*, e uma variável de saída, que representa o ajuste no *prefetch count*, denominada de *PC*. A variável de entrada corresponde ao erro calculado como a diferença entre a taxa de chegada observada e o valor de referência (*Setpoint*), sendo discretizada no intervalo de -2000 msgs/s a 2000 msgs/s. Já a variável de saída, responsável por definir o ajuste aplicado ao *prefetch count*, é discretizada no intervalo de -6 a 6, sendo 1 o valor mínimo porque o valor de PC não pode ser negativo.

A definição desses intervalos foi fundamentada em experimentos exploratórios. Observou-se que, nos cenários simulados, não ocorreram erros absolutos superiores a 2000, tornando desnecessário o tratamento de valores além desse limite. No que se refere ao *PC*, a escolha da faixa foi motivada pela constatação de que, a partir de um valor de *prefetch count* igual a 12, não foram observados ganhos significativos na taxa de chegada de mensagens no consumidor. Essa relação é detalhada na Figura 9.

Nesse ciclo de monitoramento e ajuste, o *Controlador Fuzzy* ajusta continuamente o valor do *PC* com base na *Taxa de chegada de mensagens* observada e no *Setpoint* estabelecido. Esse comportamento é descrito no Algoritmo 1, que apresenta a lógica de atuação do Consumidor.

Inicialmente, três parâmetros são definidos: o intervalo de tempo entre as medições sucessivas (t), o tipo de controlador fuzzy a ser utilizado (c) e o *setpoint* desejado (s), que representa a taxa

Algoritmo 1: Comportamento do consumidor

```
   Entrada: t (intervalo de tempo), c (controlador), s (setpoint desejado)
1  while verdadeiro do
2      if tempo decorrido ≥ t segundos then
3          r ← ComputarMétricas();
4          pca ← InvokeFuzzyController(r, c, s);
5          AtualizarPC(pca);
6      else
7          m ← ReceberMensagem();
8          ProcessarMensagem(m);
```

de chegada ideal de mensagens. Em seguida, o consumidor estabelece conexão com o *broker*. A partir disso, o sistema entra em um loop contínuo de execução (Linhas 1 - 8). A cada intervalo de tempo t, o consumidor calcula a taxa de chegada de mensagens (r) e invoca o controlador fuzzy c para computar um novo valor de *Prefetch Count* (PC), com base em r e no *setpoint* s (Linhas 3 – 5). Esse novo valor de *PC* é então aplicado dinamicamente à configuração do consumidor (Linha 6). Caso o tempo t ainda não tenha sido atingido, o consumidor apenas recebe e processa a próxima mensagem disponível (Linhas 7 – 8).

Dentro do controlador fuzzy, primeiramente o erro entre a taxa observada e o *setpoint* é fuzzificado utilizando funções de pertinência previamente definidas. Em seguida, o mecanismo de inferência aplica um conjunto de regras fuzzy, geradas por meio do algoritmo genético, que mapeiam as variáveis de entrada fuzzy às saídas fuzzy correspondentes. Por fim, a etapa de defuzzificação converte o resultado em um valor que possa ser usado pelo *Consumidor*, que representa o novo valor ajustado para o *PC*.

3.5.1 Atuação do algoritmo genético

Como descrito na Seção 3.3, um algoritmo genético foi usado para a geração das regras que compõem o mecanismo de inferência do controlador fuzzy. Os dados utilizados para a execução do algoritmo foram coletados antes da atuação do consumidor. Para isso, o algoritmo avaliou a taxa de mensagens que chegava no consumidor enquanto o valor de *PC* era incrementado um a um. A Figura 9 exemplifica um tipo de conjunto de dados usado pelo algoritmo genético para a produção das regras. A partir desses dados, o AG calcula o conjunto de regras que guiará o comportamento

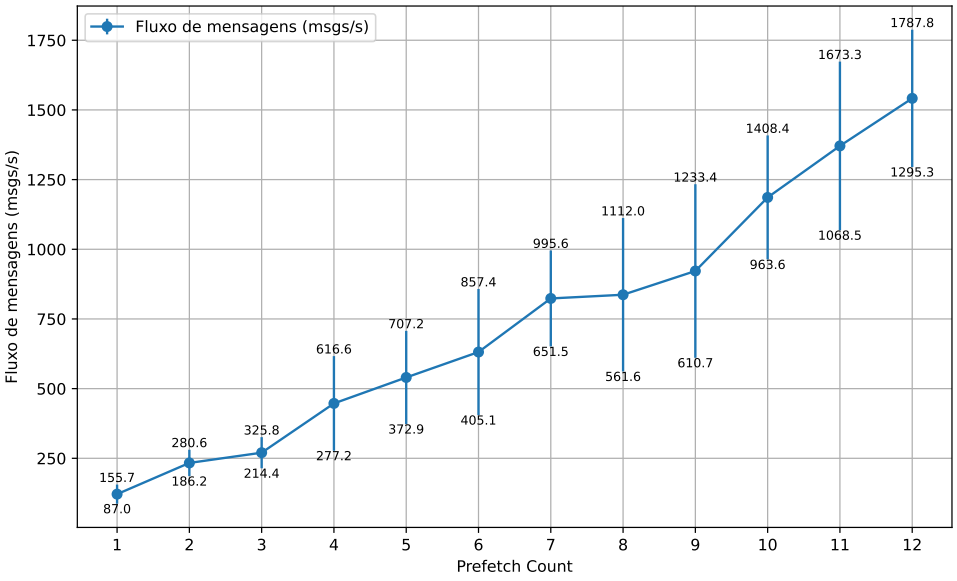

Figura 9 – Impacto do valor do *PC* na taxa de chegada de mensagens

do controlador fuzzy durante a execução.

Contudo, em alguns casos, as regras geradas podem estar excessivamente ajustadas a um determinado *setpoint*, fenômeno conhecido como sobreajuste. Isso reduz a capacidade do sistema de se adaptar a mudanças na meta, uma vez que seu comportamento se torna específico demais para uma meta particular. Esse problema é exemplificado na Figura 10, na qual, mesmo após mudanças no *setpoint*, a taxa de chegada permanece próxima da meta inicial, indicando que o controlador não consegue se ajustar de forma adequada. Para entender melhor como as amostras foram coletadas, ver Seção 4.5.1.

Para mitigar esse problema, foi incorporado ao fluxo de controle um mecanismo de atualização das regras a cada alteração do *setpoint*. Essa abordagem gera regras mais alinhadas às novas condições operacionais, ou seja, um conjunto de regras cuja aplicação resulte em melhor desempenho do sistema em relação à nova meta, sendo essa adequação avaliada por meio da métrica NRMSE. Essa atualização no mecanismo de inferência melhora o desempenho dos controladores, como mostra a Figura 11. Nela é possível ver uma melhora significativa na taxa de mensagens observada em relação ao cenário onde as regras são geradas uma única vez (Figura 10). Em relação ao NRMSE, houve uma redução de 0,4786 para 0,1967.

Apesar dos avanços, os resultados ainda apresentaram limitações, com um NRMSE de apenas

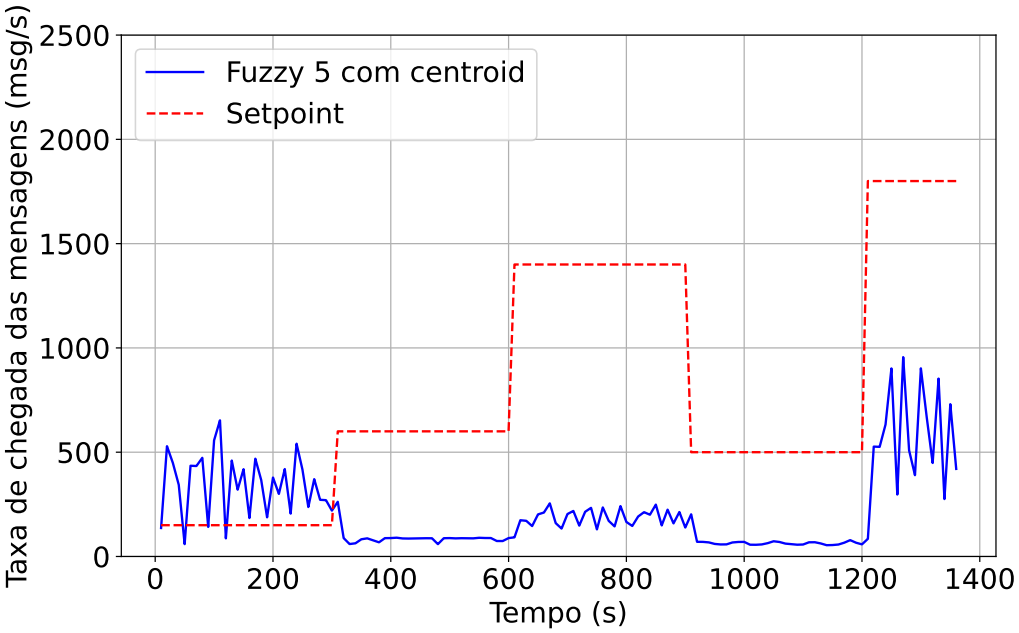

Figura 10 – Resultado com ajustes de regras específicos

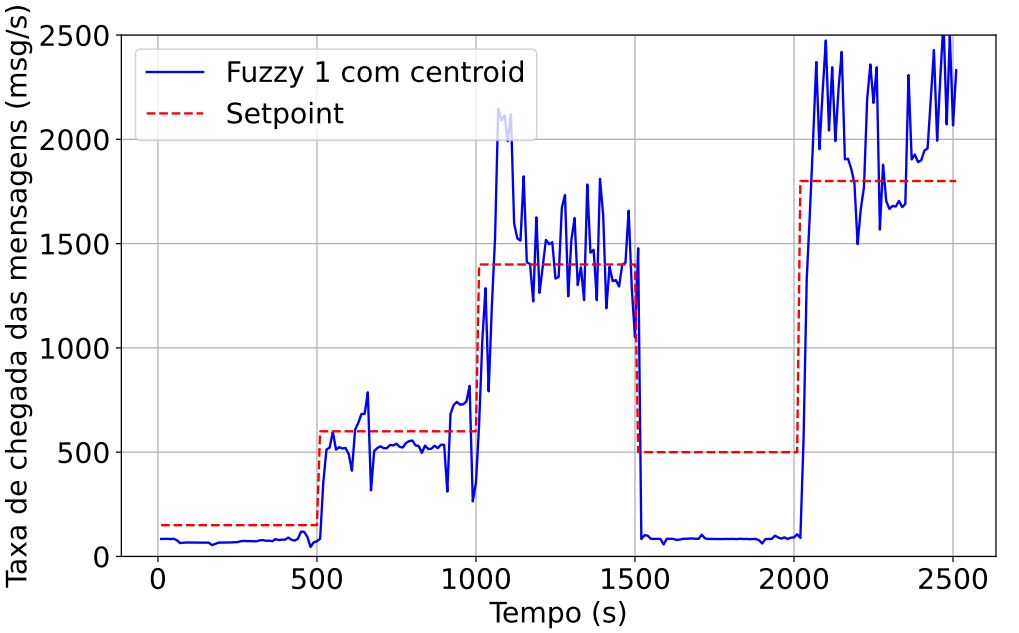

Figura 11 – Resultado com ajustes de regras para cada *setpoint*

0,365 e um *goal range* (GR) de 41% (ver Seção de métricas 4.3). Diante disso, foi desenvolvida uma abordagem adicional para tornar a atualização das regras mais dinâmica e sensível ao comportamento do sistema.

Essa nova abordagem analisa um histórico móvel da taxa de chegada das últimas amostras e verifica se os valores permanecem continuamente fora de uma faixa de tolerância de $\pm X\%$ em relação ao *setpoint*. Caso esse desvio seja consistente por um número configurável de iterações, o sistema define que a estratégia atual de controle pode não estar adequada. E então, novas regras são geradas pelo AG para o controlador, recalibrando-o para as novas condições operacionais.

Nesse processo, foi utilizada uma faixa de tolerância ±20% e uma janela de 10 iterações, ou seja, se por 10 iterações seguidas o fluxo de mensagens chegando no consumidor estiver fora da faixa de tolerância em relação ao *setpoint*, novas regras são computadas. Para esse processo, outro conjunto de dados semelhante ao da Figura 9 é gerado, pois assume-se que o anterior está obsoleto, e então o AG usa esse novo conjunto para estabelecer as novas regras.

3.6 Considerações Finais

Este capítulo apresentou uma solução para o controle de fluxo de mensagens em MOMs usando controladores baseados na lógica fuzzy. Os controladores foram incorporados no framework FALCOM, que contém também um mecanismo inteligente para adaptação das regras do controlador usando um algoritmo genético. Também foram apresentadas como usar o framework de forma descritiva e justificativas para as escolhas de cada um dos componentes que compõem os controladores.

4 Experimentos e Resultados

Este capítulo apresenta a avaliação experimental dos controladores implementados no framework FALCOM, e introduzidos no Capítulo 3. A Seção 4.1 apresenta os objetivos da avaliação, seguidos por uma descrição do ambiente em que os experimentos foram executados (Seção 4.2). Na seção seguinte, Seção 4.3, são apresentadas as métricas usadas nos experimentos. A Seção 4.4 apresenta os fatores levados em consideração nos experimentos. A Seção 4.5 apresenta como os experimentos foram conduzidos. Os resultados são avaliados e discutidos na Seção 4.6. Por fim, as considerações finais são apresentadas na Seção 4.7.

4.1 Objetivos

O objetivo dos experimentos é avaliar o desempenho dos controladores fuzzy aplicados ao controle de fluxo de mensagens em MOMs. O conjunto de controladores avaliados pode ser visto na Tabela 3. Para realizar a avaliação, foi utilizado o *broker* RabbitMQ em um ambiente com dois computadores conectados através da Internet, como mostrado na Figura 12.

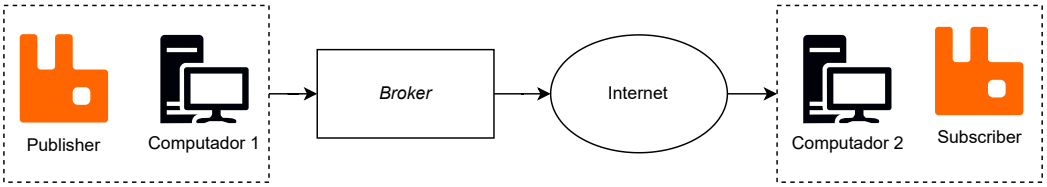

Figura 12 – Diagrama simplificado da configuração usada nos experimentos

4.2 Ambiente

Os experimentos foram realizados em um ambiente operacional cuja arquitetura foi projetada para isolar as funções de publicação e de consumo de mensagens. Para isso, *Publisher* e *Subscriber* foram executados em máquinas distintas. Esta configuração buscou eliminar interferências provenientes da competição por recursos computacionais, garantindo que a taxa de chegada de mensagens no *Subscriber* não fosse impactada por outros processos concorrentes. Assim, os resultados obtidos nas métricas são provenientes das alterações dos fatores utilizados.

