Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura Multiagente com Modelos de Raciocínio Distintos para Gerenciamento de Recursos em Múltiplos Provedores de Nuvem

Aldo Henrique Dias Mendes

Tese apresentada como requisito parcial
para conclusão do Doutorado em Informática

Orientadora
Profª. Drª. Célia Ghedini Ralha

Brasília
2024

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Arquitetura Multiagente com Modelos de Raciocínio Distintos para Gerenciamento de Recursos em Múltiplos Provedores de Nuvem

Aldo Henrique Dias Mendes

Tese apresentada como requisito parcial
para conclusão do Doutorado em Informática

Profª. Drª. Célia Ghedini Ralha (Orientadora)
IE/CIC/UnB

Profª. Drª. Lúcia Maria de Assumpção Drummond
IC/UFF

Prof. Dr. Alexandre da Costa Sena
IME/UERJ

Profª. Drª. Aletéia Patrícia Favacho de Araújo
IE/CIC/UnB

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Programa de Pós-Graduação em Informática

Brasília, 02 de fevereiro de 2024

Dedicatória

Dedico esta tese à minha esposa Thaynara Mendes, cuja dedicação e companheirismo me inspiram todos os dias.

Ao meu filho Bento, por me mostrar o maior amor do mundo sem dizer uma palavra ainda.

Aos meus pais, Antônio e Elvira, exemplos de tudo o que um dia almejo ser.

Dedico também esta obra à minha orientadora, Profª. Drª. Célia Ghedini Ralha, pela confiança, paciência, amizade e orientação.

Sem o apoio e inspiração dessas pessoas incríveis, este trabalho jamais teria sido concluído.

Agradecimentos

Agradeço

A Deus, pela vida e pela oportunidade de estudar.

A toda a minha família, pelo apoio incondicional e pelo constante incentivo. Aos meus pais, Antônio e Elvira, por me ensinarem a humildade e dedicação, sendo sempre uma fonte de amor inesgotável. Às tias, tios, primos, primas e todos os outros membros da família: muito obrigado.

À Thaynara Mendes, minha esposa e companheira, por estar sempre ao meu lado durante todo o tempo em que precisei me dedicar a este trabalho, sendo sempre o meu ponto de apoio quando necessário. Obrigado por estar comigo, Meu Amor.

À minha orientadora, Drª. Célia Ghedini Ralha, pela confiança depositada em mim, pelo constante apoio e aconselhamento ao longo de toda a jornada deste projeto. Obrigado por toda orientação ao longo do processo.

Aos amigos da minha Igreja Apostólica, pelos momentos de alegria e pelos louvores que entoamos juntos em cada apresentação. Obrigado por sempre me apoiarem e compreenderem minha ausência nos últimos tempos.

Ao Prof. Me. Anderson Hander, pela revisão ortográfica e gramatical, e revisão crítica desta tese.

À Profa. Drª. Alba Cristina M. A. de Melo pela coordenação do Projeto na Chamada CNPq/AWS No. 32/2019 - Acesso às Plataformas de Computação em Nuvem da AWS.À Profa. Drª. Maria Cristina S. Boeres pela coordenação do Projeto na Chamada Pública CNPq/AWS No. 64/2022 para acesso aos serviços de computação em nuvem da AWS (*Cloud Credits for Research*).

À Profa. Drª. Aleteia e ao Prof. Drº. Marotta, que me orientaram com maestria no direcionamento da minha pesquisa.

À Universidade de Brasília e ao Departamento de Ciência da Computação, que me proporcionaram um ambiente acolhedor e bem estruturado.

A todos que, de forma direta ou indireta, participaram da minha jornada: obrigado, pois nada teria sido possível sem vocês.

"Ama-se mais o que se conquista com esforço."
Benjamin Disraeli

Resumo

As aplicações científicas e comerciais em ambientes de nuvem demandam diversos recursos resultando na necessidade de uma gestão eficiente. A complexidade dos sistemas distribuídos em nuvem com diferentes características, tecnologias e custos variados, torna desafiador o gerenciamento otimizado de recursos. A tecnologia multiagente pode oferecer melhorias significativas para o gerenciamento de recursos, com agentes inteligentes decidindo de forma autônoma os recursos de máquinas virtuais (MVs). Neste trabalho propõe-se o MAS-Cloud+, uma arquitetura multiagente para predição, provisionamento e monitoramento de recursos otimizados em nuvem computacional. MAS-Cloud+ implementa agentes com três modelos de raciocínio, incluindo heurístico, otimizado e meta-heurístico. O MAS-Cloud+ instancia MVs considerando o Acordo de Nível de Serviço (*Service Level Agreement – SLA*) em plataformas de nuvem, priorizando as necessidades do usuário em relação ao tempo de execução, custo e desperdício de recursos, fornecendo seleção apropriada para cargas de trabalho avaliadas. O MAS-Cloud+ foi avaliado com uma aplicação de Bioinformática para comparação de sequências de DNA e diferentes tamanhos de carga de trabalho, executadas na plataforma AWS EC2. Um estudo comparativo com aplicações de *Big Data* usando o *Apache Spark Benchmark* foi executado nas plataformas AWS EC2 e *Google Cloud*. Observa-se por meio dos resultados, que para executar a aplicação de comparação de sequências, o melhor desempenho refere-se ao modelo de otimização, enquanto o heurístico apresenta o melhor custo. Ao fornecer a escolha entre vários modelos de raciocínio, os resultados revelam que o MAS-Cloud+ fornece uma seleção mais econômica das instâncias, reduzindo aproximadamente 58% do custo médio de execução de cargas de trabalho com *WordCount*, *Sort*, e *PageRank*. Quanto ao tempo de execução, o *WordCount* e o *PageRank* apresentam redução de aproximadamente 58%, enquanto o *Sort* um aumento de 532,30%. Os resultados indicam que o MAS-Cloud+ é uma solução promissora para gerenciamento eficiente de recursos em nuvem.

Palavras-chave: otimização combinatória, agentes de raciocínio dedutivo, provisionamento automático de recursos, programação linear, meta-heurística, monitoramento de nuvem, abordagem multiagente

Abstract

Scientific and commercial applications in cloud environments require diverse resources, resulting in the need for efficient management. The complexity of distributed systems in the cloud with different characteristics, technologies, and varied costs makes optimized resource management challenging. Multi-agent technology can offer significant improvements for resource management, with intelligent agents autonomously deciding on virtual machine (VM) resources. In this work, we propose MAS-Cloud+, a multi-agent architecture for the prediction, provisioning, and optimized resource monitoring in the cloud. MAS-Cloud+ implements agents with three reasoning models: heuristic, optimized, and meta-heuristic. MAS-Cloud+ instantiates VMs considering the Service Level Agreement (SLA) on cloud platforms, prioritizing user needs regarding time, cost, and resource waste, and providing appropriate selection for evaluated workloads. MAS-Cloud+ was evaluated with a Bioinformatics application for DNA sequence comparison with different workload sizes, executed on the AWS EC2 platform. A comparative study with Big Data applications using the Apache Spark Benchmark was performed on the AWS EC2 and Google Cloud platforms. The results with the Bioinformatics application indicate that the optimization model performs the best, while the heuristic presents the best cost. By providing a choice among various reasoning models, the results reveal that MAS-Cloud+ allows a more economical selection of instances, reducing the average execution cost of workloads by approximately 58% for WordCount, Sort, and PageRank. Regarding execution time, WordCount and PageRank show a reduction of 58%, while Sort shows an increase of 532.30%. The results indicate that MAS-Cloud+ is a promising solution for efficient resource management in the cloud.

Keywords: Combinatorial optimization, deductive reasoning agents, automatic resource provisioning, linear programming, metaheuristics, cloud monitoring, multi-agent approach

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ACL Linguagem de Comunicação de Agentes - *Agent Communication Language*.

Adapt Agente de Adaptação - *Adaptation Agent*.

AG Algoritmo Genético.

API Interface de Programação de Aplicações - *Application Programming Interface*.

AsQ *Adaptive-Scheduling-with-QoS-Satisfaction*.

AWS *Amazon Web Services*.

BC Base de Conhecimento.

BRMS Gerenciamento de Regras de Negócios - *Business Rules Management System*.

CB Custo Benefício.

CBR Raciocínio Baseado em Casos - *Case-Based Reasoning*.

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico.

CPU Unidade Central de Processamento - *Central Processing Unit*.

CRUD *Create-Read-Update-Delete*.

CSO *Cat Swarm Optimization*.

CSP Provedor de Serviços em Nuvem - *Cloud Service Provider*.

CSV *Comma Separated Values*.

DSTAT Ferramenta para Gerar Estatísticas de Recursos de Sistema.

EC2 *Amazon Elastic Compute Cloud*.

EFA *Elastic Fabric Adapter*.

ERP *Enterprise Resource Planning*.

FA Algoritmo Vagalume - *Firefly Algorithm*.

FF *First-Fit*.

FIFO *First In, First Out*.

FIFO-C *First In First Out Consolidate*.

FIPA Fundação para Agentes Físicos Inteligentes - *Foundation for Intelligent Physical Agents*.

GIO *Greedy Iterative Optimization*.

GPU Unidade de Processamento Gráfico - *Graphics Processing Unit*.

GRASP *Greedy Randomized Adaptive Search Procedures*.

HCSBAT *Hybrid Cat Swarm BAT*.

HPC Computação de Alto Desempenho - *High Performance Computing*.

IA Inteligência Artificial.

IaaS Infraestrutura como Serviço - *Infrastructure as a Service*.

IRNAM *Intelligent Recommender and Negotiation Agent Model*.

JADE Framework de Desenvolvimento de Agentes Java - *Java Agent DEvelopment Framework*.

MADRL-DRA *MultiAgent Deep Reinforcement Learning-Dynamic Resource Allocation*.

MAE Erro Absoluto Médio - *Mean Absolute Error*.

MAS-Cloud+ **M**ulti**A**gent **S**ystem **Cloud Plus**.

MASA *Multi-Platform Architecture for Sequence Aligner*.

MILP Programação Linear Mista Inteira - *Mixed-Integer Linear Programming*.

Mnt Agente de Monitoramento - *Monitoring Agent*.

MSE Erro Quadrático Médio - *Mean Squared Error*.

MV Máquina Virtual.

PaaS Plataforma como Serviço - *Platform as a Service* -.

PL Programação Linear.

PRP-RM-SLA Provisionamento de Recursos Proativos de SLA.