O ambiente de execução do *Subscriber* consistiu em um computador equipado com um processador Intel® Core™ i5-13420H de 13ª geração (8 núcleos físicos e 12 *threads*), 8 GB de memória RAM e sistema operacional Ubuntu 24.04 LTS. Já a máquina responsável pela execução dos *Produtores* tinha um processador Intel® Core™ i5-8250U de 8ª geração (4 núcleos físicos e 8 *threads*), 12 GB de memória RAM e sistema operacional Ubuntu 22.04.4 LTS.

Ambos os computadores se comunicaram por meio de uma rede com velocidade nominal de 300 Mbps, cuja estabilidade e largura de banda foram suficientes para evitar gargalos na transmissão de mensagens durante os experimentos.

4.3 Métricas

Os controladores foram empregados para atuar no controle de fluxo de mensagens entre o *broker* e o consumidor. Na prática, a atuação dos controladores consistiu em ajustar dinamicamente um parâmetro no consumidor, o que, por sua vez, influenciou diretamente o comportamento do *broker* na entrega das mensagens. Conforme definido na Seção 4.1, o *broker* utilizado foi o RabbitMQ, e o parâmetro controlado foi o PC, que determina a quantidade máxima de mensagens que o *broker* pode enviar a um consumidor sem aguardar confirmações. Assim, embora o ajuste ocorra no lado do consumidor, é o comportamento do *broker* que é regulado.

Para avaliar o desempenho dos controladores, foi mensurado o quão próximo e estável o sistema permaneceu em relação ao *setpoint* durante a execução do consumidor. Com esse objetivo, foram utilizadas duas métricas: o *Normalized Root Mean Square Error* (NRMSE) e *Goal Range* (GR).

O RMSE (*Root Mean Square Error*) mede a diferença quadrática média entre os valores previstos por um modelo (ou esperados por um sistema) e os valores reais, que foram observados por meio de monitoramento. O RMSE é calculado de acordo com a Equação 4.1.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{4.1}$$

onde y_i representa o valor real (observado) da variável na i-ésima amostra, $\hat{y}_i$ é o valor estimado pelo modelo e n é o número total de amostras.

A normalização do erro visa eliminar a dependência escalar dos dados, facilitando, portanto, a comparação entre diferentes modelos ou, neste caso, entre os controladores. A Equação 4.2 define

como é realizado o cálculo do NRMSE.

$$\text{NRMSE} = \frac{RMSE}{y_{\text{ref}}}, \tag{4.2}$$

onde o RMSE é calculado como mostrado na Equação 4.1 e y_{ref} é o valor médio da amostra. Esta métrica foi escolhida porque oferece uma medida adimensional do erro, permitindo a comparação entre diferentes controladores. Quanto mais próximo de zero for o valor do NRMSE, melhor o desempenho do controlador, indicando menor discrepância entre os valores observados e os valores esperados.

Quanto ao GR, ela avalia o percentual de amostras cujos valores permaneceram dentro de uma faixa de tolerância ao redor do *setpoint*. A faixa define os limites inferiores e superiores considerados aceitáveis para a variável controlada. Ela pode ser definida como uma porcentagem do total de amostras que estão dentro dessa faixa estabelecida. Nesta avaliação, foi definida uma faixa de 20%, conforme descrito na Equação 4.3.

$$\text{GR} = \left(\frac{1}{n} \sum_{i=1}^{n} \delta_i \right) \times 100 \tag{4.3}$$

$$\delta_i = \begin{cases} 1, & \text{se } \hat{y}_i \in [0{,}8 \cdot s,\ 1{,}2 \cdot s] \\ 0, & \text{caso contrário} \end{cases}$$

Essa métrica foi escolhida como uma forma complementar de avaliar o desempenho dos controladores, focando não só na precisão média, como também na consistência e estabilidade ao longo da execução. Enquanto o NRMSE indica a discrepância média entre o valor observado e a referência, o GR indica a proporção de tempo que o fluxo de mensagens foi mantido dentro de uma faixa considerada aceitável em torno do *setpoint*. Nesse contexto, por ser uma métrica simples e expressa em percentual, o GR proporciona interpretação direta e intuitiva: quanto maior o valor do GR, maior o tempo em que o sistema esteve operando dentro da faixa estabelecida e, portanto, melhor a estabilidade do controlador. Essa interpretação facilita a comparação entre diferentes estratégias de controle e ajuda a identificar quais controladores são mais robustos frente a variações no sistema.

As métricas definidas são relevantes principalmente para o contexto de melhorias no controlador, pois indicam não só a eventual necessidade de reduzir o erro, mas também a necessidade

de aumentar a estabilidade, fazendo com que o fluxo permaneça dentro de uma faixa de operação aceitável.

4.4 Fatores

Para a avaliação dos experimentos, três fatores principais foram levados em consideração: o tipo de controlador, o perfil da referência adotada (*setpoint*) e a carga de trabalho gerada pelos produtores. Cada fator foi explorado em diferentes níveis.

O primeiro fator refere-se ao tipo de controlador fuzzy, representado por diferentes combinações entre funções de fuzzificação e de defuzzificação, totalizando vinte e quatro variações distintas (ver Tabela 3).

O segundo fator corresponde ao perfil do *setpoint* utilizado como alvo para o fluxo de mensagens processadas pelos consumidores. Foram definidos dois níveis para esse fator: fixos e variáveis.

O *setpoint* fixo avalia a capacidade dos controladores em manter a estabilidade do sistema quando a meta de processamento permanece constante. Nesse cenário, a referência utilizada foi de 1100 mensagens por segundo, valor escolhido por representar uma carga intermediária capaz de exercer impacto suficiente sobre o sistema sem, no entanto, levá-lo à saturação.

Por outro lado, os *setpoints* variáveis foram utilizados para analisar o comportamento dos controladores diante de mudanças dinâmicas na meta, simulando situações mais próximas daquelas encontradas em ambientes reais, onde o volume de mensagens pode variar de forma abrupta ou gradual.

A sequência adotada para os experimentos seguiu um perfil de variação suave e contínuo, aproximando-se visualmente de uma distribuição uniforme: 400, 800, 1200, 1600, 1200, 800 e 400 mensagens por segundo. Esse padrão foi selecionado para observar o desempenho do controlador em cenários com flutuações moderadas, onde é possível avaliar sua capacidade de adaptação sem a ocorrência de picos abruptos.

A adoção de diferentes tipos de referência (fixa e variável) permitiu realizar uma análise abrangente sobre dois aspectos fundamentais do desempenho dos controladores: sua estabilidade e sua adaptabilidade.

O terceiro fator incorporado aos experimentos é relacionado à carga de trabalho, representado pela taxa de publicação de mensagens pelos produtores. Esse fator foi incluído com o objetivo de

avaliar como o controlador se comporta sob diferentes níveis de demanda, refletindo desde cenários de operação leve até condições de carga mais elevada.

Foram definidos três níveis distintos de carga de trabalho: (i) 1000 mensagens por segundo, representando um cenário de baixa demanda, onde o sistema opera confortavelmente abaixo da sua capacidade máxima; (ii) 2500 mensagens por segundo, caracterizando uma situação de carga intermediária, na qual o sistema já se encontra sob pressão considerável, porém ainda dentro de condições controláveis; (iii) 5000 mensagens por segundo, simulando um cenário de alta demanda, em que eventuais deficiências no controle de fluxo tendem a se tornar mais evidentes.

Ao combinar os fatores de tipo de referência, perfil de variação da referência e níveis de carga de trabalho, os experimentos puderam abranger diversas situações operacionais. Essa abordagem possibilitou uma avaliação melhor quanto à eficácia, estabilidade e capacidade de adaptação dos controladores frente às diferentes condições impostas. A Tabela 4 resume os fatores e os níveis utilizados nos experimentos.

Tabela 4 – Fatores e níveis

Fatores	**Níveis**
#Tipo do controlador	Fuzzy 1, ..., Fuzzy 24 (Ver Tabela 3)
#*Setpoint*	*Setpoint* fixo, *setpoint* variável
#*Carga de trabalho*	1000 mensagens/s, 2500 mensagens/s, 5000 mensagens/s

4.5 Projeto dos experimentos

A comparação dos controladores foi realizada com um experimento *crossover* do tipo AB/BA. Este tipo de experimento permite comparar o desempenho dos controladores sob as mesmas condições experimentais, pois ele isola as variáveis e garante que os valores das métricas observadas são consequência da alteração dos fatores.

Cada controlador foi testado em todos os cenários definidos: com *setpoints* fixos e variáveis, bem como com diferentes níveis de carga de trabalho por parte dos produtores. Essa abordagem possibilitou não apenas uma análise comparativa mais elaborada, mas também a observação do desempenho dos controladores em situações que simulam tanto ambientes operacionais estáveis quanto dinâmicos, refletindo melhor as condições encontradas em sistemas de produção.

4.5.1 Coleta de dados

A coleta de dados foi realizada de forma periódica e cada amostra foi computada após uma janela temporal fixa de 10 segundos. Durante cada janela, o sistema monitorava continuamente a chegada de mensagens e contabilizava o total de mensagens processadas. Ao final do intervalo, eram computadas as métricas de interesse, como o valor da taxa de chegada média de mensagens (total de mensagens processadas/10s) e o valor atual do *PC*.

Cada experimento foi composto por uma sequência de amostras sucessivas, cuja quantidade definida foi 50 para cada *setpoint*, resultando em aproximadamente 58 minutos de atuação de cada controlador para a sequência de metas variáveis e 9 minutos para a sequência de meta fixa. O processo de coleta seguia até que a quantidade máxima de amostras para cada *setpoint* fosse atingida. Dessa forma, todos os controladores puderam ser avaliados de forma direta.

Além de medir o desempenho e computar as métricas, o processo de amostragem interagia com o consumidor, por meio do controlador. Ao final de cada janela, o controlador recebia as métricas computadas e devolvia o ajuste necessário para o *PC*. O processo de amostragem também foi sensível à dinâmica das metas. A cada quantidade definida de amostras, a meta vigente era atualizada conforme a sequência previamente estabelecida no cenário experimental (ver Seção 4.4).

4.6 Resultados e discussões

Conforme descrito na Seção 4.4, as métricas GR e NRMSE foram utilizadas para avaliar o desempenho dos controladores no controle de fluxo, com foco na estabilidade do sistema e na aderência à meta estabelecida. A seguir, são apresentados os resultados obtidos para cada controlador, considerando os fatores e níveis definidos na Tabela 4.

4.6.1 Desempenho com meta fixa

Considerando a atuação dos controladores com uma meta fixa, a Figura 13 apresenta a média de NRMSE obtida pelos controladores agrupados por método de fuzzificação.

A estabilidade do sistema sob a atuação dos controladores foi avaliada por meio da métrica GR, que complementa a análise de desempenho ao refletir a capacidade de manter a saída em torno da

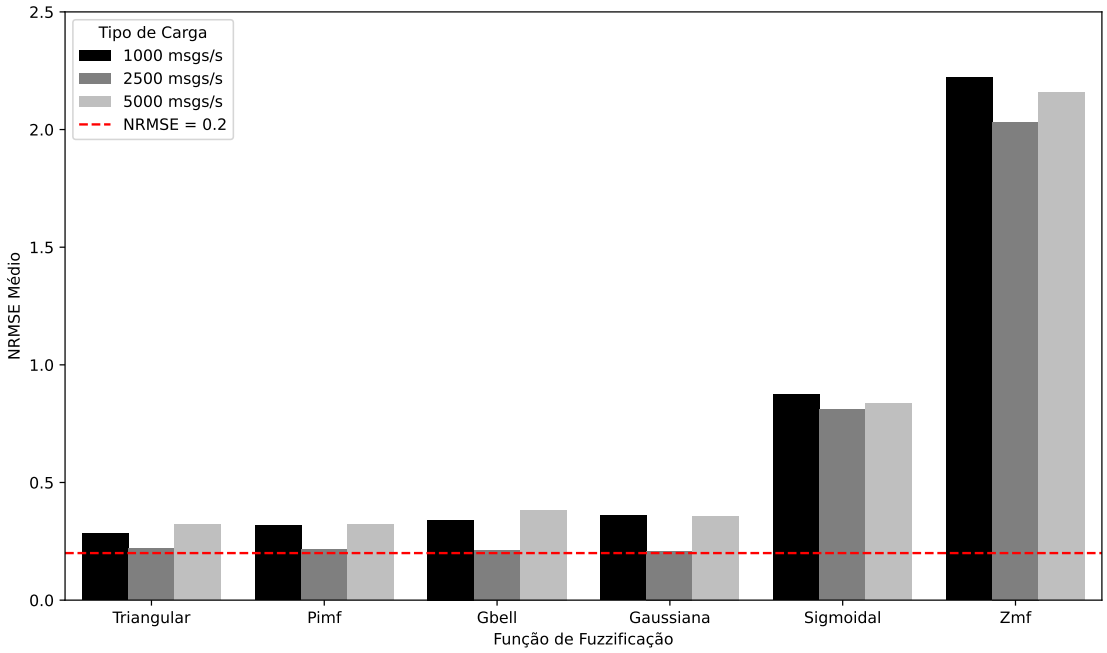

Figura 13 – NRMSE médio por função de fuzzificação e tipo de carga

meta. A Figura 14 apresenta a média dos valores de GR obtidos para cada função de fuzzificação considerada.

Considerando as cargas moderadas, os controladores baseados nas funções Triangular, *PI-shaped*, *G-Bell* e Gaussiana obtiveram resultados próximos, em torno de 70% a 79%, indicando um bom desempenho, ainda que abaixo do valor desejável de 80% representado pela linha tracejada. No entanto, há uma defasagem na estabilidade quando a carga é aumentada (5000 msgs/s) ou reduzida (1000 msgs/s), especialmente para as funções Gaussiana e *G-Bell*, que apresentam quedas acentuadas de aproximadamente 6% no GR, sob carga alta.

Por outro lado, a função Sigmoidal apresenta desempenho consistentemente inferior em todos os cenários, com GR médio variando entre 20% e 27%, demonstrando limitações na sua capacidade de adaptação às diferentes condições de carga. A função *Z-shaped* se destaca negativamente, com desempenho extremamente baixo, inferior a 2% em todas as cargas, tornando-se inviável para este contexto.

Considerando os métodos de defuzzificação, a Figura 15 apresenta as diferenças de desempenho de cada método utilizado, especialmente quando analisados sob cargas moderadas (2500 msgs/s). O método CoG apresenta o melhor desempenho relativo, alcançando aproximadamente 70% na carga de 2500 msgs/s, e valores de 42% e 26% nas cargas de 1000 msgs/s e 5000 msgs/s, respectivamente,

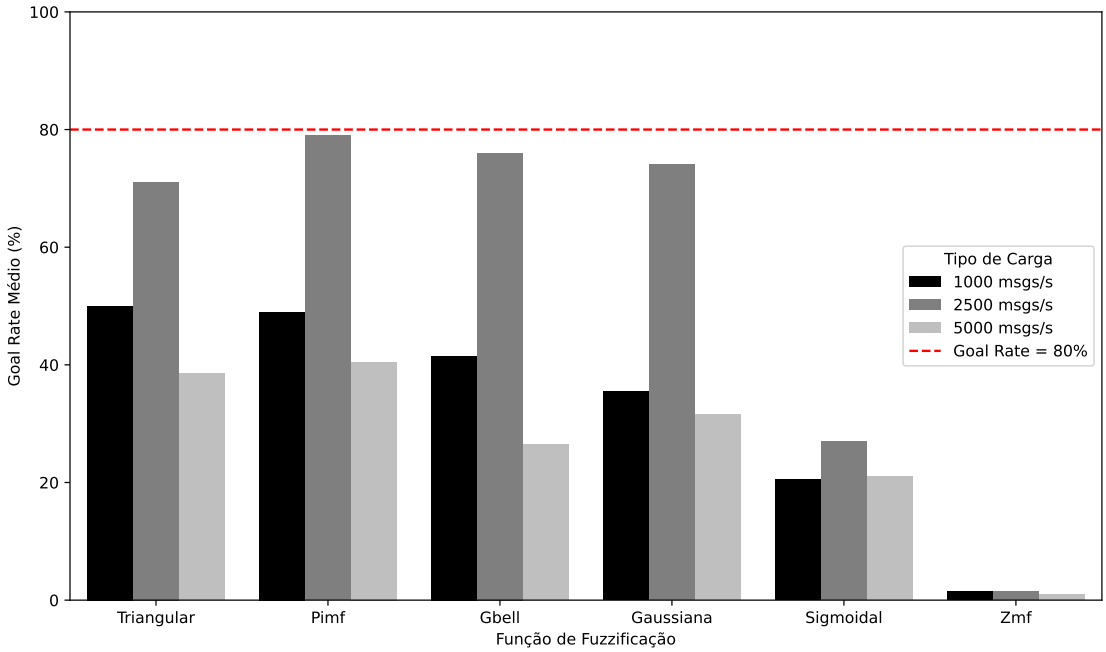

Figura 14 – GR médio por função de fuzzificação e tipo de carga

apresentando a melhor capacidade de adaptação em cenários moderados e baixos, embora sofra uma redução significativa sob carga alta.

O método BSC também apresenta bom desempenho, mantendo um comportamento semelhante ao CoG, porém ligeiramente inferior em cargas mais baixas (cerca de 37% em 1000 msgs/s) e em torno de 66% na carga moderada. Na carga alta, o BSC apresenta uma leve melhora em relação à carga baixa, chegando próximo dos 29%, o que sugere uma maior resistência à sobrecarga em comparação ao CoG.

O método MoM apresenta um comportamento semelhante ao BSC, porém com resultados ligeiramente inferiores na carga moderada (64%) e baixa (32%). Seu desempenho na carga alta (31%) é o mais consistente entre os três métodos que obtiveram o melhor desempenho (CoG, BSC e MoM), indicando menor variação entre os cenários de carga. Esse comportamento, embora não traga os maiores índices de GR, sugere uma maior robustez em situações de variação de carga.

Por fim, o método LoM apresenta o pior desempenho dentre todos os avaliados, com GR na faixa de 18% a 21%, independentemente da carga. Esse comportamento indica que o LoM não é adequado para este tipo de aplicação, dada sua incapacidade de adaptar-se às flutuações de carga.

Em resumo, os controladores fuzzy que utilizaram funções de fuzzificação Triangular, *PI-shaped*, Gaussiana e *G-Bell* obtiveram os melhores resultados, enquanto os controladores que usa-

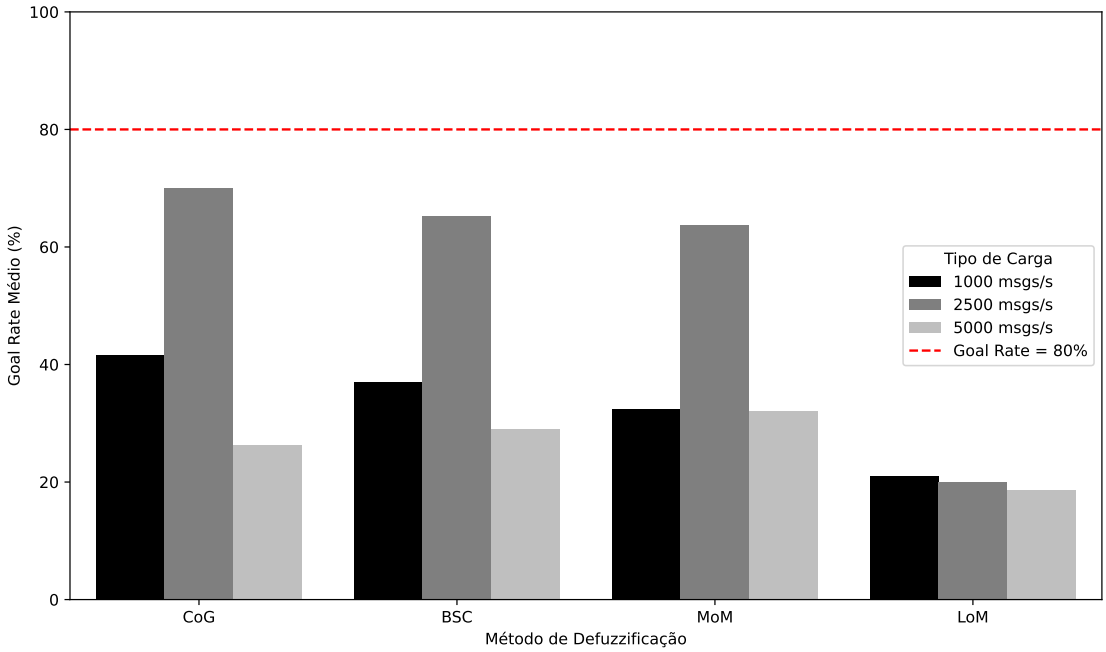

Figura 15 – GR médio por função de defuzzificação e tipo de carga

ram a função Sigmoidal tiveram um desempenho inferior em relação a eles. E aqueles controladores baseados na função *Z-shaped* obtiveram o pior desempenho geral, considerando ambas as métricas NRMSE e GR.

Além disso, nota-se que os métodos de defuzzificação baseados em cálculos de centro (CoG e BSC) demonstram melhor desempenho quando comparados aos métodos baseados em máximos (MoM e, especialmente, LoM). Esse resultado evidencia que métodos que consideram a distribuição completa dos graus de pertinência proporcionam decisões mais robustas, principalmente em ambientes de carga moderada.

4.6.2 Desempenho com metas variáveis

De forma geral, observou-se que os controladores fuzzy baseados nas funções de pertinência Triangular, Gaussiana, *G-Bell* e *PI-shaped* apresentaram desempenho significativamente superior em relação aos demais. Esses controladores mantiveram o NRMSE em níveis consistentemente baixos, independentemente da carga aplicada, indicando boa precisão e uma boa capacidade de controle.

A Figura 16 resume o desempenho de cada controlador baseado em cada uma dessas funções,

considerando o NRMSE médio de todos os controladores que usaram cada função de fuzzificação. Os controladores baseados nas funções Sigmoidal e *Z-shaped* tiveram os piores resultados, independentemente da carga do produtor. Os dados que permitiram a geração desse gráfico encontram-se no repositório do FALCOM[1].

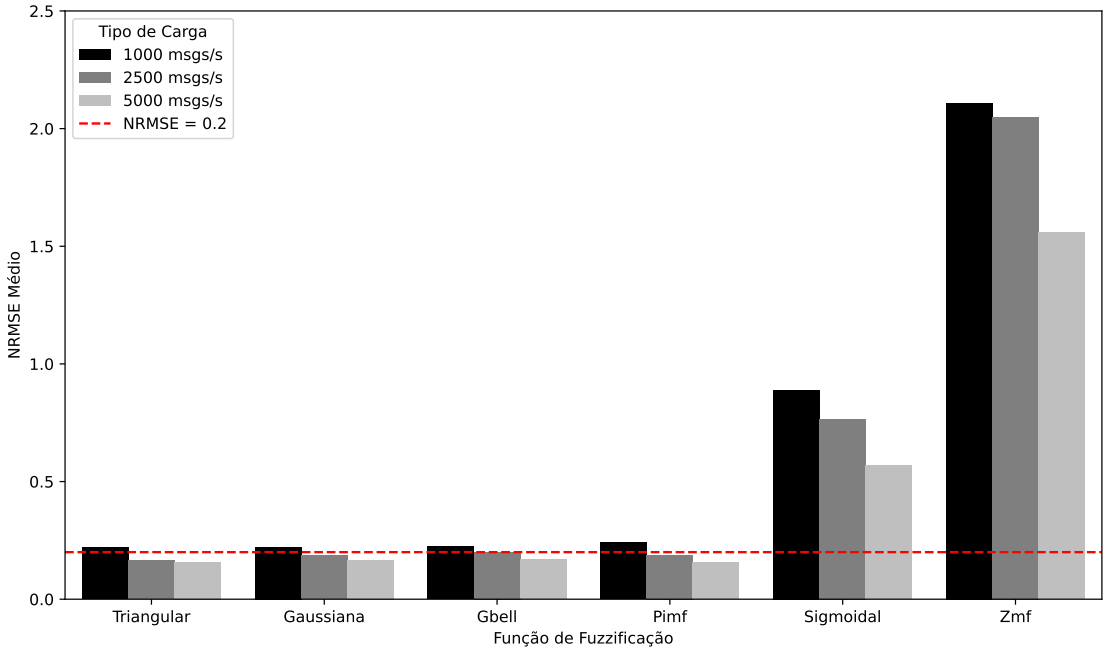

Figura 16 – NRMSE médio por função de fuzzificação e tipo de carga

Além disso, a meta definida para o NRMSE foi de um valor inferior a 0,2. Essa meta só foi alcançada nas configurações com carga superior a 2500 msgs/s. No entanto, a diferença em relação à carga mais baixa foi pouco expressiva, com variação inferior a 0,05.

Controladores com a função triangular destacaram-se frente aos demais, apresentando os menores valores médios de NRMSE nas cargas média e alta, além de alcançar o maior GR no cenário de carga média, como apresentado na Figura 17. Este resultado evidencia que, apesar da simplicidade da função triangular, ela oferece uma boa capacidade de generalização no controle do fluxo.

O GR também se manteve elevado para esses modelos (acima de 50%), especialmente no cenário de carga média, indicando uma boa capacidade de adaptação e estabilidade frente às variações da meta.

A Figura 17 apresenta um resumo do desempenho de cada controlador baseado em cada função de fuzzificação considerando a métrica GR. Nela é possível notar que, considerando a média,

[1] https://github.com/gfads/FALCOM/blob/main/final_datasets_FALCOM.zip

a função Triangular manteve seu destaque, obtendo a melhor média geral, considerando as cargas médias e altas (acima de 60%). Ao mesmo tempo, a melhor média geral considerando cargas baixas foi da função *G-Bell* (57%). As funções Sigmoidal e *Z-shaped* obtiveram os piores resultados, com destaque para a função *Z-shaped* cuja média ficou próxima de zero, apresentando muita instabilidade e uma capacidade de controle baixa.

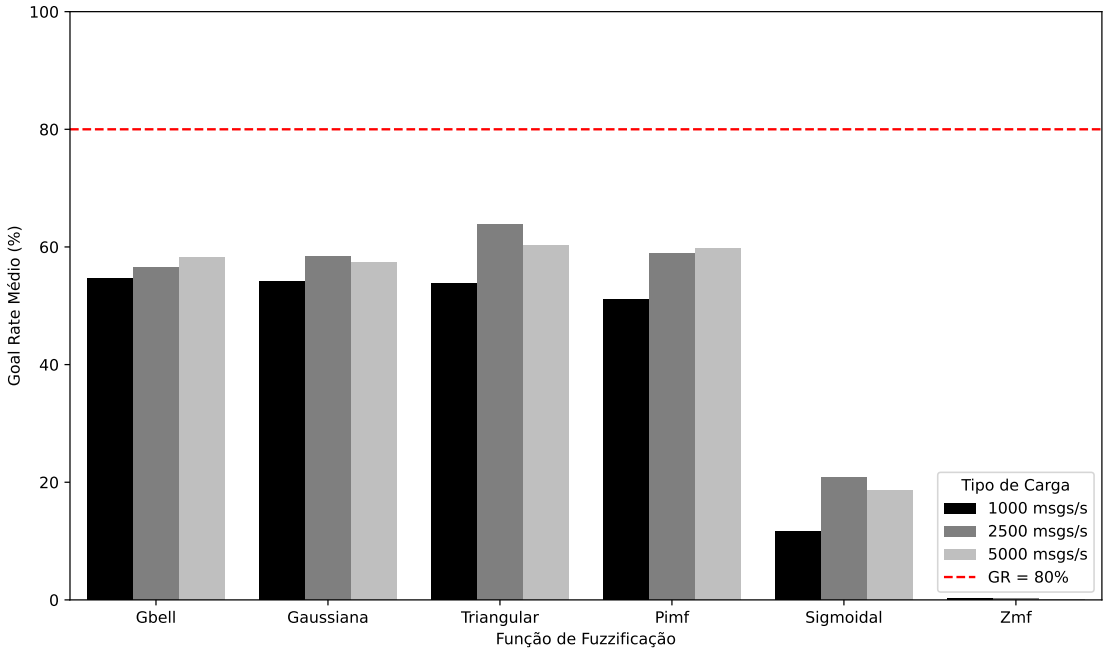

Figura 17 – GR médio por função de fuzzificação e tipo de carga

O destaque para os controladores com a função Triangular deve-se às suas transições lineares e bem definidas entre os conjuntos. Essa função não apresenta suavizações, o que resulta em respostas mais diretas e objetivas do controlador. Apesar de se destacar nesse contexto, é importante considerar que, com o aumento do número de variáveis ou a presença de maior variabilidade nos dados, essa falta de suavização pode levar ao sobreajuste.

O sobreajuste ocorre quando o controlador reage de forma excessivamente precisa a pequenas variações ou ruídos nos dados de entrada, tratando-os como padrões relevantes. Por exemplo, caso ocorra um pico de carga, o controlador pode interpretar essas flutuações como mudanças reais no comportamento do sistema e ajustar sua ação de forma desnecessária, comprometendo a estabilidade do sistema.

A função *PI-shaped*, por outro lado, atua de forma semelhante à triangular, mas com transições mais suaves entre os conjuntos, embora ainda mantenha bordas e limites bem definidos. Essa sua-

vização contribui para um equilíbrio entre a rigidez necessária para seguir as metas e a flexibilidade que evita reações exageradas a variações pontuais, o que ajuda a reduzir o risco de sobreajuste.