PSO *Particle Swarm Optimization*.

QC Quantidade de CPU na MV.

QM Quantidade de Memória na MV.

QoS Qualidade de Serviço - *Quality of Service*.

QP Questão de Pesquisa.

R^2 Coeficiente de Determinação - *Coefficient of Determination*.

RAM Memória de Acesso Aleatório.

RDMA Acesso Direto à Memória Remota - *Remote Direct Memory Access*.

REST *Representational State Transfer*.

RL Aprendizado por Reforço - *Reinforcement Learning*.

RLM Regressão Linear Múltipla.

SaaS software como Serviço - *Software as a Service*.

SLA Acordo de Nível de Serviço - *Service Level Agreement*.

SMA Sistema Multiagente.

TI Tecnologia da Informação.

vCPU Unidade Central de Processamento Virtual - *Virtual Central Processing Unit*.

VMMgr Gerente de Máquina Virtual - *Virtual Machine Management*.

Sumário

Capítulo 1

Introdução

A expansão acelerada da computação em nuvem é impulsionada com base em três fatores-chave. O primeiro relaciona-se ao mercado estabelecido por grandes corporações de tecnologia, tais como Google, Amazon e Microsoft, cujas pesquisas e serviços têm relevância econômica nesse paradigma. O segundo fator justifica-se pela maturidade dos componentes tecnológicos da nuvem, incluindo hardware, servidores, *cluster*s de alta disponibilidade, computação em grade e tecnologias de virtualização, que formam a base da computação em nuvem. O terceiro fator impulsionador do crescimento acelerado da computação em nuvem consiste na aceitação positiva por parte do público [Foster et al., 2008].

O desenvolvimento e a expansão acelerada da computação em nuvem geram a necessidade de modelos avançados para o gerenciamento de recursos. Do ponto de vista do provedor, são essenciais data *centers* capazes de atender múltiplos usuários com garantia de acesso contínuo, oferecendo serviços sob demanda, mensuráveis e com ajuste dinâmico de recursos [Shen and Chen, 2018]. Nesse modelo de negócio, o cliente paga pelo que consome (*pay-per-use*), e a empresa pode redistribuir sua capacidade computacional entre vários clientes, otimizando a utilização dos recursos. Do ponto de vista do usuário, em relação à demanda deste, a utilização de serviços em nuvem deve satisfazer critérios como desempenho, custo e escalabilidade. Para o usuário, a disponibilidade constante do serviço e uma relação custo-benefício ideal são cruciais. Apesar de haver estudos que visam otimizar recursos na nuvem, há lacunas nas pesquisas, considerando os requisitos tanto do provedor quanto do usuário final [Kaur et al., 2019].

Na computação em nuvem, é crucial utilizar os recursos de maneira eficiente, uma vez que mudanças abruptas no comportamento dos usuários e nas aplicações são frequentes. A gestão dinâmica deve incluir a predição, o provisionamento, o monitoramento e a elasticidade dos recursos alocados nas Máquina Virtuals (MVs), abrangendo CPU, memória, tempo de execução e custo. Isso visa atender às necessidades do usuário, ajustando-se conforme o uso e comportamento das aplicações e serviços. Para tal, os recursos devem

ser alocados conforme o perfil do usuário e de suas aplicações, adaptando-se às suas necessidades de forma transparente, para garantir a Qualidade de Serviço - *Quality of Service* (QoS) estabelecida pelo contrato ou Acordo de Nível de Serviço - *Service Level Agreement* (SLA) [Elloumi et al., 2019].

A possibilidade de utilizar modelos inteligentes para gerenciar recursos em nuvem gera a necessidade de abordagens que permitam a tomada de decisão de forma autônoma e otimizada. Uma maneira de implementar modelos autônomos e distribuídos de tomada de decisão envolve o uso de Sistema Multiagente (SMA) [Wooldridge, 2009]. A interação entre agentes inteligentes e o ambiente de nuvem computacional revela-se um campo promissor para agilizar a execução de funções cruciais para provedores e usuários finais. O uso de agentes inteligentes facilita a aplicação de conceitos de raciocínio da Inteligência Artificial (IA) para alcançar objetivos específicos por meio da interação com entidades em diversos tipos de ambientes [Ferber, 1999].

O ambiente de nuvem computacional caracteriza-se por sua natureza não determinística e dinâmica, devido às aplicações associadas aos serviços e recursos disponíveis em diferentes provedores [Xu et al., 2019]. O emprego de agentes nesse ambiente, com diferentes modelos de raciocínio, viabilizam a tomada de decisão autônoma quanto a otimização de recursos. Torna-se essencial identificar modelos de tomada de decisão que otimizem o tempo de execução e custo, evitando desperdício de recursos [Jouini and Rabai, 2019]. A atuação autônoma dos agentes pode garantir a QoS e cumprir o SLA, com enfoque tanto para o provedor quanto para o usuário final [De la Prieta et al., 2020].

Abordagens baseadas em SMA tornam-se atrativas para possibilitar a adaptação dos sistemas ao ambiente dinâmico. Através do comportamento dos agentes é possível a adaptação dos sistemas ao dinamismo do ambiente, e, por sua vez, ações e simulações de agentes em contextos reais [Weiss, 2013, Wooldridge, 2009]. A utilização de agentes inteligentes apresenta um potencial adequado para a otimização de recursos e o cumprimento dos requisitos de SLA. O emprego de SMA na computação em nuvem tem crescido, pois soluções com agentes inteligentes permitem determinar comportamentos com base na percepção do ambiente e agir para alcançar objetivos pré-estabelecidos [Afrin et al., 2021, Al-Ayyoub et al., 2015, 2016, De la Prieta et al., 2017, Marir et al., 2019, Neto et al., 2019, Ralha et al., 2019, Saif et al., 2022].

Na pesquisa apresentada nesta tese define-se o comportamento dos agentes por meio de diferentes modelos de raciocínio incluindo heurístico, meta-heurístico e otimizado. Por meio do modelo heurístico, empregam-se regras práticas ou soluções aproximadas para direcionar a tomada de decisão dos agentes, com base em conhecimento prévio ou experiência. No meta-heurístico a busca é guiada por soluções sub-ótimas para problemas complexos, combinando diferentes estratégias para superar desafios como mínimos locais

em grandes espaços de busca. O método é considerado genérico podendo ser aplicado a diversos problemas, enquanto o heurístico é definido de forma específica para um problema [Resende and Ribeiro, 2016]. O modelo otimizado baseia-se em técnicas de otimização matemática, como a programação linear, buscando soluções que maximizem ou minimizem critérios específicos, respeitando as restrições e objetivos definidos.

Cada modelo de raciocínio tem vantagens e limitações, e a escolha adequada depende do contexto e dos requisitos específicos do problema. A utilização desses diferentes modelos de raciocínio permite aos agentes flexibilidade e adaptabilidade, garantindo uma tomada de decisão eficiente em ambientes complexos [Arruda et al., 2015, Bădică et al., 2011, Bratman, 1987, da Costa Pereira et al., 2017, Marir et al., 2019, Saadi et al., 2020, Şahin and Alhajj, 2020, Yuan et al., 2021].

1.1 Motivação

Há desafios significativos em relação ao gerenciamento eficiente de recursos em ambientes de nuvem computacional, principalmente, em modelos independentes de provedores. Com o crescente uso da computação em nuvem, torna-se imperativo desenvolver soluções aplicáveis a diferentes provedores, visando a uma gestão de recursos flexível e eficiente Kumar and Umamaheswari [2018], Lahmar and Mezni [2018], Singh and Chana [2016]. Deve-se considerar, em relação aos modelos de gerenciamento de recursos, que diversas aplicações podem exibir comportamentos, custos e desempenhos variados em diferentes provedores, devido a requisitos de configuração de MVs, latência de rede e estratégia de precificação.

Considerando-se o contexto apresentado, concentra-se nesta tese o desafio de elencar recursos adequados de MVs, independente da aplicação, considerando múltiplos provedores de nuvem computacional. Ao contrário das aplicações específicas, que se beneficiam de informações detalhadas para prever recursos, as aplicações genéricas demandam modelos capazes de aprender e se ajustar a uma variedade de perfis de carga de trabalho. Esse aspecto reflete a importância de desenvolver modelos flexíveis, capazes de atender a um espectro variado de necessidades dos usuários.

Os provedores de nuvem oferecem uma ampla gama de opções de MVs, tornando crucial a busca por métodos que otimizem o desempenho, custo e alocação de recursos para aplicações submetidas. A otimização nessa seleção é vital para a eficiência das aplicações, destacando-se a relevância de algoritmos inteligentes e autônomos na escolha das configurações das MVs [Islam et al., 2022, Nadeem et al., 2023].

Considera-se como a principal motivação desta pesquisa, o apoio ao usuário na escolha de MVs, desenvolvimento de uma arquitetura multiagente com diferentes modelos de ra-

ciocínio, permitindo adaptabilidade ao dinamismo do ambiente, garantindo uma tomada de decisão autônoma eficiente para garantir a QoS e cumprir o SLA, independente de aplicação para múltiplos provedores de nuvem. A arquitetura modular que permita a adição de novos componentes é também um desafio que impulsiona este trabalho. A abordagem modular proporciona flexibilidade e escalabilidade, permitindo a incorporação de funcionalidades adicionais e a adaptação a diferentes requisitos e cenários. Dessa forma, torna-se possível a expansão contínua do sistema, acompanhando as mudanças tecnológicas e as necessidades dos usuários. O desenvolvimento de uma arquitetura modular robusta e flexível desempenha um papel fundamental para a superação dos desafios em constante evolução para o gerenciamento de recursos em ambientes de nuvem computacional.

1.2 Problema

No contexto de complexidade do ambiente de nuvem computacional, caracterizado por sua natureza não determinística e dinâmica, visa-se, neste trabalho, abordar o desafio associado às dificuldades do gerenciamento autônomo e otimizado dos recursos de nuvem, independente de aplicação e para múltiplos provedores. Esse gerenciamento, são previstos e monitorados recursos das MVs, incluindo CPU, memória, tempo de execução e custo. É necessário abranger os requisitos do usuário, monitorar, prever e provisionar as MVs, com foco nas demandas de custo e tempo de execução das aplicações.

A revisão da literatura indica que, ainda, há lacunas a respeito desse problema, especialmente considerando que a computação em nuvem está em constante evolução com desafios que emergem regularmente. Com base nesse cenário, define-se como Questão de Pesquisa (QP):

- QP1 - Como desenvolver uma arquitetura que gerencie recursos na computação em nuvem de maneira autônoma e otimizada, considerando as necessidades e preferências do usuário para reduzir o tempo de execução, o custo e o desperdício de recursos?
- QP2 - Quais modelos de tomada de decisão são adequados para gerenciar os recursos na computação em nuvem de forma autônoma e otimizada?
- QP3 - Quais são as métricas adequadas para avaliar a eficiência da proposta arquitetural de gerenciamento de recursos na nuvem, independente de aplicação e para múltiplos provedores, incluindo os modelos de predição e provisionamento de MVs?

Por meio dessas questões de pesquisa, busca-se explorar e resolver os desafios do gerenciamento autônomo e otimizado de recursos na nuvem computacional, considerando

a especificidade do ambiente não determinístico, além das necessidades do usuário e das exigências de custo e tempo de execução das aplicações em múltiplos provedores.

1.3 Objetivos

O objetivo principal deste trabalho consiste no desenvolvimento de uma solução orientada à agentes inteligentes, com diferentes modelos de raciocínio, independentemente de aplicação, para realizar o gerenciamento otimizado de recursos em múltiplos provedores de nuvem. A solução deve possibilitar a escolha autônoma de MVs com foco nas necessidades dos usuários, otimizando o tempo de execução, o custo e o uso de recursos.

Como objetivos específicos deste trabalho pode-se citar:

- implementar e validar a solução através do desenvolvimento de uma arquitetura para o gerenciamento eficiente de recursos em ambientes reais de múltiplos provedores de nuvem computacional;
- realizar experimentos para avaliar a precisão do modelo de predição, incluindo o provisionamento eficiente de MVs, utilizando parâmetros de tempo de execução, custo, e desperdício de recursos, em cumprimento com a SLA definida;
- analisar o desempenho dos modelos de raciocínio (heurístico, meta-heurístico e otimizado), em particular a eficiência de seleção automática do melhor modelo utilizando uma base histórica de aplicações e a influência do tempo de escolha da MV.

1.4 Contribuições

A arquitetura proposta neste trabalho apresenta contribuições que representam áreas de interesse para melhorias e pesquisas futuras.

- arquitetura multiagente para o gerenciamento otimizado de recursos em nuvem, adaptável à diversidade das aplicações, balanceando o custo, o tempo de execução e o desperdício em um contexto de múltiplos provedores de nuvem;
- diferentes modelos de raciocínio para agentes inteligentes, incluindo heurística, otimização formal e meta-heurística, permitindo uma gestão autônoma eficiente dos recursos na nuvem com a seleção de instâncias de MV adequadas, independente de aplicação e para múltiplos provedores;
- uma abordagem para escolher o melhor método de tomada de decisão entre os implementados na arquitetura (i.e., heurístico, meta-heurístico e otimizado). A abor-

dagem baseia-se nas características específicas das cargas de trabalho submetidas pelo usuário, garantindo uma adaptação eficiente aos recursos de MVs disponíveis;

- uma arquitetura modular e extensível que possibilita a integração de novos módulos e serviços, respondendo às demandas emergentes e evoluindo com as necessidades do ambiente de nuvem computacional;
- publicação de um artigo científico na *Future Generation Computer Systems – FGCS*, Elsevier, sendo um prestigiado periódico revisado por pares que começou em 1984 com foco nas áreas de sistemas distribuídos, ambientes colaborativos, computação de alto desempenho, *Big Data* em infraestruturas como redes, nuvem computacional e Internet das Coisas [Mendes et al., 2024]. A *FGCS* está classificada na CAPES no quadriênio de 2017-2020 com estrato Qualis A1 na Ciência da Computação, com fator de impacto (JCR 2022) de 7.5.

1.5 Metodologia

Para atingir os objetivos propostos, classifica-se este estudo como exploratório, experimental, com enfoque de análise quantitativo (dados históricos das aplicações, com escalas de valores numéricos claramente definidas como tempo de execução, custo e desperdício de recursos) e qualitativo (atribuição de valores de qualidade, e.g. taxa de violação de SLA, inferior ou superior ao limite estabelecido) no campo do gerenciamento otimizado de recursos em nuvem computacional, com foco específico no modelo de monitoramento, predição e provisionamento.

Na Figura 1.1, descreve-se a aplicação dessa metodologia em três etapas principais, incluindo estudo dos fundamentos teóricos e revisão da literatura (Etapa 1), desenvolvimento da solução com implementação da solução arquitetural (Etapa 2) e realização de experimentos com análise de resultados (Etapa 3). Pode-se citar como tarefas específicas da metodologia:

- estudar os conceitos-chave de gerenciamento de recursos em nuvem computacional, arquiteturas baseadas em agentes inteligentes e modelos de raciocínio distintos;
- realizar uma revisão sistemática da literatura para mapear o estado da arte, identificar resultados de trabalhos relacionados e posicionar a pesquisa no contexto internacional;
- investigar a aplicabilidade de técnicas de gerenciamento de recursos na computação em nuvem;
- definir, implementar e validar o modelo arquitetural proposto neste trabalho;

- analisar, quantitativa e qualitativamente, os resultados obtidos com o desenvolvimento do modelo em ambiente real de nuvem computacional;
- escrita e submissão de artigo para periódico na área de interesse da pesquisa;
- elaboração da tese, detalhando o trabalho de pesquisa realizado e os resultados obtidos.

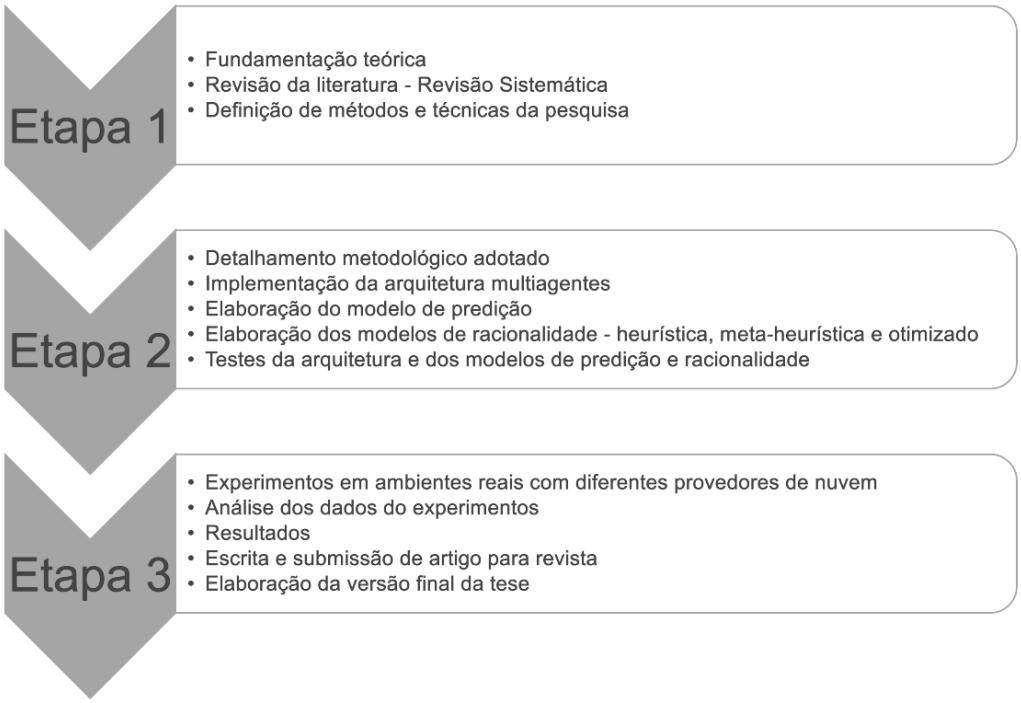

Figura 1.1: Metodologia do trabalho.

1.6 Organização do Documento

A estrutura desta tese está organizada, além deste capítulo introdutório, em mais cinco capítulos. No Capítulo 2, é apresentada a fundamentação teórica, explorando os conceitos principais envolvidos na pesquisa. Discutem-se os fundamentos da computação em nuvem e dos sistemas multiagentes, além das técnicas empregadas nos modelos de raciocínio heurístico, meta-heurístico e otimização combinatória.

No Capítulo 3 realiza-se uma revisão da literatura, abordando os estudos principais relacionados ao uso de computação em nuvem e sistemas multiagentes, bem como os métodos inteligentes empregados para o gerenciamento de recursos. Além disso, examinam-se as pesquisas mais correlatas a este trabalho, fornecendo a base necessária para o desenvolvimento da proposta.

No Capítulo 4 detalha-se a proposta de solução, com a apresentação do modelo conceitual e dos elementos arquiteturais. Descrevem-se os componentes e as funcionalidades da arquitetura, com enfoque no gerenciamento eficiente de recursos em ambientes de nuvem, utilizando um modelo distinto de raciocínio.

No Capítulo 5 apresentam-se os experimentos realizados, descrevendo as metodologias, os cenários de teste e as aplicações utilizadas. Expõem-se os resultados alcançados juntamente às análises efetuadas para avaliar o desempenho da solução proposta.

Por último, no Capítulo 6 discutem-se as conclusões, as limitações e as considerações relativas aos trabalhos futuros. São destacados os objetivos que foram alcançados através dos principais resultados obtidos e as possibilidades de continuidade do trabalho, indicando direções para futuras pesquisas nessa área de estudo.

Capítulo 2

Fundamentação Teórica

Neste capítulo apresentam-se os conceitos de computação em nuvem e Sistema Multiagente (SMA), áreas que apresentam soluções de pesquisa com aplicabilidade ao mercado. Em relação à computação em nuvem, apresentam-se as principais características e serviços básicos, com métodos de predição relacionados com este trabalho. No que se refere ao desenvolvimento de SMA, discutem-se tipos de agentes, coordenação e comunicação entre agentes e modelos de raciocínio, tais como heurístico, meta-heurístico e otimizado.