Os resultados obtidos pelas funções Gaussianas e *G-Bell* são consequências de suas naturezas mais suaves. Esta característica promove uma menor sensibilidade à mudanças na meta e flutuações abruptas na carga, o que evita oscilações. Porém, esse tipo de característica acaba tornando a resposta à mudanças na meta mais lenta, o que pode ter causado o resultado significativamente inferior à função triangular.

Já as funções Sigmoidal e *Z-shaped* demoram a responder às variações e respondem pouco às mesmas. Esse tipo de propriedade acaba por gerar atrasos na atuação do controlador, bem como a incapacidade de acompanhar mudanças na meta, o que gera acúmulo de erros e uma baixa taxa de acerto nas metas.

Considerando as funções de defuzzificação utilizadas, os resultados mostraram um comportamento aproximado, dentro de uma faixa de 35% a 60%. A Figura 18 apresenta uma comparação do desempenho dos quatro métodos de defuzzificação utilizados sob os três perfis de carga diferentes.

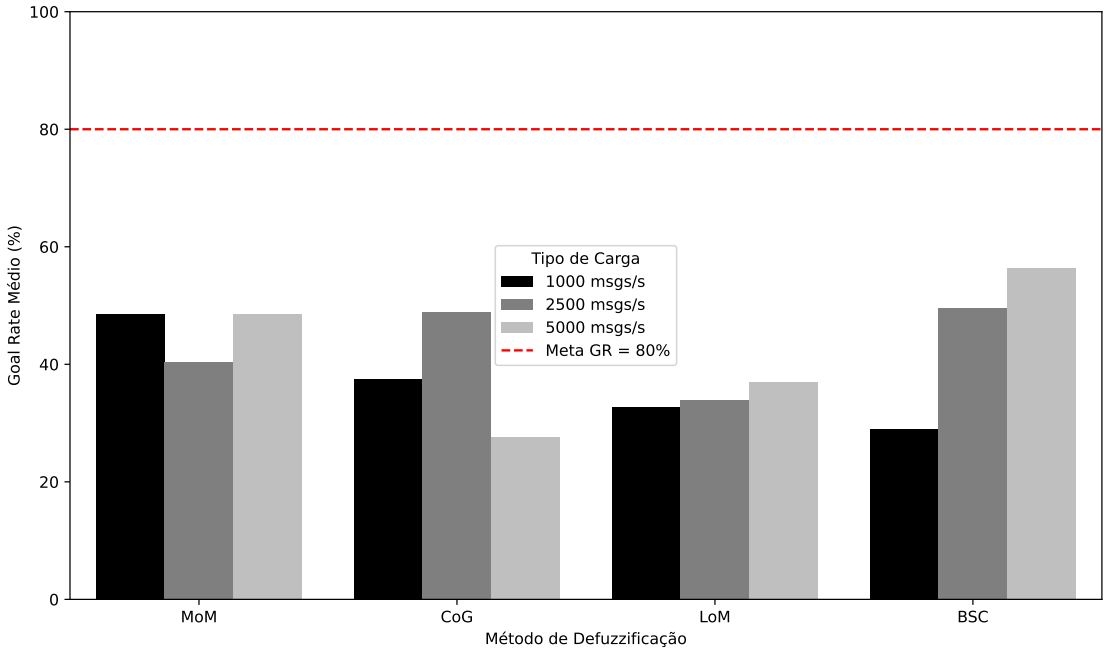

Figura 18 – GR médio por método de defuzzificação e tipo de carga

O MoM apresentou um desempenho equilibrado, com o GR próximo de 50% nas cargas baixa e alta, com um desempenho menor com a carga média, ficando em torno de 40%. Esse tipo de resultado sugere que este método é mais sensível em cenários de média pressão operacional.

O método LoM foi consistente, porém com desempenho inferior a MoM, mantendo o GR na faixa de 33% a 37%, relativamente estáveis independentemente do perfil de carga. Isso sugere que, apesar de sua simplicidade e baixo custo computacional, o LoM não é suficiente para atingir os níveis desejados de controle de metas.

Já o CoG apresentou comportamento mais sensível à variação de carga, com um desempenho que se deteriora significativamente sob carga alta (GR abaixo de 30%). Isso indica que, apesar de ser um dos métodos mais comuns em sistemas fuzzy, sua dependência de cálculos de centroides o torna menos robusto em ambientes sujeitos a flutuações rápidas e alta variabilidade.

O método BSC se destacou como aquele com melhor desempenho geral entre os avaliados, especialmente nas cargas mais altas. O GR avaliado evoluiu progressivamente conforme o aumento da carga, chegando a aproximadamente 57% em 5000 msgs/s, o que representa uma melhora de aproximadamente 5% frente ao MoM, que foi o segundo destaque. E com pouco mais de 10% em relação ao CoG e LoM. Esse comportamento sugere que o BSC é mais adequado em cenários com maior dinâmica operacional, possivelmente devido à sua capacidade de ponderar melhor os valores centrais em situações de maior dispersão nas funções de pertinência.

A Figura 19 apresenta o NRMSE médio por método de defuzzificação e tipo de carga.

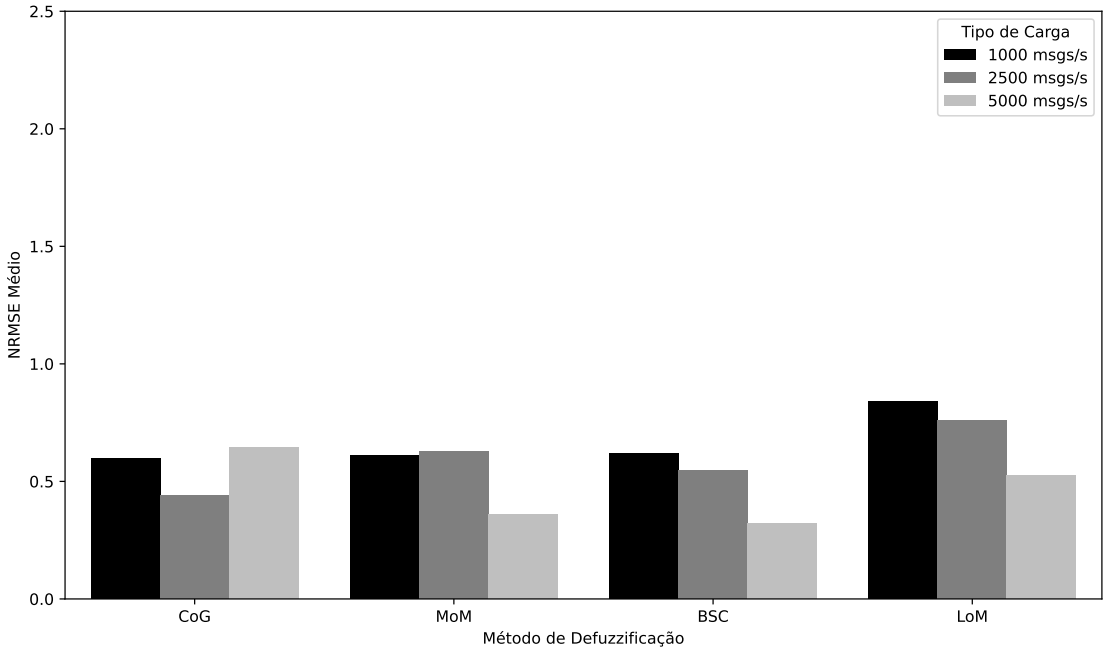

Figura 19 – NRMSE médio por método de defuzzificação e tipo de carga

O BSC se destacou apresentando os menores valores da métrica NRMSE, especialmente na

carga alta, com valores próximos a 0,32, que é o melhor resultado entre todos os métodos e cenários. Isso evidencia que, além de proporcionar melhores níveis de GR, o BSC também oferece rastreamento mais preciso do *setpoint*, especialmente quando o sistema está sob forte estresse operacional.

Já o LoM apresentou o pior resultado da métrica NRMSE, consistentemente mais alto em todas as cargas, chegando a valores próximos de 0,9 na carga baixa, o que reflete uma baixa capacidade de rastreamento do *setpoint*. Isso reforça a percepção de que o LoM, apesar de simples, não é adequado para esse tipo de controle.

O CoG apresentou um comportamento razoável, com erros mais baixos na carga média (aproximadamente 0,45) e crescimento do erro nas cargas mais altas (0,65 em 5000 msgs/s). Esse comportamento revela certa sensibilidade ao aumento da carga, embora seu erro permaneça menor que o do LoM.

Por fim, o MoM apresentou um erro relativamente constante nas cargas baixa e média (0,62), mas uma queda expressiva no NRMSE sob carga alta, descendo para 0,37. Esse comportamento é coerente com sua estabilidade observada no GR, sugerindo que o MoM mantém sua atuação robusta independentemente da carga, com melhora sob cenários mais dinâmicos.

De forma geral, embora os diferentes métodos de defuzzificação exerçam influência sobre o desempenho do controlador, a análise dos resultados evidencia que o fator que exerce maior impacto na capacidade de controle é a escolha do método de fuzzificação, ou seja, das funções de pertinência utilizadas. Observou-se que certas funções de pertinência conduziram a desempenhos significativamente inferiores, tanto em termos de GR quanto de NRMSE, quando comparadas a outras.

Além disso, a combinação entre o método de fuzzificação e o método de defuzzificação amplifica ou atenua os efeitos no desempenho. Por exemplo, funções de pertinência mais suaves, como a Gaussiana, *G-Bell* ou a *PI-shaped*, mostraram maior capacidade de adaptação frente a variações de *setpoint* e diferentes cargas, especialmente quando associadas a métodos de defuzzificação como o *bisector*, que privilegia respostas mais assertivas e conservadoras.

4.6.3 Impacto da carga de trabalho do produtor

A carga foi um fator importante para o desempenho dos controladores, como apresentado nas seções anteriores. Diferentes métodos de fuzzificação e defuzzificação tiveram resultados diferentes de acordo com a carga que estava sendo aplicada pelo produtor. A seguir, serão apresentados e discutidos os resultados considerando *setpoints* fixos e variáveis.

A Tabela 5 apresenta o desempenho de cada controlador utilizado considerando uma carga de 1000 msgs/s. A combinação Triangular com CoG apresentou o melhor desempenho geral, com baixo NRMSE (0,104) e alta GR (94%). Outros métodos, como *G-Bell* com BSC e *PI-shaped* com BSC, tiveram desempenho consideravelmente inferior, especialmente em relação ao GR, que ficou abaixo de 62%. As funções baseadas em *Z-shaped*, independentemente do método de defuzzificação, apresentaram os piores resultados, com NRMSE superior a 2 e GR praticamente nula, indicando ineficiência nesse cenário.

Tabela 5 – Desempenho dos controladores para carga de 1000 msgs/s com meta fixa.

Função de Fuzzificação	Método de Defuzzificação	Taxa de Entrega	NRMSE	GR (%)
Triangular	CoG	1000 msgs/s	0.104564	94.00
G-Bell	BSC	1000 msgs/s	0.259432	62.00
PI-shaped	BSC	1000 msgs/s	0.301546	54.00
Gaussiana	MoM	1000 msgs/s	0.304559	44.00
PI-shaped	CoG	1000 msgs/s	0.306904	50.00
Triangular	BSC	1000 msgs/s	0.306935	36.00
PI-shaped	MoM	1000 msgs/s	0.309401	46.00
G-Bell	MoM	1000 msgs/s	0.343647	44.00
Gaussiana	BSC	1000 msgs/s	0.352464	38.00
PI-shaped	LoM	1000 msgs/s	0.354987	46.00
Triangular	MoM	1000 msgs/s	0.359929	38.00
Triangular	LoM	1000 msgs/s	0.360438	32.00
G-Bell	CoG	1000 msgs/s	0.366748	38.00
Sigmoidal	CoG	1000 msgs/s	0.370166	30.00
Gaussiana	CoG	1000 msgs/s	0.370573	36.00
G-Bell	LoM	1000 msgs/s	0.391480	22.00
Gaussiana	LoM	1000 msgs/s	0.408488	24.00
Sigmoidal	BSC	1000 msgs/s	0.426980	30.00
Sigmoidal	MoM	1000 msgs/s	0.573273	20.00
Z-shaped	BSC	1000 msgs/s	2.107886	2.00
Sigmoidal	LoM	1000 msgs/s	2.127388	2.00
Z-shaped	CoG	1000 msgs/s	2.165312	2.00
Z-shaped	MoM	1000 msgs/s	2.179610	2.00
Z-shaped	LoM	1000 msgs/s	2.437143	0.00

Com o aumento da carga para 2500 msgs/s, observou-se uma mudança significativa no comportamento dos controladores. Combinações que anteriormente não se destacaram, como *G-Bell* com MoM e *G-Bell* com CoG, passaram a apresentar excelente desempenho, com NRMSE abaixo de 0,15 e GR superior a 92%. A função Gaussiana também mostrou desempenho robusto nesse cenário, especialmente com os métodos BSC e CoG, alcançando GR de até 92%. Por outro lado, os controladores baseados em *Z-shaped* mantiveram o desempenho significativamente inferior, com Taxa de Entrega muito baixa e GR próxima de zero. A Tabela 6 apresenta o desempenho de cada controlador utilizado considerando uma carga de 2500 msgs/s.

Tabela 6 – Desempenho dos controladores para carga de 2500 msgs/s com meta fixa.

Função de Fuzzificação	Método de Defuzzificação	Taxa de Entrega	NRMSE	GR (%)
G-Bell	MoM	2500 msgs/s	0.134051	92.00
Triangular	MoM	2500 msgs/s	0.136820	90.00
G-Bell	CoG	2500 msgs/s	0.146616	96.00
G-Bell	BSC	2500 msgs/s	0.149791	96.00
PI-shaped	MoM	2500 msgs/s	0.149791	88.00
Gaussiana	MoM	2500 msgs/s	0.152258	86.00
Gaussiana	BSC	2500 msgs/s	0.153696	92.00
Gaussiana	CoG	2500 msgs/s	0.154064	92.00
Triangular	BSC	2500 msgs/s	0.154130	84.00
PI-shaped	CoG	2500 msgs/s	0.161736	90.00
PI-shaped	BSC	2500 msgs/s	0.179812	96.00
Triangular	CoG	2500 msgs/s	0.185134	84.31
Sigmoidal	CoG	2500 msgs/s	0.208589	56.00
Sigmoidal	BSC	2500 msgs/s	0.285026	22.00
PI-shaped	LoM	2500 msgs/s	0.366281	42.00
Gaussiana	LoM	2500 msgs/s	0.376129	26.00
Triangular	LoM	2500 msgs/s	0.403049	26.00
Sigmoidal	MoM	2500 msgs/s	0.420541	24.00
G-Bell	LoM	2500 msgs/s	0.423395	20.00
Z-shaped	CoG	2500 msgs/s	1.753311	2.00
Z-shaped	BSC	2500 msgs/s	1.779351	2.00
Z-shaped	MoM	2500 msgs/s	1.929182	2.00
Sigmoidal	LoM	2500 msgs/s	2.323608	6.00
Z-shaped	LoM	2500 msgs/s	2.665522	0.00

Ao elevar a carga para 5000 msgs/s, nota-se uma queda geral no desempenho dos controladores, o que é esperado devido ao aumento da pressão sobre o sistema. Mesmo assim, alguns controladores mantiveram desempenho relativamente estável. A Tabela 7 apresenta o desempenho de cada controlador utilizado, considerando uma carga de 5000 msgs/s e metas fixas. A função Triangular, quando combinada com BSC ou MoM, apresentou os menores valores de NRMSE (próximos de 0,29) e GR entre 44% e 46%, mostrando uma boa capacidade de adaptação. Já os controladores que utilizam *G-Bell* ou Gaussiana, apesar de apresentarem NRMSE um pouco mais elevado, mantiveram GR entre 24% e 44%, dependendo do método de defuzzificação. De forma consistente, os controladores baseados em *Z-shaped* mantiveram o pior desempenho em todas as cargas testadas,

com GR próximo ou igual a zero e NRMSE sempre acima de 2.