2.1 Computação em Nuvem

De acordo com Armbrust et al. [2009], a computação em nuvem refere-se a serviços oferecidos por meio da Internet, com hardware e software localizados em *data centers*. No entanto, essa definição pode levar a interpretações relacionadas a outros paradigmas, como a computação distribuída em *grid* computacional [Buyya et al., 2018]. Na literatura, há diferentes definições de computação em nuvem:

- Buyya et al. [2010] descrevem a computação em nuvem como um paradigma de computação que permite o acesso onipresente, conveniente e sob demanda a recursos compartilhados configuráveis (como redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e liberados com o mínimo esforço de gerenciamento ou interação com o provedor de serviços;
- Mell and Grance [2011] definem a computação em nuvem como um modelo que permite acesso *on-demand* a recursos computacionais compartilhados configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser provisionados e liberados rapidamente, com esforço mínimo de gerenciamento ou interação com o provedor de serviços;

- Armbrust et al. [2009] definem a computação em nuvem como um modelo de computação baseado em recursos virtualizados e escaláveis, acessíveis por meio de uma rede, normalmente a Internet. Esses recursos podem incluir armazenamento, processamento, memória e serviços, os quais são fornecidos de forma elástica e sob demanda. A Computação em Nuvem permite que os usuários provisionem, gerenciem e utilizem recursos de forma flexível, pagando, apenas, pelo consumo.

Na literatura de nuvem, um provedor fornece serviços aos clientes, mas estes precisam especificar a qualidade desses serviços em termos de QoS. A definição de QoS envolve atender a requisitos específicos para diferentes serviços, tais como desempenho, disponibilidade, confiabilidade ou custo. Provedores e clientes precisam negociar um SLA que formalmente especifique a QoS conforme os requisitos [Raza and Varol, 2020]. Na Figura 2.1, apresentam-se um esboço do acesso a provedores de computação em nuvem e seus pacotes de serviços.

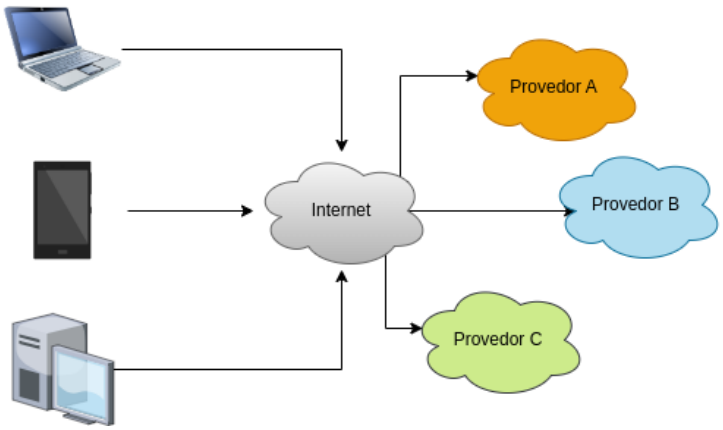

Figura 2.1: Arquitetura de provedores de computação em nuvem.

As definições de computação em nuvem disponíveis na literatura geralmente incluem termos como computação orientada a serviços, virtualização, provisão de recursos sob demanda e elasticidade. Gill and Buyya [2018] afirmam que a computação em nuvem é um modelo que permite o uso e compartilhamento independentemente de recursos, como redes de comunicação, arquiteturas de software e hardware, aplicativos e serviços. Ou seja, é um modelo no qual o usuário pode configurar, contratar e alterar recursos com o mínimo de interação humana.

2.1.1 Modelos de Implantação

Mell and Grance [2009] apresentam quatro modelos de implantação de nuvem computacional. O primeiro modelo é o de nuvem privada, em que se disponibiliza a infraestrutura,

exclusivamente, para o uso de uma organização, gerenciada por esta ou por terceiros. Nesse modelo, estabelecem-se políticas de utilização dos serviços pela organização proprietária.

O segundo modelo é o de nuvem pública, provisionado para uso aberto ao público, podendo ser acessado por qualquer usuário que tenha conhecimento da localização dos serviços. A infraestrutura é de propriedade do provedor de nuvem e disponibilizam-se, amplamente, os serviços.

O terceiro modelo é o de nuvem comunitária, compartilhada por diversas organizações que possuem interesses em comum. A infraestrutura pode ser gerenciada pelas próprias organizações ou por terceiros designados. Nesse modelo, estabelecem-se as políticas de utilização dos serviços juntamente às organizações participantes da comunidade, visando atender às necessidades específicas do grupo.

Por fim, o quarto modelo é o de nuvem híbrida, que consiste na combinação de duas ou mais infraestruturas distintas de nuvem. Esse modelo permite a caracterização de aspectos de nuvem privada, comunitária ou pública, conforme a necessidade e os requisitos da organização. É uma abordagem flexível que possibilita a integração de diferentes tipos de nuvens.

2.1.2 Modelos de Serviços

Também há diversas propostas de arquiteturas de nuvem computacional na literatura [Ghahramani et al., 2017, Tao et al., 2018]. Por meio de uma das abordagens, proposta por Gill and Buyya [2018], identificam-se três grupos de serviços de software: Software como Serviço - *Software as a Service* (SaaS), Plataforma como Serviço - *Platform as a Service* - (PaaS), e Infraestrutura como Serviço - *Infrastructure as a Service* (IaaS), conforme Figura 2.2.

Na arquitetura de nuvem da Figura 2.2, verifica-se três camadas. A camada mais baixa é a camada de infraestrutura, na qual a IaaS fornece recursos físicos, como Unidade Central de Processamento - *Central Processing Unit* (CPU), memória, armazenamento e rede, para a PaaS por meio de MVs e armazenamento. A PaaS, por sua vez, é responsável por fornecer serviços aos desenvolvedores de aplicações via *frameworks* e bibliotecas de desenvolvimento, *Enterprise Resource Planning* (ERP) , banco de dados, entre outros serviços, entregues de forma flexível e escalável. Por fim, os usuários finais têm acesso a aplicações como *e-mail*, ERP e multimídia, disponibilizadas por meio da Internet com base no modelo de consumo.

O uso de arquiteturas de nuvem computacional tem se mostrado uma solução eficiente para as organizações que desejam reduzir os custos de infraestrutura de Tecnologia da Informação (TI), aumentar a flexibilidade e a elasticidade dos serviços, e aprimorar a

eficiência da entrega de serviços de TI. A escolha da arquitetura de nuvem mais adequada para uma organização depende de diversos fatores, como o porte da organização, o tipo de serviço oferecido e os requisitos de segurança e privacidade de dados. Portanto, é fundamental que as organizações realizem uma análise criteriosa de seus requisitos antes de optar pela arquitetura de nuvem mais apropriada.

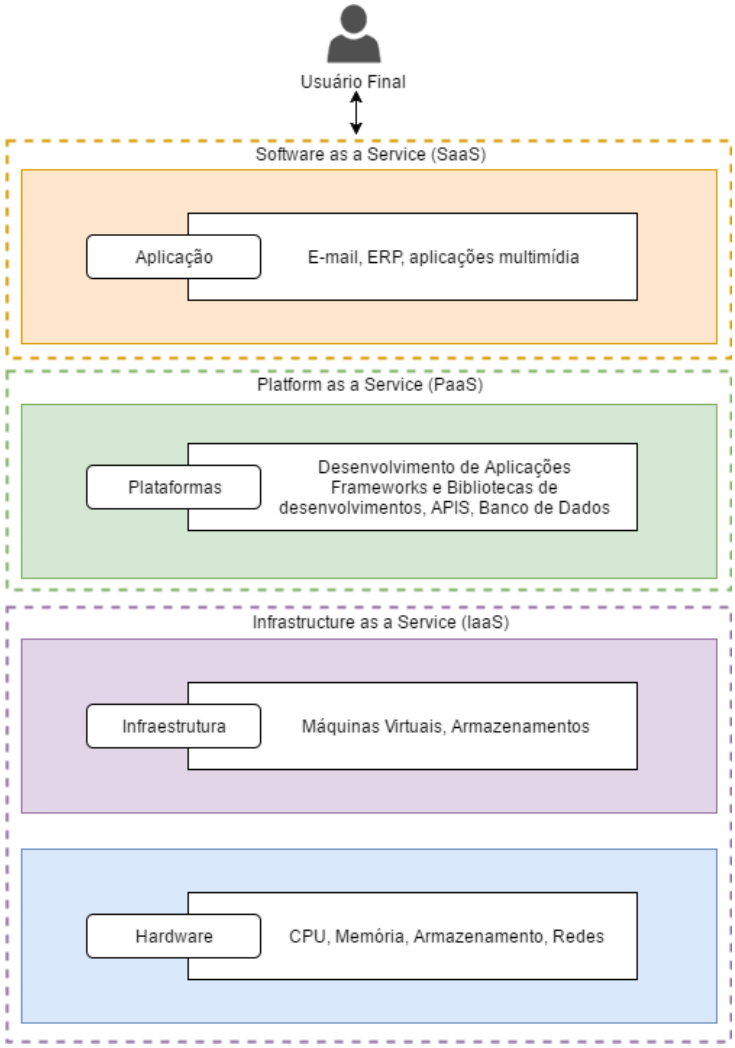

Figura 2.2: Arquitetura de nuvem proposta por Gill and Buyya [2018].

Considerando o foco no usuário, o serviço oferecido pela IaaS inclui o acesso a recursos de infraestrutura de TI virtualizados pela Internet, disponibilizando servidores virtuais, armazenamento, redes e outros componentes disponibilizados e gerenciados pelo provedor de nuvem [Al-Roomi et al., 2013]. Os provedores de nuvem oferecem uma ampla variedade de opções de MVs para atender às diferentes necessidades de desempenho, capacidade e preço dos usuários. Por exemplo, a AWS oferece várias instâncias EC2, incluindo instâncias otimizadas para computação, armazenamento, memória e rede, cada uma com características específicas de desempenho e preço. Da mesma forma, o *Google Cloud* ofe-

rece uma extensa gama de MVs, permitindo que os usuários escolham a configuração que melhor se adapta às suas necessidades [Wankhede et al., 2020].

Neste cenário de amplas opções de provedores e MVs, o desafio reside em selecionar a combinação ideal de recursos e modelo de tarifação para maximizar a eficiência e minimizar os custos, considerando os requisitos de desempenho, disponibilidade e orçamento dos usuários, conforme características das aplicações. Estratégias como otimização de recursos, escalonamento automático e análise de custo-benefício são essenciais para gerenciar efetivamente a infraestrutura em nuvem e garantir um uso eficiente dos recursos disponíveis.

2.1.3 Características Fundamentais

Conforme as definições propostas por Mell and Grance [2009] e Vaquero et al. [2008], a computação em nuvem apresenta características fundamentais que a distinguem de outras concepções de computação. Essas características são: serviços sob demanda, suporte a múltiplas plataformas, medição de serviço, agrupamento de recursos e elasticidade rápida.

A oferta de serviços sob demanda na nuvem permite que os usuários adquiram recursos computacionais, como capacidade de processamento, memória e armazenamento, por meio de catálogos fornecidos pelos provedores, sem a necessidade de interação humana. Dessa forma, os usuários podem solicitar e provisionar recursos de acordo com suas necessidades específicas, permitindo uma flexibilidade e agilidade no uso dos recursos computacionais [Mell and Grance, 2009] .

A computação em nuvem deve ser capaz de oferecer suporte a múltiplas plataformas, o que significa que os serviços e os recursos disponíveis devem ser acessíveis de forma transparente, independentemente da plataforma utilizada pelo usuário. Seja por meio de computadores, *notebooks*, *tablets* ou dispositivos móveis, os serviços em nuvem devem estar disponíveis e acessíveis por meio da Internet, permitindo uma gama de dispositivos e aplicações compatíveis [Ghahramani et al., 2017, Tao et al., 2018].

A medição de serviço desempenha um papel importante para a computação em nuvem, permitindo o monitoramento, controle e relatório do uso dos recursos pelos usuários. Essa característica proporciona transparência no que diz respeito aos serviços prestados, e, por sua vez, o rastreamento e registro de provedores em relação ao consumo de recursos pelos usuários. Além disso, a medição possibilita aos provedores o controle e a otimização do uso dos recursos disponíveis, garantindo a QoS e o comprimento dos SLA.

O agrupamento de recursos é uma característica fundamental da computação em nuvem, em que se organizam os recursos físicos, como servidores, armazenamento e redes, em um *pool* compartilhado. Esse agrupamento permite que os recursos sejam alocados dinamicamente e compartilhados entre múltiplos usuários e aplicações, conforme a demanda

e as necessidades específicas de cada um. Essa abordagem proporciona uma utilização eficiente e otimizada dos recursos, maximizando o aproveitamento da capacidade disponível [Kumar and Umamaheswari, 2018, Remesh Babu and Samuel, 2018].

Por fim, a escalabilidade rápida é uma característica-chave da computação em nuvem, permitindo que os recursos sejam escalados vertical ou horizontal de forma rápida e automática. A escalabilidade vertical refere-se ao aumento ou à diminuição da capacidade de recursos em um único servidor; enquanto a escalabilidade horizontal envolve a adição ou remoção de servidores ao *pool* de recursos. Essa flexibilidade e capacidade de escalabilidade permitem que as aplicações em nuvem atendam às demandas variáveis de recursos de forma ágil e eficiente, garantindo o desempenho e a disponibilidade adequados.

Em suma, a computação em nuvem apresenta um conjunto de características que proporcionam flexibilidade, agilidade e otimização no uso de recursos computacionais. A capacidade de fornecer serviços sob demanda, suporte a múltiplas plataformas, medição de serviço, agrupamento de recursos e escalabilidade rápida são elementos-chave para a expansão desse paradigma computacional [Singh and Chana, 2016, Syed et al., 2017].

2.1.4 Gerenciamento de Recursos

O gerenciamento de recursos em nuvem é essencial para garantir a eficiência e a qualidade de serviço no ambiente com vários consumidores oferecido pelos provedores. Nesse modelo, disponibilizam-se os recursos em um *pool* compartilhado, sendo atribuídos, dinamicamente, aos usuários e tarifados conforme o uso. Os usuários se beneficiam do elevado nível de abstração proporcionado pelos *middlewares*, pois não precisam se preocupar com a localização geográfica dos recursos.

Mell and Grance [2009] subdividem o gerenciamento de recursos em nuvem em serviços de provisionamento, predição e monitoramento, conforme a taxonomia apresentada na Figura 2.3. A provisão de recursos inclui características essenciais como seleção, descoberta, elasticidade, baseada em requisitos de SLA e recursos estáticos. Por meio da predição de recursos, podem-se adotar modelos de aprendizado de máquina ou modelos estatísticos. O monitoramento de recursos inclui aspectos de uso, desempenho e cobrança.

No estudo de Mell and Grance [2009], destacam-se diferentes aspectos e características essenciais do gerenciamento de recursos na computação em nuvem, como:

- provisionamento de recursos envolve a seleção, descoberta, elasticidade com base nos requisitos de SLA e o uso de recursos estáticos. Essas características são fundamentais para garantir o alcance dos recursos e a disponibilização destes conforme as necessidades dos usuários e das aplicações.

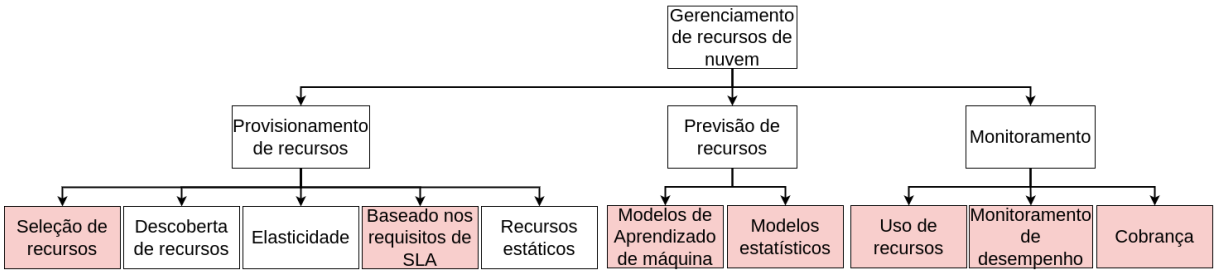

Figura 2.3: Taxonomia de gerenciamento de recursos em nuvem computacional (adaptado de [Kumar and Umamaheswari, 2018]).

- predição de recursos é uma área que pode adotar diferentes abordagens, como modelos de aprendizado de máquina, técnicas de mineração de dados e modelos estatísticos. Utilizam-se essas técnicas para prever a demanda futura por recursos, permitindo que os provedores de nuvem realizem o provisionamento adequado e evitem problemas de sobrecarga ou subutilização de recursos.
- monitoramento de recursos abrange aspectos como o uso dos recursos, o desempenho das aplicações e a cobrança pelos serviços prestados. É essencial acompanhar e registrar o consumo de recursos pelos usuários, garantindo transparência, controle e otimização do uso dos recursos disponíveis.

No contexto do provisionamento de recursos, há cinco características essenciais a serem consideradas:

- seleção de recursos envolve a escolha adequada dos recursos que serão provisionados, considerando as necessidades e os requisitos específicos do usuário ou da aplicação.
- descoberta de recursos refere-se à identificação e a disponibilidade dos recursos no ambiente de nuvem. É importante que os recursos possam ser facilmente encontrados e acessados.
- elasticidade diz respeito à capacidade de dimensionar os recursos de forma dinâmica, aumentando ou diminuindo conforme a demanda. Isso permite uma utilização mais eficiente e adaptável dos recursos.
- baseado nos requisitos de SLA refere-se ao provisionamento dos recursos que deve ser realizado considerando os SLA estabelecidos. Os recursos devem atender aos requisitos de desempenho, disponibilidade e confiabilidade definidos no SLA.
- recursos estáticos são os recursos que possuem características estáticas e não podem ser alterados ou dimensionados dinamicamente. Eles podem ser provisionados conforme a demanda inicial, mas não permitem ajustes durante a execução.

A predição de recursos na nuvem pode ser realizada utilizando diferentes abordagens, tais como:

- modelos de aprendizado de máquina utilizam-se algoritmos para analisar dados históricos e identificar padrões que possam ser utilizados para a predição da demanda futura por recursos.
- modelos estatísticos utilizam técnicas estatísticas para analisar dados históricos e fazer projeções sobre a demanda futura por recursos. Consideram-se, nesses modelos, tendências, sazonalidades e outros fatores relevantes.

O monitoramento de recursos na computação em nuvem abrange diferentes aspectos, tais como:

- uso de recursos que envolve o acompanhamento do consumo de recursos pelos usuários e aplicações. Isso permite a identificação de padrões de uso e o monitoramento do desempenho dos recursos.
- desempenho refere-se à avaliação do desempenho dos recursos, como a capacidade de processamento, velocidade de rede, tempo de resposta, entre outros. O monitoramento do desempenho permite identificar gargalos e otimizar o uso dos recursos.
- cobrança envolve o registro e monitoramento do consumo de recursos pelos usuários, a fim de calcular os custos associados ao uso da nuvem. Isso permite uma cobrança precisa e transparente aos usuários.

A nomenclatura dessa taxonomia é amplamente discutida e adotada por diversos autores na literatura de computação em nuvem, como Kumar and Umamaheswari [2018], Remesh Babu and Samuel [2018], Singh and Chana [2016] e Syed et al. [2017]. Exploram-se, nesses estudos, os desafios e as soluções relacionados ao gerenciamento de recursos em nuvem, considerando aspectos como escalabilidade, otimização do uso dos recursos, predição de demanda e garantia de QoS.

Dada a natureza NP-difícil do problema de escalonamento de recursos em nuvem, é necessário recorrer a abordagens heurísticas e algoritmos de aproximação para encontrar soluções eficientes e próximas à ótima. Projetam-se esses algoritmos para lidar com problemas dessa natureza e fornecer soluções aceitáveis em tempo de processamento razoável. Uma das abordagens utilizadas para o gerenciamento de recursos em nuvem é a predição de recursos, que envolve estimar o uso futuro de recursos com base em padrões e históricos de uso. Isso permite a antecipação das necessidades de recursos e o provisionamento adequado antes que ocorram gargalos ou falha de capacidade. Aplicam-se, frequentemente,

algoritmos de aprendizado de máquina, como redes neurais e modelos de séries temporais, para a predição de recursos em ambientes de nuvem Li et al. [2022].

O provisionamento de recursos refere-se à alocação inicial de recursos para diferentes serviços ou MVs. Envolve a seleção apropriada de recursos com base nos requisitos de desempenho e nas demandas dos aplicativos. Utilizam-se abordagens como algoritmos de otimização, heurísticas e modelos de programação linear para otimizar o provisionamento de recursos em nuvem, considerando restrições e objetivos específicos [Buyya et al., 2008].

Por fim, a alocação de recursos diz respeito à distribuição dinâmica de recursos em tempo de resposta à solicitação, considerando a carga de trabalho atual, as prioridades do sistema e as políticas de gerenciamento. Empregam-se algoritmos de escalonamento, como algoritmos baseados em filas e técnicas de balanceamento de carga, para alocar recursos eficientemente e garantir o desempenho e a disponibilidade dos serviços em nuvem [Youseff et al., 2008].