Tabela 7 – Desempenho dos controladores para carga de 5000 msgs/s com meta fixa.

Função de Fuzzificação	Método de Defuzzificação	Taxa de Entrega	NRMSE	GR (%)
Triangular	BSC	5000 msgs/s	0.293090	44.00
Triangular	MoM	5000 msgs/s	0.295838	46.00
PI-shaped	BSC	5000 msgs/s	0.298958	34.00
PI-shaped	MoM	5000 msgs/s	0.300688	46.00
Gaussiana	MoM	5000 msgs/s	0.316934	44.00
Triangular	CoG	5000 msgs/s	0.318233	40.00
PI-shaped	CoG	5000 msgs/s	0.328095	38.00
PI-shaped	LoM	5000 msgs/s	0.354366	44.00
G-Bell	BSC	5000 msgs/s	0.358541	34.00
Gaussiana	BSC	5000 msgs/s	0.362733	32.00
G-Bell	MoM	5000 msgs/s	0.370641	32.00
G-Bell	CoG	5000 msgs/s	0.371379	26.00
Gaussiana	LoM	5000 msgs/s	0.373373	26.00
Gaussiana	CoG	5000 msgs/s	0.378634	24.00
Triangular	LoM	5000 msgs/s	0.382382	24.00
Sigmoidal	CoG	5000 msgs/s	0.391574	28.00
G-Bell	LoM	5000 msgs/s	0.433289	14.00
Sigmoidal	BSC	5000 msgs/s	0.434093	28.00
Sigmoidal	MoM	5000 msgs/s	0.552147	24.00
Sigmoidal	LoM	5000 msgs/s	1.964064	4.00
Z-shaped	CoG	5000 msgs/s	2.024951	2.00
Z-shaped	MoM	5000 msgs/s	2.178225	0.00
Z-shaped	BSC	5000 msgs/s	2.211072	2.00
Z-shaped	LoM	5000 msgs/s	2.216367	0.00

Por outro lado, nos cenários com metas variáveis, os resultados são significativamente melhores, sobretudo para cargas mais altas. Na carga de 5000 msgs/s, como mostrado na Tabela 8, a combinação da função *PI-shaped* com MoM apresentou o melhor desempenho global, alcançando o menor NRMSE entre todos (0,1016) e uma GR de 64,80%. A função de fuzzificação Gaussiana com defuzzificação MoM também obteve excelente desempenho, com NRMSE de 0,1023 e GR de 73,71%, revelando boa acurácia e adaptação. Outro destaque foi a combinação Triangular com BSC, que apresentou NRMSE de 0,1122 e a maior GR da tabela (83,33%), indicando uma boa estabilidade ao chegar próximo ao setpoint.

Em contraste, os piores desempenhos foram observados nas combinações envolvendo a função *Z-shaped* e o método CoG, como no caso de *Z-shaped* com CoG, que obteve o maior NRMSE (2,3369) e GR de 0%. Outros controladores com funções como Sigmoidal e *Z-shaped*, independentemente do método de defuzzificação, também apresentaram NRMSE acima de 1,2 e baixas taxas de GR (abaixo de 1%), sendo o pior desempenho obtido.

Tabela 8 – Desempenho dos controladores para carga de 5000 msgs/s com metas variáveis

Função de Fuzzificação	Método de Defuzzificação	Taxa de Entrega	NRMSE	GR (%)
PI-shaped	MoM	5000 msgs/s	0.101615	64.80
Gaussiana	MoM	5000 msgs/s	0.102327	73.71
Triangular	MoM	5000 msgs/s	0.110088	69.14
Triangular	BSC	5000 msgs/s	0.112243	83.33
PI-shaped	BSC	5000 msgs/s	0.112672	72.40
G-Bell	MoM	5000 msgs/s	0.126846	66.80
PI-shaped	LoM	5000 msgs/s	0.127132	59.60
G-Bell	BSC	5000 msgs/s	0.139890	67.14
Gaussiana	BSC	5000 msgs/s	0.144775	71.42
G-Bell	LoM	5000 msgs/s	0.169984	58.00
Gaussiana	LoM	5000 msgs/s	0.162175	59.14
Triangular	LoM	5000 msgs/s	0.177376	55.71
Sigmoidal	BSC	5000 msgs/s	0.223440	18.40
Triangular	CoG	5000 msgs/s	0.270281	37.14
PI-shaped	CoG	5000 msgs/s	0.280937	42.28
G-Bell	CoG	5000 msgs/s	0.296948	32.85
Gaussiana	CoG	5000 msgs/s	0.322165	28.00
Sigmoidal	CoG	5000 msgs/s	0.347931	25.42
Sigmoidal	MoM	5000 msgs/s	0.443023	29.60
Z-shaped	MoM	5000 msgs/s	1.254402	0.40
Sigmoidal	LoM	5000 msgs/s	1.257041	0.80
Z-shaped	BSC	5000 msgs/s	1.274286	0.00
Z-shaped	LoM	5000 msgs/s	1.364325	0.00
Z-shaped	CoG	5000 msgs/s	2.336882	0.00

Na carga de 2500 msgs/s com metas variáveis, os resultados apresentados na Tabela 9 indicam um desempenho consistente por parte de algumas combinações, especialmente aquelas envolvendo a função de fuzzificação Triangular. O melhor desempenho foi obtido com Triangular e BSC, que apresentou o menor NRMSE da tabela (0,1362) e a maior GR (74,00%), destacando-se pela boa

estabilidade e proximidade ao *setpoint*. Outras combinações com a mesma função, como Triangular com CoG (NRMSE de 0,1385 e GR de 65,71%) e Triangular com MoM (NRMSE de 0,1497 e GR de 65,71%), também se mostraram eficazes, reforçando a robustez dessa função de fuzzificação nesse cenário.

Além disso, a função *PI-shaped* com CoG obteve desempenho competitivo, com NRMSE de 0,1469 e GR de 70,00%, enquanto a Gaussiana com BSC apresentou NRMSE de 0,1670 e GR de 68,00%, ambas demonstrando boa adaptação e controle sob metas variáveis. A função *G-Bell*, por sua vez, teve desempenho intermediário, com destaque para a combinação com MoM (NRMSE de 0,1775 e GR de 61,43%).

Em contrapartida, os piores resultados foram obtidos pelas combinações com a função *Z-shaped*, como *Z-shaped* com MoM (NRMSE de 2,2738 e GR de apenas 0,28%) e *Z-shaped* com BSC (NRMSE de 2,2755 e GR de 0,29%). A função Sigmoidal também apresentou desempenho insatisfatório na maioria das combinações, como Sigmoidal com LoM (NRMSE de 1,8036 e GR de 0%) e Sigmoidal com MoM (NRMSE de 0,7686 e GR de apenas 5,14%). Esses resultados reforçam o baixo grau de adaptação dessas funções sob condições de metas dinâmicas, demonstrando limitações significativas para o uso em sistemas de controle adaptativo.

Tabela 9 – Desempenho dos controladores para carga de 2500 msgs/s com metas variáveis.

Função de Fuzzificação	Método de Defuzzificação	Taxa de Entrega	NRMSE	GR (%)
Triangular	BSC	2500 msgs/s	0.136202	74.00
Triangular	CoG	2500 msgs/s	0.138469	65.71
PI-shaped	CoG	2500 msgs/s	0.146857	70.00
Triangular	MoM	2500 msgs/s	0.149748	65.71
Gaussiana	BSC	2500 msgs/s	0.166988	68.00
Gaussiana	MoM	2500 msgs/s	0.173245	57.42
G-Bell	MoM	2500 msgs/s	0.177470	61.42
Gaussiana	CoG	2500 msgs/s	0.180013	58.85
G-Bell	LoM	2500 msgs/s	0.194851	50.85
PI-shaped	LoM	2500 msgs/s	0.197539	52.85
PI-shaped	BSC	2500 msgs/s	0.199649	60.85
G-Bell	CoG	2500 msgs/s	0.203390	58.28
PI-shaped	MoM	2500 msgs/s	0.209963	52.00
Gaussiana	LoM	2500 msgs/s	0.217882	49.42
Sigmoidal	CoG	2500 msgs/s	0.227384	39.71
Triangular	LoM	2500 msgs/s	0.229271	50.00
G-Bell	BSC	2500 msgs/s	0.229645	55.71
Sigmoidal	BSC	2500 msgs/s	0.266025	38.28
Sigmoidal	MoM	2500 msgs/s	0.768609	5.14
Z-shaped	CoG	2500 msgs/s	1.741810	0.28
Sigmoidal	LoM	2500 msgs/s	1.803604	0.00
Z-shaped	LoM	2500 msgs/s	1.903538	0.00
Z-shaped	MoM	2500 msgs/s	2.273833	0.28
Z-shaped	BSC	2500 msgs/s	2.275495	0.28

Nos cenários com metas variáveis e carga de 1000 msgs/s, conforme apresentado na Tabela 10. O melhor desempenho global foi alcançado pela combinação *G-Bell* com MoM, que obteve o menor NRMSE da tabela (0,1220) e a maior GR (88,86%), evidenciando excelente capacidade de adaptação e precisão. A combinação *PI-shaped* com MoM também apresentou bons resultados, com NRMSE de 0,1337 e GR de 82,00%, reforçando a eficácia do método MoM nesse contexto. Outro destaque foi a combinação *G-Bell* com CoG, com NRMSE de 0,1637 e GR de 64,29%, demonstrando bom equilíbrio entre estabilidade e aproximação ao *setpoint*.

Em contraste, os piores desempenhos foram observados em combinações com a função *Z-shaped*, como *Z-shaped* com BSC e *Z-shaped* com MoM, que apresentaram NRMSE superiores a

2,1 e GR próximos de 0%. As combinações com a função Sigmoidal também foram insatisfatórias, como Sigmoidal com LoM (NRMSE de 2,0500 e GR de 0%) e Sigmoidal com MoM (NRMSE de 0,7566 e GR de apenas 3,71%). Esses resultados indicam baixa capacidade de adaptação e elevada oscilação diante de metas dinâmicas, o que compromete a efetividade do controle fuzzy nesses casos.

Tabela 10 – Desempenho dos controladores para carga de 1000 msgs/s com metas variáveis.

Função de Fuzzificação	Método de Defuzzificação	Taxa de Entrega	NRMSE	GR (%)
G-Bell	MoM	1000 msgs/s	0.122022	88.85
PI-shaped	MoM	1000 msgs/s	0.133708	82.00
G-Bell	CoG	1000 msgs/s	0.163737	64.28
Triangular	LoM	1000 msgs/s	0.169367	64.00
Gaussiana	LoM	1000 msgs/s	0.187931	60.00
Gaussiana	CoG	1000 msgs/s	0.197679	57.71
Gaussiana	MoM	1000 msgs/s	0.206664	65.14
Triangular	MoM	1000 msgs/s	0.239254	50.85
Triangular	BSC	1000 msgs/s	0.250389	46.57
PI-shaped	LoM	1000 msgs/s	0.270729	40.28
PI-shaped	CoG	1000 msgs/s	0.274413	41.71
G-Bell	BSC	1000 msgs/s	0.281932	34.00
PI-shaped	BSC	1000 msgs/s	0.282311	40.28
Gaussiana	BSC	1000 msgs/s	0.293779	33.71
G-Bell	LoM	1000 msgs/s	0.326225	31.71
Sigmoidal	CoG	1000 msgs/s	0.340297	23.14
Sigmoidal	BSC	1000 msgs/s	0.409855	19.42
Sigmoidal	MoM	1000 msgs/s	0.756600	3.71
Z-shaped	CoG	1000 msgs/s	2.014329	0.28
Z-shaped	LoM	1000 msgs/s	2.039393	0.57
Sigmoidal	LoM	1000 msgs/s	2.050016	0.00
Z-shaped	BSC	1000 msgs/s	2.188746	0.00
Z-shaped	MoM	1000 msgs/s	2.192349	0.28

Por fim, em relação aos resultados com metas variáveis, é possível perceber que, mesmo sob cargas mais altas, os controladores apresentam um comportamento um pouco mais estável em termos de NRMSE, especialmente para funções Triangular, *PI-shaped* e *G-Bell* combinadas com BSC e MoM. No entanto, embora o erro permaneça controlado (em torno de 0,1 a 0,14 nos melhores

casos), o GR não acompanha essa estabilidade, indicando que há uma limitação na capacidade do controlador em manter o fluxo de mensagens sob controle de maneira eficiente. A função *Z-shaped* apresentou um desempenho ineficiente em todos os cenários.

4.6.4 Análise individual dos controladores

Conforme apresentado nas seções anteriores, em ambientes com metas variáveis, o método de defuzzificação que se destaca é o *bisector* combinado com funções de fuzzificação mais suaves, que são menos sensíveis a ruídos nas taxas monitoradas. Além disso, essas características apresentaram um bom desempenho em cargas mais altas do produtor. Sendo essas características presentes nos cinco melhores controladores de acordo com o NRMSE e o GR calculados (ver Tabela 8).

O melhor controlador em relação a todos os outros foi a combinação do método *PI-shaped* e MoM para fuzzificação e defuzzificação, respectivamente, com uma carga de 5000 msgs/s. Porém, a diferença não foi tão significativa em relação aos demais presentes no conjunto dos três melhores. A Figura 20 mostra o comportamento do fluxo de mensagens ao longo do tempo sob controle dos três melhores controladores observados nos experimentos.