Neste trabalho, abordam-se os elementos destacados na taxonomia da Figura 2.3, com enfoque para o gerenciamento de recursos, que engloba problemas de monitoramento, predição, e provisionamento. Estas são áreas essenciais para garantir a eficiência na gestão de recursos em ambientes de nuvem computacional.

2.2 Métodos de Predição

A predição consiste em uma técnica amplamente aplicada em diversos domínios, permitindo estimar valores futuros com base em informações históricas e padrões identificados nos dados. Nesta seção, descreve-se o método de predição linear adotado em nuvem computacional, destacando suas características, aplicações e fundamentos teóricos [Hastie et al., 2009, Montgomery et al., 2021].

2.2.1 Regressão Linear

A regressão linear é um método estatístico amplamente utilizado para modelar a relação entre uma variável dependente e uma ou mais variáveis independentes. O método estabelece uma relação linear que melhor se ajuste aos dados observados, por meio da estimativa de coeficientes que quantificam a contribuição das variáveis independentes na predição da variável dependente. A regressão linear é conhecida por sua interpretabilidade, pois os coeficientes estimados fornecem informações sobre a magnitude e direção da influência de cada variável independente na variável dependente. Utiliza-se essa variável a fim de compreender o relacionamento linear entre variáveis e realizar previsões com base nessa relação Montgomery et al. [2021].

No presente trabalho, utiliza-se Regressão Linear Múltipla (RLM) para prever uma variável dependente com base em uma única variável independente, diferentemente da regressão linear simples comumente empregada para esse fim. A principal diferença entre elas refere-se ao número de variáveis independentes consideradas. Enquanto, na regressão linear simples, utiliza-se, apenas, uma variável independente, a RLM permite incorporar várias variáveis independentes simultaneamente no modelo. Isso permite analisar como essas variáveis influenciam, conjuntamente, a variável dependente e controlar seus efeitos individuais, resultando em um modelo mais preciso e capaz de capturar relações complexas entre as variáveis. A escolha da RLM, para este trabalho, é motivada pela necessidade de considerar múltiplas variáveis independentes que podem afetar a variável dependente, visando obter resultados mais precisos conforme a configuração da MV e a aplicação executada Kutner et al. [2004].

A interpretação dos coeficientes de regressão é um aspecto fundamental para a análise de um modelo de regressão linear. Cada coeficiente representa a mudança esperada na variável dependente para uma unidade de mudança na variável independente correspondente, mantendo as outras variáveis independentes constantes. Por exemplo, em um modelo de RLM, considerando-se as variáveis independentes VCPU e a memória de uma MV, os coeficientes estimados para essas variáveis indicarão o efeito esperado na variável dependente (tempo de execução da aplicação) para cada unidade de aumento na VCPU e memória, mantendo os outros fatores constantes. Isso permite uma compreensão mais profunda da influência das variáveis independentes na variável dependente e auxilia a tomada de decisões Kumar et al. [2016].

Regressão Linear Simples pode ser representada conforme a Equação 2.1. Onde y é a variável dependente, x é a variável independente, β_0 é o coeficiente linear, β_1 é o coeficiente de regressão e ε é o termo de erro.

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{2.1}$$

Regressão Linear Múltipla pode ser representada conforme a Equação 2.2. Onde y é a variável dependente, $x_1, x_2, \ldots, x_p$ são as variáveis independentes, β_0 é o coeficiente linear, $\beta_1, \beta_2, \ldots, \beta_p$ são os coeficientes de regressão correspondentes às variáveis independentes e ε é o termo de erro.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon \tag{2.2}$$

2.2.2 Avaliação de Predição

A avaliação de modelos de predição desempenha um papel fundamental para áreas como a Ciência de Dados e Aprendizado de Máquina. Essa etapa é essencial para determinar a confiabilidade dos modelos desenvolvidos e garantir que eles possam fornecer resultados precisos e úteis Raschka and Mirjalili [2019].

De acordo com James et al. [2013], há várias métricas e técnicas utilizadas para avaliar modelos de predição, cada uma com seu próprio propósito e aplicação. Entre as métricas comumente utilizadas, destacam-se:

- Coeficiente de Determinação - *Coefficient of Determination* (R^2) que avalia a proporção da variabilidade dos dados explicada pelo modelo. Esse coeficiente varia de 0 a 1, sendo que valores mais próximos de 1 indicam um bom ajuste do modelo aos dados observados.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.3}$$

  onde $\bar{y}$ é a média dos valores reais y_i, e i representa o índice de cada observação.

- Erro Quadrático Médio - *Mean Squared Error* (MSE) que mede a diferença média entre os valores previstos ($\hat{y}_i$) e os valores reais (y_i). Quanto menor o valor do MSE, melhor é a capacidade do modelo de fazer previsões precisas.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.4}$$

  onde y_i são os valores reais, $\hat{y}_i$ são os valores previstos pelo modelo, n é o número total de observações, e i representa o índice de cada observação.

- Erro Absoluto Médio - *Mean Absolute Error* (MAE) que calcula a média das diferenças absolutas entre os valores previstos ($\hat{y}_i$) e os valores reais (y_i). O MAE fornece uma medida da precisão geral do modelo, independentemente da direção das diferenças.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.5}$$

2.3 Sistema Multiagente

A área de SMA é um campo de pesquisa que combina aspectos de IA e Sistemas Distribuídos, sendo aplicáveis em vários domínios. Um SMA é composto por entidades computacionais, denominadas agentes inteligentes, os quais podem interagir entre si e com o ambiente para alcançar objetivos individuais ou coletivos. Kolp et al. [2006] enfatizam

que os agentes inteligentes são mais que meros componentes de *software*, tendo em vista sua capacidade intencional baseada na habilidade de planejar e negociar.

Segundo Wooldridge [2009], um agente é um sistema computacional capaz de agir de maneira autônoma e flexível em um ambiente, como ilustrado na Figura 2.4, adaptada de Russell and Norvig [2010]. A autonomia dos agentes envolve as características de (i) reatividade — perceber o ambiente e responder a mudanças; (ii) proatividade — decidir por si o curso de ação para atingir seus objetivos; e (iii) sociabilidade — interagir com outros agentes em situações complexas por meio de negociação, cooperação e coordenação.

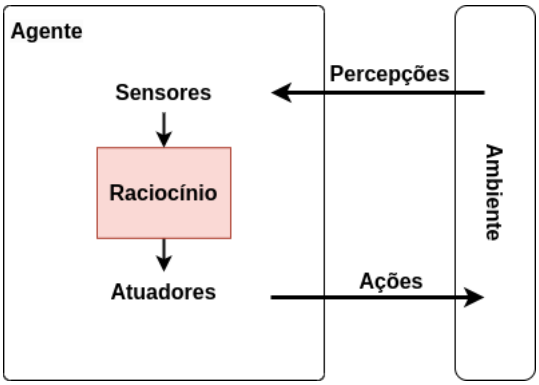

Figura 2.4: Arquitetura básica de um agente inteligente (adaptada de Russell and Norvig [2010]).

O projeto adequado de SMA depende das percepções dos agentes, suas ações, objetivos e do ambiente em que se encontram. Segundo Russell and Norvig [2010], o ambiente pode ser caracterizado conforme as seguintes propriedades:

- completamente observável ou parcialmente observável, quando os sensores do agente conseguem obter informações completas sobre o estado do ambiente, enquanto é parcialmente observável quando há ruído ou imprecisão na percepção;
- determinístico ou estocástico, sendo determinístico quando o próximo estado é determinado pelo estado atual ou pela ação tomada pelo agente, sendo estocástico quando uma mesma ação pode levar a estados diferentes devido a variações do ambiente;
- episódico ou sequencial, quando a interação do agente com o ambiente pode ser dividida em episódios independentes dos anteriores, sendo sequencial quando a decisão atual afeta todas as decisões subsequentes;
- estático ou dinâmico, quando permanece inalterado até que uma ação do agente seja executada, desencadeando mudanças no ambiente. O ambiente é dinâmico quando seus estados podem se modificar enquanto o agente delibera sobre sua próxima ação;

- discreto ou contínuo, quando há um número finito de estados distintos é discreto e contínuo quando não se discerne a quantidade de estados, ações e percepções do agente em relação ao tempo e espaço.

Russell and Norvig [2010], Weiss [2013], Wooldridge [2009] afirmam que os ambientes mais complexos de serem tratados são aqueles que possuem as características de parcialmente observável, estocástico, sequencial, dinâmico, contínuo e multiagente. Além disso, dependendo das características do ambiente, como em problemas reais, pode ser necessário utilizar diversos tipos de agente, com diferentes modelos de raciocínio para alcançar soluções adequadas.

2.3.1 Tipos de Agentes

A capacidade de um agente inteligente é limitada pela sua capacidade de perceber o ambiente, os recursos computacionais disponíveis e o conhecimento adquirido [Sycara, 1998]. Segundo Russell and Norvig [2010], é possível classificar os agentes em cinco categorias de acordo com suas características:

1. agente reativo simples, que mapeia suas ações diretamente por meio da percepção atual do ambiente;
2. agente reativo com registro de estado mantém um estado interno, com base no histórico de percepções e no estado do ambiente, mas ainda age reativamente;
3. agente orientado a objetivos, além das informações de estado, baseia sua decisão atual nos objetivos, que descrevem situações desejáveis, podendo ser objetivos individuais ou coletivos;
4. agente orientado à utilidade, além de considerar os objetivos, considera, também, a utilidade de suas decisões e ações, por meio de métricas que podem reforçar as escolhas conforme a satisfação do agente. Essas métricas podem ser baseadas em modelos estatísticos, teoria de jogos com *payoff* ou ganho em diferentes estratégias dos jogadores;
5. agentes com aprendizado podem atuar em ambientes totalmente desconhecidos e se tornarem mais eficientes à medida que adquirem conhecimento. Os modelos de aprendizado obedecem à classificação da área de IA, podendo ser supervisionado, não supervisionado, semi-supervisionado ou por reforço.

2.3.2 Coordenação de Agentes

A coordenação de agentes é um aspecto fundamental para o funcionamento eficiente de um SMA. Ela deve considerar as características do ambiente onde os agentes operam e os objetivos do sistema como um todo [Boutilier, 1999]. Algumas das características importantes incluem a distribuição de atividades entre os agentes, a descentralização dos dados e o tipo de comunicação utilizado (síncrona ou assíncrona). Essas características explicitam a necessidade de diferentes tipos de coordenação para os componentes do sistema. O objetivo da coordenação é reduzir os conflitos durante a execução das tarefas e permitir que os objetivos individuais ou coletivos dos agentes sejam alcançados [Jennings et al., 1998].

O planejamento de um grupo de agentes pode ser deliberado por um gerente ou por um grupo de agentes. É importante definir o protocolo de coordenação do SMA. No entanto, a definição do tipo de coordenação é um problema complexo, sendo necessário balancear a liberdade dos agentes com sua autonomia deliberativa, permitindo que os objetivos individuais ou coletivos sejam alcançados no processo de resolução do problema [Russell and Norvig, 2010].

2.3.3 Protocolos de Comunicação

Em um SMA, os agentes consistem em entidades independentes capazes de interagir de diferentes maneiras. Faz-se necessário definir protocolos de comunicação e interação para garantir a cooperação e o alcance dos objetivos desejados. A maneira pela qual os agentes comunicam é importante, pois podem impactar diretamente a efetividade das trocas de mensagens e a realização das tarefas prejudicando o alcance dos objetivos.

A Fundação para Agentes Físicos Inteligentes - *Foundation for Intelligent Physical Agents* (FIPA) é uma organização que desenvolve padrões para sistemas baseados em agentes, incluindo padronização de linguagens de comunicação entre os agentes Linguagem de Comunicação de Agentes - *Agent Communication Language* (ACL) Li et al. [2013]. A FIPA apresenta protocolos padrões para a interação ou conversação entre agentes, especificando tipos de performativas de comunicação para as trocas de mensagem na linguagem ACL.

De acordo com Dale and Lyell [2014], as performativas de um protocolo de comunicação FIPA podem ser divididos em quatro categorias:

1. protocolos de perguntas e respostas, que incluem o FIPA-*Query*, utilizado quando o agente deseja fazer uma pergunta e receber apenas uma resposta, e o FIPA-*Subscribe*, usado quando um agente acredita ter a resposta para uma pergunta e subscreve sua resposta.

2. protocolos de solicitação de execução de ações, como o FIPA-*Request* e o FIPA-*Request-When*. Utilizam-se ambos para solicitar aos agentes a execução de uma ou mais ações. No primeiro, a execução deve ser incondicional, e no segundo, condicional.

3. protocolos de negociação para tarefas distribuídas, como o FIPA *Contract Net Interaction Protocol*, que define dois papéis de agentes – o contratante, que anuncia a tarefa a ser executada, recebe ofertas de possíveis contratados (*bids*), contrata e recebe os resultados, e – o contratado, que recebe o anúncio da tarefa, avalia suas capacidades, aceita enviando *bids* ou recusa a tarefa, executa quando contratado e reporta os resultados ao contratante.

4. outros protocolos, como o FIPA-*Inform*, utilizado para informar outros agentes sobre algo, e o FIPA-*Request-When*, utilizado para solicitar que outro agente execute uma ação quando determinada condição for satisfeita.

A escolha do protocolo de comunicação adequado dependerá do tipo de tarefa a ser realizada e dos objetivos do SMA em questão.

A literatura apresenta diversos protocolos mais elaborados para o desenvolvimento de SMA, como o *Blackboard System*, que trabalha com bancos de conhecimento distribuídos mantendo a independência de domínio dos agentes. O protocolo de manutenção de crenças, baseado em sistemas de manutenção de crença, conhecidos como *Truth Maintenance Systems*, que usa estruturas de dados para manter a trilha de fatos em bancos de conhecimento, permitindo um grande nível de interação entre os agentes com derivação e justificação lógica. Também há mecanismos de mercado para a manutenção de equilíbrio competitivo, que aplicam a teoria de jogos da área de Economia. Além disso, existem os protocolos sociais, nos quais há dependências bem definidas entre os agentes (voluntárias, compostas, recíprocas), que devem obedecer a normas e regras de convivência social.

Há diversas ferramentas para o desenvolvimento de SMA, implementadas em diferentes linguagens de programação. Normalmente, os *frameworks* e *middlewares* de desenvolvimento de SMA trazem os protocolos de interação e comunicação já implementados para facilitar o desenvolvimento de sistemas. Uma ferramenta conhecida por seguir estritamente o padrão de especificação FIPA é o (Framework de Desenvolvimento de Agentes Java - *Java Agent DEvelopment Framework* (JADE)) [Bellifemine et al., 2007], elencado para ser utilizado nesta tese. A área de desenvolvimento de SMA tem evoluído bastante em tecnologias, portanto sugere-se a realização de uma revisão sistemática da literatura que inclua linguagens para SMA, conforme proposto por Cardoso and Ferrando [2021].

2.4 Modelos de Raciocínio

Pode-se viabilizar a tomada de decisão dos agentes inteligentes por meio do uso de técnicas de raciocínio. No contexto deste trabalho empregam-se os modelos de raciocínio específicos para otimizar o gerenciamento de recursos em nuvem. Ressalta-se que há outros modelos de raciocínio que poderiam aprimorar o presente trabalho, podendo ser implementados em novos módulos da arquitetura multiagente proposta [Russell, 2016].

Dentre os modelos de raciocínio considerados, destaca-se a ampla aplicação de agentes com aprendizado de máquina em ambientes dinâmicos, utilizando técnicas de *multiagent deep reinforcement learning* [Nguyen et al., 2020]. Além disso, o modelo *Belief-Desire-Intention (BDI)*, baseado no raciocínio humano [Bratman, 1987, Georgeff et al., 1999], é tradicionalmente utilizado. Outra opção de raciocínio para os agentes é o modelo baseado em casos (*Case-Based Reasoning* – CBR), que se baseia em experiências passadas para fundamentar as decisões atuais, com base no conhecimento adquirido [Cervantes et al., 2016].

Um exemplo de modelo de raciocínio bastante utilizado é a inferência baseada em regras de produção lógica, que consiste em um conjunto de regras condicionais que representam conhecimentos específicos sobre um domínio [Arruda et al., 2015, Bădică et al., 2011, Yuan et al., 2021]. Aplicam-se essas regras aos dados de entrada para produzir uma conclusão ou ação.

Em resumo, o uso de diferentes modelos de raciocínio possibilita que os agentes não sejam meramente reativos, ou seja, não reajam somente por reflexos pré-estabelecidos. Em vez disso, viabiliza a construção de agentes orientados a objetivos, utilidade e com aprendizado [Şahin and Alhajj, 2020].

No contexto específico do uso de IA em nuvem computacional, diversos estudos têm explorado essas tecnologias para otimizar o gerenciamento de recursos, melhorar a escalabilidade e a eficiência dos sistemas em nuvem, bem como aprimorar a QoS oferecida aos usuários. Os distintos modelos de raciocínio também têm recebido atenção na literatura. Esses modelos combinam diferentes abordagens de tomada de decisão, como neste trabalho com uso de heurísticas, meta-heurísticas e técnicas de otimização, para melhorar a eficiência dos resultados no processo de deliberação dos agentes inteligentes. Embora ainda haja espaço para pesquisa e aprimoramento, o uso de distintos modelos de raciocínio pode proporcionar maior flexibilidade e adaptabilidade aos sistemas multiagentes em ambientes complexos Zhang et al. [2021].

2.4.1 Heurístico

Uma heurística pode ser definida como uma estratégia ou técnica que busca soluções aproximadas ou satisfatórias para problemas complexos, mesmo sem garantir uma solução ótima. Essas abordagens baseiam-se em conhecimento especializado e experiência para tomar decisões rápidas e eficientes em situações em que a análise de todas as possibilidades não é viável. Amplamente utilizada em diversos domínios, como IA, otimização, planejamento e tomada de decisão, as heurísticas são fundamentais para a identificação e a seleção das regras mais relevantes ou significativas, especialmente para o contexto da inferência baseada em regras Russell and Norvig [2010].

A inferência baseada em regras emprega heurísticas para deduzir conhecimento a partir de um conjunto de regras, associando atributos e condições dos dados de entrada para produzir resultados ou previsões. Essa abordagem tem aplicação em áreas como sistemas especialistas, mineração de dados e aprendizado de máquina, oferecendo interpretabilidade e explicabilidade aos modelos construídos Lakkaraju et al. [2016].

Agentes com raciocínio lógico, baseado em regras de produção, utilizam uma Base de Conhecimento (BC), composta por regras de inferência, que podem ser ampliadas progressivamente. A BC inclui também um conjunto de fatos que refletem a realidade dos estados no ambiente. O motor de inferência, componente essencial para o processamento das regras, é um algoritmo aplicado nas regras de produção da BC e os fatos do ambiente na busca de soluções para os problemas [Russell and Norvig, 2010].

Em uma BC com várias regras, é crucial dispor um motor de inferência eficiente para não prejudicar o tempo de resposta do sistema. O processo de combinação de fatos e regras ocorre ciclicamente, incluindo duas etapas em cada ciclo: avaliação e execução. Na avaliação, o motor de inferência identifica regras ativáveis com base nos fatos atuais. Na execução, disparam-se as regras consequentes, alterando o estado da BC [Russell and Norvig, 2010].

No processo de inferência baseada em regras, implementam-se diferentes etapas por meio de componentes específicos. Um desses componentes, o casamento de regras, é responsável por combinar as regras com os fatos, ativando execuções em uma ordem determinada para resolver possíveis conflitos. A agenda controla a ordem de execução das regras, assegurando que as tarefas ocorram na sequência e prioridade adequadas após a ativação das regras. Por fim, o motor de inferência executa as regras ativas e ordenadas na agenda, possibilitando o encadeamento progressivo (*forward chaining*, começa com sentenças atômicas na BC e aplica regras de inferência *Modus Ponens*) ou regressivo (*backward chaining*, retrocedendo a partir de uma solução para determinar as condições e regras iniciais), que pode ser combinado durante o processo de inferência [Hübner et al., 2011].

Para exemplificar uma heurística no contexto de gerenciamento de recursos, considera-se um cenário em que um agente, com o objetivo de alocar recursos computacionais para diferentes tarefas submetidas por usuários, decide qual MV selecionar de um conjunto disponível em diferentes Provedor de Serviços em Nuvem - *Cloud Service Provider* (CSP). A base de conhecimento e o raciocínio lógico do agente fundamentam-se em parâmetros como tempo de execução, custo, uso de memória e CPU.