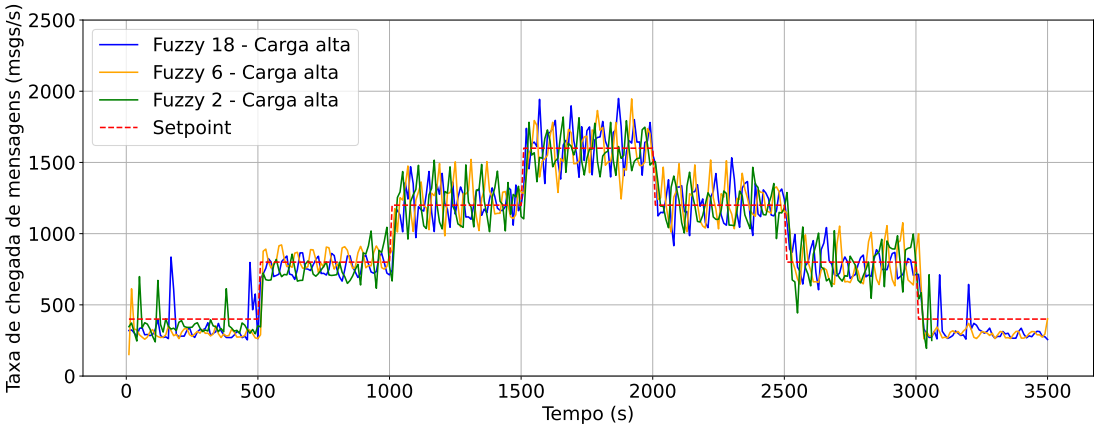

Figura 20 – Fluxo de mensagens controlados pelos três melhores controladores experimentados

A função de fuzzificação *PI-shaped* se destaca por estar presente em duas das cinco melhores combinações, juntamente com a função Triangular. Além disso, os métodos de defuzzificação MoM e *bisector* também se destacam, estando presentes entre os três primeiros colocados e nas quarta e quinta posições, respectivamente.

Em contrapartida, a função de fuzzificação *Z-shaped* se destacou de forma negativa, apresentando os piores resultados individuais. A Figura 21 apresenta o fluxo de mensagens quando controlado pelos três controladores que obtiveram o pior desempenho.

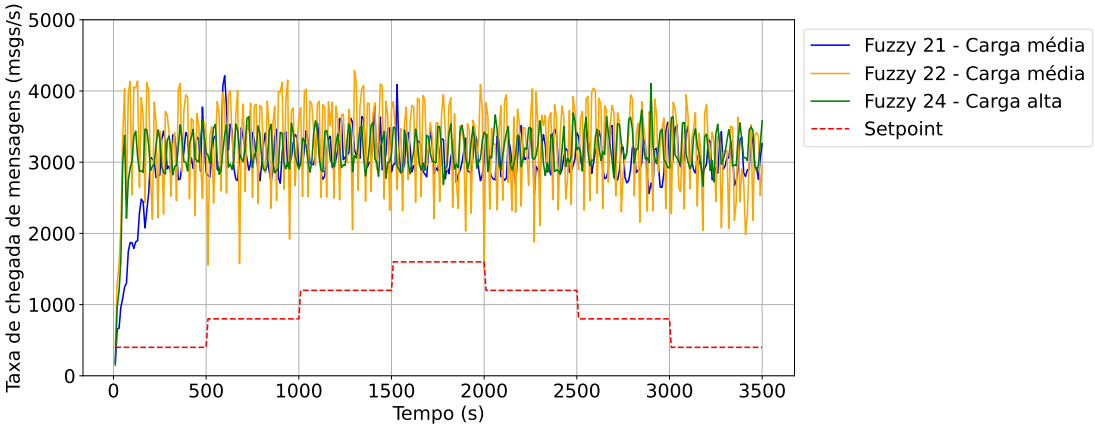

Figura 21 – Fluxo de mensagens controlados pelos três piores controladores experimentados

Por outro lado, considerando metas fixas, foi observado que as funções triangular e *G-Bell* se destacaram como funções de fuzzificação, enquanto CoG e MoM se destacaram como funções de defuzzificação. A Figura 22 mostra o desempenho dos três melhores controladores considerando a meta fixa em 1100 msgs/s.

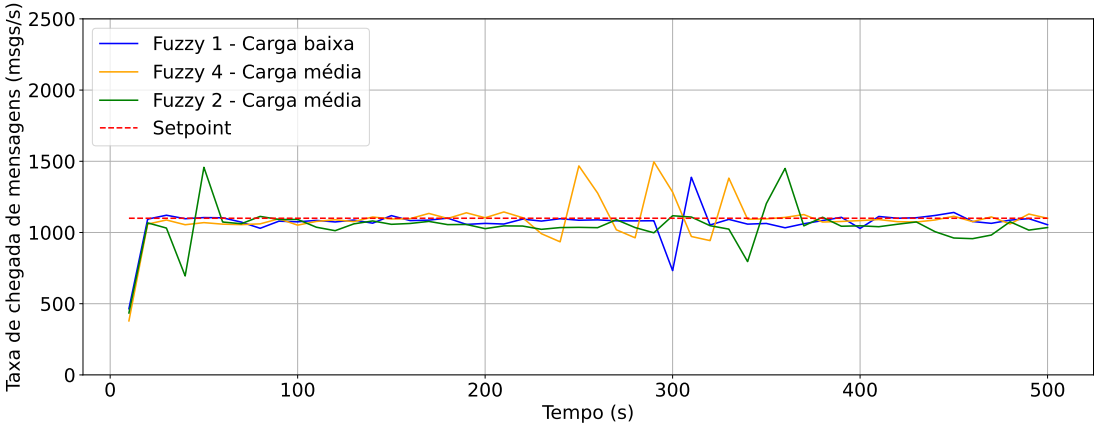

Figura 22 – Fluxo de mensagens sob atuação dos controladores Fuzzy 1, 2 e 4

Conforme apresentado nas seções anteriores, a função de fuzzificação *Z-shaped* e o método de defuzzificação LoM obtiveram o pior desempenho nos experimentos conduzidos. A Figura 23 mostra o fluxo de mensagens sob o controle dos controladores fuzzy 21, 22 e 24.

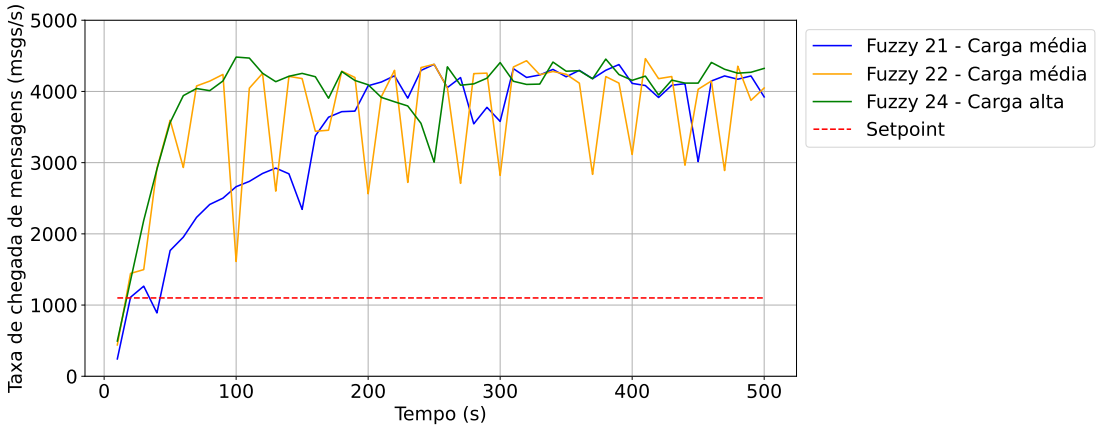

Figura 23 – Fluxo de mensagens sob atuação dos controladores Fuzzy 21, 22 e 24

4.7 Considerações Finais

Este capítulo apresentou a condução dos experimentos realizados com os controladores fuzzy desenvolvidos no framework FALCOM, avaliando seu desempenho sob diferentes combinações de funções de fuzzificação e métodos de defuzzificação, em cenários com metas fixas e variáveis de fluxo de mensagens. A análise foi fundamentada em duas métricas principais: NRMSE, que mensura a precisão em relação ao *setpoint*, e GR, que expressa a estabilidade do sistema ao longo do tempo. Os resultados revelaram padrões consistentes de desempenho, permitindo identificar configurações mais adequadas para diferentes condições operacionais. De modo geral, verificou-se que certas combinações de métodos de fuzzificação e defuzzificação apresentaram melhor capacidade de controle e adaptação frente a variações de carga, enquanto outras demonstraram maior sensibilidade ou instabilidade.

5 Trabalhos relacionados

Esse capítulo apresenta os trabalhos relacionados com o que está sendo proposto nesta dissertação. Trabalhos relacionados ao escopo deste estudo constituem-se no uso de controladores da teoria de controle aplicados para adaptação de middleware, bem como o uso da lógica fuzzy em sistemas computacionais e de mecanismos da computação evolucionária, como algoritmos genéticos, para incrementar a robustez do uso da lógica fuzzy. A Seção 5.1 apresenta uma visão geral do que será discutido no presente capítulo, resgatando conceitos importantes apresentados. A Seção 5.2 apresenta trabalhos relacionados ao uso da teoria do controle em middleware. Em seguida, são explorados trabalhos que usam a lógica fuzzy em sistemas computacionais na Seção 5.3 e trabalhos que utilizam computação evolutiva aplicada à lógica fuzzy na Seção 5.4. Por fim, é discutido como a abordagem aqui apresentada evolui no estado da arte na Seção 5.5 e a Seção 5.6 resume o que foi apresentado, concluindo o capítulo.

5.1 Visão geral

Nas últimas décadas, os controladores incorporados na ciência da computação como o PID, *On/Off*, têm sido amplamente empregados para promover adaptações automáticas em sistemas distribuídos. Trabalhos como (SABUHI; MAHMOUDI; KHAZAEI, 2021; FILIERI et al., 2017) propõem um processo sistemático de projeto de controladores para sistemas de software auto-adaptativos, fundamentado na teoria do controle. A abordagem abrange desde a identificação de metas e atuadores até a síntese e verificação formal de um sistema em *closed-loop*, buscando garantir propriedades como estabilidade, robustez e tempo de resposta. Esses controladores apresentam uma estrutura simples e bem compreendida, sendo eficazes em diversas aplicações industriais. No contexto de sistemas auto-adaptativos, Weyns (WEYNS, 2018) realizou uma análise abrangente de diferentes modelos baseados na teoria de controle, destacando especialmente como esses modelos podem ser utilizados para garantir propriedades como robustez e estabilidade em ambientes dinâmicos.

5.2 Teoria do Controle aplicada a middleware

Mais recentemente, a aplicação da teoria de controle em sistemas distribuídos que operam sob variações de carga, como os sistemas de middleware orientados a mensagens, tem-se destacado significativamente.

Peter et al. (PIETZUCH; BHOLA, 2003) demonstraram a aplicabilidade de *feedback loops* no problema de controle de congestionamento em MOMs, ao proporem uma arquitetura composta por dois protocolos: um voltado à limitação da taxa de chegada de novas mensagens e outro à restrição da taxa de *non-acknowledgments*, ambos implementados como parte integrante de um MOM. Os experimentos conduzidos indicaram que a arquitetura proposta era capaz de manter o congestionamento em níveis aceitáveis, mesmo diante de flutuações de carga.

No mesmo contexto, Ravi et al. (KUMAR; MUTHUSAMY; BABU, 2012) propuseram uma solução de controle adaptativo voltada à autoadaptação de *enterprise messaging servers*, por meio da utilização de um controlador proporcional (P). O estudo concentrou-se em métricas como taxa de entrega de mensagens (*throughput*) e latência, propondo mecanismos de controle capazes de adaptar dinamicamente parâmetros do middleware durante a execução. Os resultados evidenciaram que o controlador P promoveu melhorias significativas no desempenho do middleware em ambientes distribuídos.

Em esforços mais recentes voltados à adaptação dinâmica de sistemas distribuídos, Rosa et al. (ROSA; CAVALCANTI, 2022; ROSA; CAVALCANTI, 2023; ROSA; CAVALCANTI, 2024) propuseram uma abordagem sistemática para a aplicação de controladores no controle, em tempo de execução, do fluxo de mensagens entre brokers comerciais e consumidores, no contexto de arquiteturas *publish/-subscribe*. A proposta distingue-se por oferecer um guia prático e replicável, que abrange desde a definição dos controladores até sua integração efetiva em sistemas de mensageria amplamente utilizados. O objetivo central é possibilitar o ajuste dinâmico de parâmetros tradicionalmente estáticos dos brokers, em resposta a variações no ambiente ou na carga de trabalho. Além da modelagem da lógica de controle, o trabalho investigou diferentes estratégias de ajuste dos parâmetros dos controladores, com ênfase especial nos controladores do tipo PID (Proporcional-Integral-Derivativo). Essa investigação incluiu a avaliação de métodos de *tuning* sob distintos cenários operacionais, com diferentes dinâmicas e padrões de carga, permitindo identificar as abordagens que oferecem maior robustez, estabilidade e capacidade de adaptação. A análise comparativa entre variantes do

PID e suas respectivas estratégias de parametrização evidenciou que a escolha do método de *tuning* impacta diretamente o desempenho do sistema.

A partir da abordagem proposta por Rosa (ROSA; CAVALCANTI, 2023), Almeida (ALMEIDA, 2023) realizou uma análise comparativa entre diferentes controladores, incluindo variações do PID, bem como controladores recentemente propostos, como o HPA (Kubernetes, 2025) e o AsTAR (YANG et al., 2021), aplicados ao broker comercial RabbitMQ. O foco da análise esteve na adaptação dinâmica das configurações do broker, com vistas ao controle da taxa de chegada de mensagens no *subscriber* durante a execução do sistema.

5.3 Lógica Fuzzy em sistemas computacionais

Apesar do bom desempenho apresentado por controladores clássicos, uma de suas limitações está na suposição de que as variáveis do sistema se relacionam de maneira linear. Em sistemas distribuídos reais, que são caracterizados por comportamento dinâmico, incertezas e relações não-lineares entre seus parâmetros, essa premissa pode não se aplicar. Para lidar com essas complexidades, a lógica fuzzy surge como uma alternativa promissora, ao oferecer um arcabouço matemático capaz de representar incertezas, ambiguidade e relações complexas de forma mais natural e interpretável. Sistemas de controle baseados em lógica fuzzy operam com modelos aproximados, dispensando a necessidade de modelagens matemáticas precisas e detalhadas, o que os torna especialmente úteis em domínios onde o conhecimento formal do sistema é incompleto ou difícil de obter.

Barbara (PERNICI; SIADAT, 2011), por exemplo, utiliza lógica fuzzy para adaptar serviços Web sob condições de execução imprevisíveis, considerando o grau de satisfação de requisitos de QoS. Embora tenha empregado funções de pertinência simples para viabilizar a implementação inicial, os resultados experimentais indicaram boa capacidade de adaptação, além de abrirem espaço para futuras melhorias, como o uso de funções mais sofisticadas, como Gaussiana ou Sigmoidal.

Em trabalhos mais recentes, Thiruvalar (THIRUVALAR et al., 2023) propõe uma arquitetura que combina lógica fuzzy com agentes inteligentes para melhorar a experiência do usuário em ambientes de computação ubíqua. Ele utilizou a lógica fuzzy para lidar com incertezas e variações na demanda de serviços, permitindo decisões mais precisas mesmo em cenários altamente dinâmicos. Os resultados apresentados indicaram que a abordagem proposta superou métodos anteriores em

termos de eficiência e adaptabilidade, destacando-se como uma alternativa promissora para ambientes distribuídos.

No contexto de middleware, J. Cao (CAO et al., 2005) propõe um modelo de adaptação de serviços com base em lógica fuzzy, voltado para sistemas de middleware em sistemas móveis sensíveis ao contexto. Nesse modelo, denominado *Fuzzy-based Service Adaptation Model* (FSAM), a lógica fuzzy é empregada para interpretar e processar situações contextuais, como localização, tempo e preferências do usuário, de forma a selecionar dinamicamente a melhor política de execução de serviços. A adaptação ocorre a partir da definição de variáveis linguísticas e graus de pertinência que descrevem o estado atual do sistema, além do uso de funções de aptidão que quantificam o quão bem cada política se ajusta ao contexto presente. A eficácia da proposta é demonstrada no cenário prático da aplicação *Campus Assistant*, reforçando o potencial da abordagem fuzzy em sistemas computacionais que demandam decisões dinâmicas em ambientes incertos.

5.4 Computação evolucionária e lógica fuzzy

O desenvolvimento de controladores fuzzy exige conhecimento especializado sobre o domínio de aplicação. A definição de regras fuzzy e o ajuste das funções de pertinência são tarefas críticas e, frequentemente, manuais. À medida que o número de variáveis e parâmetros cresce, o número de regras necessárias pode aumentar exponencialmente, tornando a tarefa de configuração e manutenção inviável em sistemas complexos.

Para mitigar esse problema, AGs têm sido utilizados como mecanismos de otimização. Os AGs são capazes de automatizar tanto a geração das regras fuzzy quanto a calibração das funções de pertinência, reduzindo significativamente a necessidade de intervenção humana (JEBARI; BOUROUMI; ETTOUHAMI, 2012; KIM et al., 2003). Essas abordagens não apenas eliminam parte da carga de configuração, como também favorecem a adaptação contínua do controlador às mudanças de contexto, garantindo uma maior resiliência do sistema.

Trabalhos como os de Manuel et al. (BRAZ-CéSAR; BARROS; FOLHENTO, 2018), mostram a eficácia da combinação de controladores fuzzy com AGs. Neste trabalho, o controlador fuzzy foi utilizado devido à sua capacidade de lidar com sistemas altamente incertos. Porém, devido às limitações de conhecimento do domínio do problema, foi utilizado um AG para reduzir a necessidade de *expertise* no domínio da aplicação. Por fim, foram comparadas as respostas do sistema contro-

lado e não controlado, obtendo um resultado satisfatório, embora com margem para melhorias.

De forma análoga, Hacide et al. (HACIZADE, 2023) utilizou algoritmos genéticos para corrigir a base de conhecimento de controladores fuzzy. Nesse trabalho, o AG foi utilizado como mecanismo de aprendizado para o controlador fuzzy, adaptando o processo às condições adversas. Os experimentos conduzidos em simulações mostraram que a combinação de controladores fuzzy com algoritmos genéticos forneceu uma capacidade de controle suficientemente boa.

No contexto de agrupamento em clusters, o trabalho de Pytel et al. (PYTEL, 2018) utiliza algoritmos genéticos e lógica fuzzy para auxiliar na resolução de problemas de agrupamento, porém dessa vez não de forma conjunta, mas usando o AG para complementar o trabalho do controlador fuzzy. Os experimentos conduzidos mostraram que essa combinação foi bastante eficaz quando aplicada a esse contexto.

5.5 Contribuições

Neste cenário, o presente trabalho contribui para o estado da arte ao propor uma abordagem que integra controladores fuzzy adaptados por algoritmos genéticos no contexto de middleware orientado a mensagem.

Abordagens baseadas em regras fixas e controle manual são utilizadas na prática; essas abordagens apresentam limitações em cenários dinâmicos. Elas dependem de conhecimento prévio do domínio da aplicação e assumem condições operacionais estáticas. Em contraste, o FALCOM foi projetado para atuar de forma adaptativa, respondendo às alterações do ambiente em tempo de execução.

No grupo de soluções baseados na Teoria do Controle (ROSA; CAVALCANTI, 2022; ROSA; CAVALCANTI, 2023; ROSA; CAVALCANTI, 2024; KUMAR; MUTHUSAMY; BABU, 2012) há uma boa resposta em cenários bem modelados. No entanto, os controladores clássicos, como P e PID, exigem configuração manual dos parâmetros e partem da suposição de linearidade entre as variáveis de entrada e saída. O FALCOM se difere nesse aspecto ao dispensar modelagens matemáticas precisas, adotando controladores fuzzy.

Nos trabalhos que empregam a lógica fuzzy, há a necessidade de conhecimento do domínio da aplicação por parte do projetista do controlador. A proposta aqui desenvolvida visa minimizar as dificuldades inerentes relacionadas ao controle, ao fornecer uma solução adaptativa que busca

também eliminar a necessidade de conhecimento prévio aprofundado do domínio, tornando o controlador reutilizável, com características de operação *plug and play*, através do uso de um AG.

A Tabela 11 resume as principais contribuições do FALCOM em comparação com os grupos de soluções discutidos ao longo deste capítulo.

Tabela 11 – Resumo comparativo entre abordagens de controle de fluxo em MOMs

Aspecto	**Abordagens Heurísticas**	**Teoria do Controle**	**FALCOM**
Mecanismo de controle	Regras fixas e scripts manuais (PIETZUCH; BHOLA, 2003)	Controladores P/-PID baseados em modelos matemáticos (ROSA; CAVALCANTI, 2023)	Controlador fuzzy com ajuste automático via AG
Adaptação em tempo de execução	Inexistente ou limitada	Presente, mas com parâmetros fixos (KUMAR; MUTHUSAMY; BABU, 2012; ROSA; CAVALCANTI, 2024)	Reconfiguração contínua das regras fuzzy
Dependência de conhecimento do domínio	Alta	Moderada (exige modelagem e ajuste fino)	Baixa (dispensa modelagem e reduz necessidade de *tuning*)
Capacidade de lidar com não linearidades	Reduzida	Limitada (assume linearidade)	Elevada (opera em cenários não lineares)

5.6 Considerações Finais

Este capítulo apresentou três grupos diferentes de trabalhos relacionados com o tema da dissertação: teoria do controle aplicada a middleware; lógica fuzzy em sistemas computacionais; e computação evolucionária aplicada à lógica fuzzy. Também foram apresentadas as diferenças do framework FALCOM em relação aos trabalhos apresentados, destacando as contribuições para o avanço do estado da arte.

6 Conclusão e Trabalhos Futuros

Este capítulo apresenta as conclusões e principais contribuições da dissertação, bem como suas limitações. É seguido de potenciais trabalhos futuros na linha de pesquisa.

6.1 Conclusões

Nos últimos anos, observou-se um aumento significativo na adoção de MOMs. Em resposta a essa demanda crescente, novas funcionalidades vêm sendo continuamente incorporadas a esses sistemas, com o objetivo de aprimorar seu desempenho e garantir uma operação eficiente. No entanto, esses sistemas de middleware ainda enfrentam desafios estruturais relevantes, especialmente no que diz respeito ao controle do fluxo de mensagens entregues aos consumidores, o que pode comprometer a estabilidade e a qualidade do serviço.

Esta dissertação usou técnicas da teoria do controle adaptativo, baseadas em lógica fuzzy, para o gerenciamento dinâmico do fluxo de mensagens em sistemas de middleware orientados a mensagens. Nesse contexto, aplicações construídas em cima desses sistemas de middleware podem controlar o fluxo e evitar a inundação de mensagens e, consequentemente, tornar o serviço instável e diminuir a qualidade.

Abordagens tradicionais, como controladores do tipo PID ou mecanismos baseados em limiares fixos, embora eficazes em determinados contextos, apresentam limitações quando submetidas a ambientes com comportamento dinâmico e não linear. Tais limitações incluem a necessidade de sintonização manual, a sensibilidade à variação de carga e a baixa adaptabilidade frente a cenários imprevistos. Nesse cenário, a lógica fuzzy demonstrou-se uma alternativa promissora por permitir a modelagem de incertezas e relações qualitativas de forma interpretável e flexível.

Com base nessas premissas, foi desenvolvido o framework FALCOM, composto por um conjunto de controladores fuzzy configuráveis, capazes de atuar sobre variáveis do middleware em tempo de execução, como a taxa de chegada de mensagens. O framework contempla diferentes métodos de fuzzificação e defuzzificação e integra um algoritmo genético responsável pela geração automática das regras do mecanismo de inferência fuzzy. Essa abordagem buscou ampliar a autonomia e a eficácia do controle, reduzindo a necessidade de intervenção manual e melhorando

a resposta adaptativa do sistema.

Para avaliar a solução, experimentos foram conduzidos em um ambiente do tipo produtor/consumidor, com foco no ajuste dinâmico da taxa de entrega do *broker* ao consumidor, usando um *broker* comercial amplamente usado. Para mensuração do desempenho, foram utilizadas as métricas NRMSE e *goal range* (faixa de tolerância da variável controlada). Essas métricas permitiram verificar, respectivamente, a proximidade da saída em relação ao valor de referência e a estabilidade da resposta sob diferentes padrões de carga.

Os resultados obtidos evidenciaram que os controladores fuzzy foram eficazes na atenuação de picos de carga, na estabilização do tempo de resposta e na manutenção do sistema dentro da faixa de operação desejada. Controladores que utilizaram funções de fuzzificação mais suaves, como a Gaussiana e a *G-Bell*, apresentaram melhor desempenho, especialmente em cenários com variações rápidas na carga. Em contrapartida, métodos como a função Sigmoidal ou em formato de Z mostraram-se menos eficazes, principalmente em situações com comportamento mais instável.

A utilização de algoritmos genéticos para a geração automática das regras fuzzy mostrou-se particularmente eficaz para a redução da necessidade de avaliar quais regras são mais adequadas para as referências utilizadas. A capacidade de reconfiguração dinâmica das regras permitiu também que o sistema mantivesse sua adaptabilidade mesmo em condições de instabilidade ou de mudança nos objetivos de controle.

6.2 Contribuições

A principal contribuição deste trabalho consiste na proposta e implementação do framework FALCOM, concebido com uma arquitetura modular e extensível voltada para o ajuste dinâmico do fluxo de mensagens em MOMs, por meio da aplicação de controladores fuzzy com geração automática de regras. O FALCOM foi projetado para operar de forma adaptativa em tempo de execução, atuando sobre parâmetros do consumidor a partir de sinais coletados do ambiente. Essa atuação é realizada sem a necessidade de modelagens matemáticas precisas, característica que o diferencia de abordagens clássicas baseadas na teoria do controle, como controladores PID.

O FALCOM incorpora, em sua biblioteca, vinte e quatro variações distintas de controladores fuzzy, resultantes da combinação de diferentes funções de fuzzificação e métodos de defuzzificação. Essas variações foram avaliadas sob diferentes condições de carga e tipos de referência, pos-

sibilitando uma análise comparativa abrangente sobre estabilidade, precisão e adaptabilidade das soluções propostas. Os controladores foram integrados ao middleware RabbitMQ, demonstrando a viabilidade da aplicação da abordagem em uma plataforma comercial amplamente utilizada em sistemas distribuídos.

Outra contribuição relevante é a introdução de algoritmos genéticos como mecanismo de ajuste da base de regras dos controladores fuzzy. Essa integração viabilizou a geração automática de regras, reduzindo a necessidade de conhecimento especializado sobre o domínio da aplicação e promovendo uma adaptação contínua do controlador a diferentes contextos operacionais. A aplicação dos AGs também conferiu maior autonomia e resiliência ao sistema, ao permitir a reconfiguração do controlador diante de variações imprevistas nas condições de carga ou desempenho.

Além das contribuições práticas e técnicas, a pesquisa resultou em uma publicação científica.

- Matheus Alves Almeida *and* Nelson Souto Rosa. 2025. *Exploiting Fuzzy Controllers for Adapting Messaging Systems. In Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25). Association for Computing Machinery, New York, NY, USA, 292–299*. https://doi.org/10.1145/3672608.3707930

6.3 Limitações

Apesar dos avanços e resultados promissores alcançados com o framework FALCOM, há algumas limitações no projeto do framework. Em primeiro lugar, os experimentos foram conduzidos em ambiente controlado e simulado, com variação de carga programada e conjunto limitado de métricas, o que pode não capturar toda a complexidade e imprevisibilidade de ambientes produtivos com múltiplos consumidores, concorrência real e degradações imprevisíveis de infraestrutura. Embora o RabbitMQ tenha sido escolhido como plataforma de testes por sua ampla adoção e suporte ao modelo produtor/consumidor, a avaliação se restringiu a esse único MOM comercial, não sendo possível afirmar, com base apenas nos resultados obtidos, que o desempenho observado será replicado de forma semelhante em outras plataformas, como Apache Kafka ou Pulsar (APACHE, 2025a), que possuem arquiteturas distintas.

Outro ponto importante refere-se à natureza dos algoritmos genéticos utilizados. Apesar de eficazes na adaptação das regras fuzzy, os AGs têm um custo computacional considerável, princi-

palmente em fases iniciais de convergência, o que pode impactar a latência ou o desempenho geral do sistema em cenários de variabilidade de carga constante.

A lógica fuzzy também apresenta limitações intrínsecas, como a necessidade de configuração inicial de funções de fuzzificação e da base de conhecimento (ainda que parcialmente mitigadas pelo uso de AGs) e a dificuldade em garantir estabilidade formal em sistemas com múltiplos controladores concorrentes ou em topologias altamente distribuídas. Além disso, a geração de regras foi limitada a um conjunto de sete regras devido a utilização de apenas uma variável de entrada. A lógica fuzzy permite que o controlador trabalhe com múltiplas variáveis de entrada e saída.

6.4 Trabalhos Futuros

A partir dos resultados obtidos, há a possibilidade de continuidade em diferentes aspectos da solução:

- Expandir o framework FALCOM para suportar múltiplos controladores concorrentes e hierárquicos, atuando simultaneamente sobre diferentes variáveis do *broker*, como tamanho de fila e paralelismo de consumidores.
- Integrar abordagens baseadas em aprendizado de máquina, especialmente reforço e aprendizado supervisionado, para refinar a geração de regras fuzzy com base em histórico de desempenho ou comportamento previsto.
- Estender a avaliação experimental para ambientes reais de produção, com múltiplos nós, variabilidade de rede e uso de diferentes MOMs, como Kafka ou Pulsar (APACHE, 2025a), a fim de validar a generalização da abordagem.
- Investigar estratégias de autoajuste da granularidade das funções de pertinência, permitindo adaptação dinâmica não apenas das regras, mas também da base de conhecimento fuzzy.
- Incorporar mecanismos de detecção de anomalias ou falhas, integrando o controlador fuzzy com sistemas de monitoramento para ampliar a resiliência do sistema a falhas sistêmicas ou degradação progressiva de desempenho.

Referências

ABDELZAHER, T.; DIAO, Y.; HELLERSTEIN, J. L.; LU, C.; ZHU, X. Introduction to Control Theory And Its Application to Computing Systems. In: ____. *Performance Modeling and Engineering*. Boston, MA: Springer US, 2008. p. 185–215. ISBN 978-0-387-79361-0.

ALMEIDA, M. A. *Comparative Evaluation of Controllers for Adapting RabbitMQ (In Portuguese)*. [S.l.], 2023.

ALMEIDA, M. A.; ROSA, N. S. Exploiting fuzzy controllers for adapting messaging systems. In: *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2025. (SAC '25), p. 292–299. ISBN 9798400706295. Disponível em: <https://doi.org/10.1145/3672608.3707930>.

AMQP, C. *How to Optimize the RabbitMQ Prefetch Count?* 2023. <https://www.cloudamqp.com/blog/how-to-optimize-the-rabbitmq-prefetch-count.html>. Accessed: September 17, 2024.

AMQP Team. *Advanced Message Queuing Protocol*. 2025. <https://www.amqp.org/about/what>. Acesso em: 23 maio 2025.

Apache. *Apache Kafka*. 2025. <https://kafka.apache.org/>. Acesso em: 23 maio 2025.

APACHE. *Apache Pulsar*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://pulsar.apache.org/docs/4.0.x/concepts-messaging/>.

APACHE. *Apache Thrift*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://thrift.apache.org/>.

Apache. *Kafka Protocol*. 2025. <https://kafka.apache.org/protocol>. Acesso em: 23 maio 2025.

BAI, Y.; WANG, D. Fundamentals of Fuzzy Logic Control — Fuzzy Sets, Fuzzy Rules and Defuzzifications. In: ____. *Advanced Fuzzy Logic Technologies in Industrial Applications*. London: Springer London, 2006. p. 17–36. ISBN 978-1-84628-469-4.

BAO, K.; MAUSER, I.; KOCHANNECK, S.; XU, H.; SCHMECK, H. A microservice architecture for the intranet of things and energy in smart buildings: Research paper. In: *Proceedings of the 1st International Workshop on Mashups of Things and APIs*. New York, NY, USA: Association for Computing Machinery, 2016. (MOTA '16). ISBN 9781450346696. Disponível em: <https://doi.org/10.1145/3007203.3007215>.

BERNSTEIN, P. A. Middleware: a model for distributed system services. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 39, n. 2, p. 86–98, fev. 1996. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/230798.230809>.

BRAZ-CéSAR, M. T.; BARROS, R. C.; FOLHENTO, P. L. P. Fuzzy Controller Optimization Using a Genetic Algorithm for Non-Collocated Semi-Active MR Based Control of a Three-DOF Framed Structure. In: *2018 13th APCA International Conference on Automatic Control and Soft Computing (CONTROLO)*. [S.l.: s.n.], 2018. p. 364–367.

Business Research Insights. *Message Oriented Middleware Market Size, Share and Forecast [2033]*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://www.businessresearchinsights.com/market-reports/message-oriented-middleware-market-116571>.

CAO, J.; XING, N.; CHAN, A. T. S.; FENG, Y.; JIN, B. Service adaptation using fuzzy theory in context-aware mobile computing middleware. In: *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*. [S.l.]: IEEE, 2005. p. 496–501.

CloudAMQP. *Annual RabbitMQ and LavinMQ Report 2024*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://www.cloudamqp.com/blog/annual-amqp-report-2024-by-cloudamqp.html>.

COULOURIS, G. F.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. *Distributed systems: Concepts and design*. [S.l.]: Addison-Wesley, 2012.

DRIANKOV, D.; HELLENDOORN, H.; REINFRANK, M. *An Introduction to Fuzzy Control*. Berlin: Springer, 1993.

Eclipse Mosquitto. *Eclipse Mosquitto*. 2025. <https://mosquitto.org/>. Acesso em: 23 maio 2025.

EMMERICH, W. Software engineering and middleware: A roadmap. 03 2000.

EUGSTER, P. T.; FELBER, P. A.; GUERRAOUI, R.; KERMARREC, A. M. The many faces of publish/subscribe. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 35, n. 2, p. 114–131, jun 2003. ISSN 0360-0300.

FILIERI, A.; MAGGIO, M.; ANGELOPOULOS, K.; D'IPPOLITO, N.; GEROSTATHOPOULOS, I.; HEMPEL, A. B.; HOFFMANN, H.; JAMSHIDI, P.; KALYVIANAKI, E.; KLEIN, C.; KRIKAVA, F.; MISAILOVIC, S.; PAPADOPOULOS, A. V.; RAY, S.; SHARIFLOO, A. M.; SHEVTSOV, S.; UJMA, M.; VOGEL, T. Control Strategies for Self-Adaptive Software Systems. *ACM Trans. Auton. Adapt. Syst.*, ACM, New York, NY, USA, v. 11, n. 4, fev. 2017. ISSN 1556-4665.

GOLDBERG, D. E. *Genetic algorithms in search, optimization, and machine learning*. [S.l.]: Pearson, 2013.

GRPC. *gRPC*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://grpc.io/docs/>.

HACIZADE, U. Development of Adaptive Control System Using Fuzzy–Genetic Approach. In: *2023 5th International Conference on Problems of Cybernetics and Informatics (PCI)*. [S.l.: s.n.], 2023. p. 1–5.

HAUPT, R. L.; HAUPT, S. E. *Practical genetic algorithms Randy L. Haupt, Sue Ellen Haupt*. [S.l.]: Wiley-interscience, 2004.

HOHPE, G.; WOOLF, B. *Enterprise integration patterns designing, building and deploying messaging solutions Gregor Hohpe, Bobby Woolf*. [S.l.]: Addison-Wesley, 2015.

IM, J.; LEE, J.; LEE, S.; KWON, H.-Y. Data pipeline for real-time energy consumption data management and prediction. *Frontiers in Big Data*, Volume 7 - 2024, 2024. ISSN 2624-909X. Disponível em: <https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2024.1308236>.

JANERT, P. K. *Feedback control for computer systems*. Beijing ; Sebastopol, CA: O'Reilly, 2013.

JEBARI, K.; BOUROUMI, A.; ETTOUHAMI, A. Genetic fuzzy rules for DOPs. In: *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*. [S.l.]: ACM, 2012. (GECCO '12), p. 1507–1508. ISBN 9781450311786.

JHA, S. K.; BHASIN, S. Adaptive linear quadratic regulator for continuous-time systems with uncertain dynamics. *IEEE/CAA Journal of Automatica Sinica*, v. 7, n. 3, p. 833–844, 2020. ISSN 2329-9266. Disponível em: <https://www.ieee-jas.net/en/article/doi/10.1109/JAS.2019.1911438>.

JOHAN, K. *Feedback Systems: An Introduction for Scientists and Engineers*. 2nd. ed. S.L.: Princeton University Press, 2021.

JOHANSSON, L. *The Optimal RabbitMQ Guide (CloudAMQP)*. 2023. <https://www.cloudamqp.com/rabbitmq_book_success.html>. Accessed: September 17, 2024.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. *Computer*, IEEE Computer Society Press, Washington, DC, USA, v. 36, n. 1, p. 41–50, jan. 2003. ISSN 0018-9162. Disponível em: <https://doi.org/10.1109/MC.2003.1160055>.

KIM, M. W.; RYU, J. W.; KIM, S.; LEE, J. G. Optimization of fuzzy rules for classification using genetic algorithm. In: *Proceedings of the 7th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. [S.l.]: Springer-Verlag, 2003. (PAKDD '03), p. 363–375. ISBN 3540047603.

Kubernetes. *Horizontal Pod Autoscale*. 2025. <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>. Acesso em: 23 maio 2025.

KUMAR, R. G.; MUTHUSAMY, C.; BABU, A. V. Self-regulating Message Throughput in Enterprise Messaging Servers – A Feedback Control Solution. *International Journal of Advanced Computer Science and Applications*, The Science and Information Organization, v. 3, n. 1, 2012.

LAMBORA, A.; GUPTA, K.; CHOPRA, K. Genetic Algorithm - A Literature Review. In: *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. [S.l.: s.n.], 2019. p. 380–384.

LINGARAJ, H. A Study on Genetic Algorithm and its Applications. *International Journal of Computer Sciences and Engineering*, v. 4, p. 139–143, 10 2016.

MIRJALILI, S. Genetic Algorithm. In: ____. *Evolutionary Algorithms and Neural Networks: Theory and Applications*. Cham: Springer International Publishing, 2019. p. 43–55. ISBN 978-3-319-93025-1.

MQTT. *MQTT*. 2025. <https://mqtt.org/>. Acesso em: 23 maio 2025.

PERNICI, B.; SIADAT, S. H. A Fuzzy Service Adaptation Based on QoS Satisfaction. In: MOURATIDIS, H.; ROLLAND, C. (Ed.). *Advanced Information Systems Engineering*. [S.l.]: Springer Berlin Heidelberg, 2011. p. 48–61. ISBN 978-3-642-21640-4.

PETROWSKI, A.; BEN-HAMIDA, S.; MICHALEWICZ, Z. *Evolutionary algorithms: An overview*. [S.l.]: John Wiley Sons, Incorporated, 2017.

PIETZUCH, P. R.; BHOLA, S. Congestion Control in a Reliable Scalable Message-Oriented Middleware. In: ENDLER, M.; SCHMIDT, D. (Ed.). *Middleware 2003*. [S.l.]: Springer Berlin Heidelberg, 2003. p. 202–221. ISBN 978-3-540-44892-1.

PROPERTIES of Membership Functions, Fuzzification, and Defuzzification. In: FUZZY Logic with Engineering Applications. John Wiley Sons, Ltd, 2010. cap. 4, p. 89–116. ISBN 9781119994374. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119994374.ch4>.

PYTEL, K. Fuzzy-Genetic Approach to Solving Clustering Problem. In: *2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR)*. [S.l.: s.n.], 2018. p. 467–472.

RABBITMQ. *Consumer Prefetch*. 2025. Acesso em: 06 junho 2025. Disponível em: <https://www.rabbitmq.com/docs/consumer-prefetch>.

RabbitMQ. *RabbitMQ*. 2025. <https://www.rabbitmq.com/>. Acesso em: 23 maio 2025.

ROSA, N. S.; CAVALCANTI, D. J. M. A Control-Theoretical Approach to Adapt Message Brokers. In: BAROLLI, L. (Ed.). *Advanced Information Networking and Applications*. Cham: Springer International Publishing, 2023. p. 261–273. ISBN 978-3-031-29056-5.

ROSA, N. S.; CAVALCANTI, D. J. M. Exploiting Controllers to Adapt Message-Oriented Middleware . In: *2024 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. Los Alamitos, CA, USA: IEEE Computer Society, 2024. p. 91–100.

ROSA, N. S.; CAVALCANTI, D. M. Using Controllers to Adapt Messaging Systems: An Initial Experience. In: *Proceedings of the Software Visualization, Evolution and Maintenance Workshop*. [S.l.: s.n.], 2022. p. 46–50.

SABUHI, M.; MAHMOUDI, N.; KHAZAEI, H. Optimizing the Performance of Containerized Cloud Software Systems Using Adaptive PID Controllers. *ACM Trans. Auton. Adapt. Syst.*, ACM, New York, NY, USA, v. 15, n. 3, ago. 2021. ISSN 1556-4665.

SALEHI, P.; ZHANG, K.; JACOBSEN, H.-A. On delivery guarantees in distributed content-based publish/subscribe systems. In: *Proceedings of the 21st International Middleware Conference*. New York, NY, USA: Association for Computing Machinery, 2020. (Middleware '20), p. 61–73. ISBN 9781450381536. Disponível em: <https://doi.org/10.1145/3423211.3426400>.

SCHWENZER, M.; AY, M.; BERGS, T.; ABEL, D. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, v. 117, n. 5-6, 2021. Open Access. Disponível em: <https://doi.org/10.1007/s00170-021-07682-3>.

SHEVTSOV, S.; BEREKMERI, M.; WEYNS, D.; MAGGIO, M. Control-theoretical software adaptation: A systematic literature review. *IEEE Transactions on Software Engineering*, v. 44, n. 8, p. 784–810, 2018.

SOUTO, E.; GUIMARAES, G.; VASCONCELOS, G.; VIEIRA, M.; ROSA, N.; FERRAZ, C. A message-oriented middleware for sensor networks. In: *Proceedings of the 2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing*. New York, NY, USA: Association for Computing Machinery, 2004. (MPAC '04), p. 127–134. ISBN 1581139519. Disponível em: <https://doi.org/10.1145/1028509.1028514>.

SUBRAMONI, H.; MARSH, G.; NARRAVULA, S.; LAI, P.; PANDA, D. K. Design and evaluation of benchmarks for financial applications using advanced message queuing protocol (amqp) over infiniband. In: *2008 Workshop on High Performance Computational Finance*. [S.l.: s.n.], 2008. p. 1–8.

TANENBAUM, A. S.; STEEN, M. v. *Distributed systems: Principles and paradigms*. [S.l.]: Pearson, 2014.

TANOMWONG, N.; JAIKAEO, C. Adaptive middleware for costly data generation over MQTT. In: *2018 5th International Conference on Business and Industrial Research (ICBIR)*. [S.l.: s.n.], 2018. p. 63–68.

THIRUVALAR, V.; R., Y.; DR.M.A.P., M.; SURYASA, I. W.; S., S. Enhancing user experiences in ubiquitous soft computing environments with fuzzy agent middleware. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, v. 14, p. 25–35, 09 2023.

Verified Market Reports. *Message Oriented Middleware Market Size and Forecast 2030*. 2025. Acesso em: 23 maio 2025. Disponível em: <https://www.verifiedmarketreports.com/product/message-oriented-middleware-market/>.

WEYNS, D. Engineering Self-Adaptive Software Systems – An Organized Tour. In: *2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W)*. Los Alamitos, CA, USA: IEEE Computer Society, 2018. p. 1–2.

YANG, F.; THANGARAJAN, A. S.; RAMACHANDRAN, G. S.; JOOSEN, W.; HUGHES, D. AsTAR: Sustainable Energy Harvesting for the Internet of Things through Adaptive Task Scheduling. *ACM Trans. Sen. Netw.*, ACM, v. 18, n. 1, 2021. ISSN 1550-4859.

ZADEH, L. Fuzzy sets. *Information and Control*, v. 8, n. 3, p. 338–353, 1965. ISSN 0019-9958.

ZURADA, J. M.; MARKS, R. J.; ROBINSON, C. J. Context vectors: general purpose approximate meaning representations self-organized from raw data. *Computational Intelligence: Imitating Life, IEEE Press*, p. 43–56, 1994.