A BC do agente contém informações sobre cada MV disponível, incluindo:

- o CSP fornecedor da MV;
- o custo por hora de utilização da MV;
- o desempenho da MV, caracterizado pela capacidade da CPU e quantidade de memória;
- o tempo de resposta ou latência da MV, influenciado por fatores como localização do *datacenter* e carga na MV.

Nesse sentido, uma heurística possível seria: selecione a MV com a melhor relação desempenho a respeito do custo (CPU e memória), desde que o tempo de resposta não ultrapasse um limite pré-definido. Essa regra busca maximizar o desempenho enquanto minimiza o custo, considerando o tempo de resposta para garantir a conclusão das tarefas em um prazo aceitável.

Aplicando raciocínio lógico, o agente utiliza a heurística na BC para cada tarefa submetida:

- filtra as MVs com tempo de resposta no limite aceitável;
- calcula o desempenho (CPU e memória) em relação ao custo para cada MV filtrada;
- escolhe a MV com a melhor relação desempenho/custo.

Esse processo se repete para cada tarefa submetida pelo usuário. A heurística não assegura a escolha da melhor MV para todas as tarefas, mas oferece um método para alocar recursos de maneira satisfatória.

2.4.2 Meta-heurístico

Um modelo meta-heurístico é desenvolvido para resolver problemas, principalmente na área de otimização combinatória. Utiliza-se esse modelo para tratar problemas NP-completos, para os quais não se conhecem algoritmos eficientes [Talbi, 2009].

Uma meta-heurística apresenta estratégias de busca que exploram espaços de soluções em potencial, empregando soluções aleatórias para encontrar soluções locais que possam

conduzir ao ótimo global [Yang, 2010]. A eficiência das estratégias de busca está ligada à organização e definição da solução aleatória no espaço de soluções, tornando importante modelar os procedimentos de aleatoriedade e as buscas locais. Inicia-se a busca local a partir de um ponto aleatório, e, posteriormente, define-se a heurística de construção. A cada iteração do algoritmo, buscam-se soluções similares, procurando a solução ótima [Lourenço et al., 2003].

Neste trabalho, utilizou-se a meta-heurística *Greedy Randomized Adaptive Search Procedures* (GRASP), que consiste na combinação de heurísticas de construção e de busca local [Festa and Resende, 2002]. A heurística utilizada no GRASP é considerada gulosa, porque, conforme as iterações da etapa de construção, selecionam-se as melhores soluções de busca local [Feo and Resende, 1995]. Os Algoritmos 2.1, 2.2 e 2.3 apresentam o GRASP.

Algoritmo 2.1: Meta-heurística GRASP.

```
1   Entrada: Insts, maxIteracoes
2   Retorna: b
3   B == θ
4   F(B) == ∞
5   i == 0
6   enquanto i < maxIteracoes faca
7       b == construcao(Insts)
8       b == buscaLocal(b, Insts)
9       se (F(b) < F(B))
10          B == b
11      i==i+1
12  retorna B
```

Algoritmo 2.2: Etapa de construção.

```
1   Entrada: Insts
2   Retorna: b
3   Insta* == Ordena(Insts)
4   instancia == escolha aleatoriamente uma maquina no Insts*;
5
6   retorna instancia
```

Algoritmo 2.3: Etapa de busca local.

```
1  ***Entrada***: b, construcao
2  ***Retorna***: B
3
4  ***enquanto*** B melhorar ***faca***
5      ***se***(F(B) < F(b))
6          B == b
7  ***retorna*** B
```

Exemplifica-se o uso da meta-heurística GRASP, no contexto de MVs, em um ambiente de computação em nuvem para um problema simplificado, conforme segue.

Problema: O objetivo é alocar um conjunto de tarefas a um conjunto de MVs disponíveis de tal forma a minimizar o custo total, enquanto se satisfazem os requisitos de tempo de execução, uso de memória e de CPU de cada tarefa.

Meta-heurística GRASP:

- Fase Construtiva *Greedy Randomizada*: inicialmente é criado uma lista de MVs candidatos para cada tarefa, ordenada de acordo com uma função de custo-benefício, que pode considerar o custo por hora de uso da MV, tempo de execução estimado, uso de memória e capacidade de CPU. Selecione a MV com melhor custo-benefício de uma lista restrita de candidatos que consiste nas melhores soluções da lista total.
- Busca local: uma vez gerada uma solução inicial, aplica-se uma busca local para refiná-la. Isso pode envolver a troca de MVs entre tarefas, buscando uma melhora na solução corrente.
- Esse processo é repetido até que um critério de parada seja atingido, como um número máximo de iterações ou um limite de tempo de busca do algoritmo.
- Solução final: a solução final será a melhor solução encontrada durante todas as iterações do algoritmo GRASP.

2.4.3 Otimização Combinatória

A otimização combinatória visa encontrar a solução que melhor se aproxima do ótimo dentre um conjunto de opções candidatas, considerando algumas restrições [Vanderplaats, 1984]. A qualidade da resposta é medida pela função objetivo, que deve ser minimizada ou maximizada.

O problema de otimização é representado por $\min / \max f(x)$, sujeito a $x \in P$, onde x é a solução, f é a função objetivo e P é o conjunto de possíveis soluções, determinado

pelas características e restrições de cada problema específico. Ressalta-se que, para uma determinada função de otimização combinatória, pode haver várias soluções, e o conjunto de soluções pode ser contável ou incontável [Xu et al., 2018].

Há diversos algoritmos para resolver problemas de otimização combinatória, dentre os quais destacam-se a busca exaustiva, os algoritmos genéticos, a otimização por enxame de partículas, a otimização por colônia de formigas, a busca tabu, as meta-heurísticas, entre outros [Dai et al., 2017]. Cada algoritmo utiliza uma abordagem específica para encontrar a solução ótima e apresenta vantagens e desvantagens em relação aos demais.

Alguns algoritmos também utilizam técnicas de aprendizado de máquina, como rede neural artificial e árvores de decisão, para encontrar soluções mais precisas e eficientes [Xu et al., 2018]. A escolha do algoritmo adequado para resolver um problema específico depende das características desse problema e do conjunto de soluções possíveis.

Em resumo, a otimização combinatória é uma área importante da pesquisa operacional, que visa encontrar a melhor solução para um problema de escolha dentre um conjunto de opções candidatas, sujeito a restrições específicas. Há diversos algoritmos para resolver problemas de otimização combinatória, que utilizam abordagens específicas e técnicas de aprendizado de máquina para encontrar soluções mais precisas e eficientes.

A programação linear é uma das técnicas mais utilizadas em otimização combinatória. Ela consiste em modelar problemas de otimização como um sistema de equações lineares e desigualdades. Essa técnica tem aplicação em diversas áreas, tais como engenharia, economia, logística [Dantzig and Thapa, 2003]. O problema de programação linear pode ser formulado como:

$$\begin{aligned} \text{Maximizar} \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{sujeito a:} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Nessa formulação a função objetivo é maximizar o produto escalar entre o vetor de custos $\mathbf{c}$ e o vetor de variáveis de decisão $\mathbf{x}$, representado por $\mathbf{c}^\intercal\mathbf{x}$. A matriz $\mathbf{A}$ e o vetor $\mathbf{b}$ representam as restrições do problema, em que $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ indica que todas as combinações lineares de $\mathbf{A}$ e $\mathbf{x}$ devem ser menores ou iguais a $\mathbf{b}$. Além disso, a restrição $\mathbf{x} \geq \mathbf{0}$ indica que todas as variáveis de decisão devem ser não negativas.

Considerando-se a formulação do problema, há diversos algoritmos para a sua resolução. O mais conhecido é o método simplex, que consegue encontrar a solução ótima em um número finito de iterações [Papadimitriou and Steiglitz, 1998].

No entanto, em alguns casos, o número de variáveis e restrições é tão elevado que o método simplex torna-se impraticável. Nessas situações, pode-se associar a programação

linear com outras técnicas de otimização combinatória, tais como programação inteira, para encontrar soluções aproximadas para o problema. Além disso, pode-se utilizar a programação linear como um componente-chave em outras técnicas de otimização combinatória, como a Análise de Redes e a Teoria dos Jogos [Rahmanian et al., 2014].

Para exemplificar a aplicação da otimização combinatória, especificamente a Programação Linear (PL), para a seleção de MVs em um ambiente de computação em nuvem, considera-se a seguinte problemática: duas tarefas a serem alocadas e duas MVs disponíveis. O objetivo é minimizar o custo total.

Em relação à base de dados, suponha que é conhecido o custo por hora de cada MV e o tempo de execução necessário para cada tarefa nas MVs. Suponha ainda, que o custa da MV1 é de R$ 5,00 por hora e da MV2 de R$ 7,00 por hora. A Tarefa 1 demanda duração de 1 hora para execução em ambas as MVs, e a Tarefa 2 requer 2 horas nas MVs. Podem-se definir, nesse sentido, as seguintes variáveis:

- x_{ij}: variável binária que é igual a 1 se a tarefa i é alocada à MV j, e 0 caso contrário.
- c_j: custo por hora de uso da MV j.
- t_i: tempo de execução da tarefa i.

O problema pode ser formulado da seguinte maneira:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{2} \sum_{j=1}^{2} c_j \cdot t_i \cdot x_{ij} \\ \text{sujeito a:} \quad & \\ & \sum_{j=1}^{2} x_{ij} = 1 \quad \forall i \\ & x_{ij} \in \{0, 1\} \quad \forall i, j \end{aligned}$$

Nesse caso, a função objetivo é minimizar o custo total das MVs para executar todas as tarefas. A restrição garante que cada tarefa seja alocada a exatamente uma MV. Esse é um problema de PL que pode ser resolvido facilmente, mas serve para ilustrar o conceito. Na prática, o problema pode ser mais complexo, envolvendo várias outras restrições e parâmetros.

Capítulo 3

Revisão da Literatura

Neste capítulo, apresenta-se o resultado da revisão de literatura com uso de um protocolo de revisão sistemática. O objetivo principal da revisão sistemática foi identificar na literatura a existência de estudos primários que proponham métodos para o gerenciamento de recursos em nuvem computacional, preferencialmente, englobando modelos de predição, monitoramento, e provisionamento dinâmico de recursos. Procurou-se também analisar as abordagens do estado da arte encontradas na literatura com relação ao desenvolvimento e à aplicação de soluções eficientes, otimizadas e inteligentes para gerenciamento de recursos em nuvem.

3.1 Revisão Sistemática

O protocolo proposto por Kitchenham [2004] foi utilizado na revisão sistemática de literatura, que compreende três fases principais: o planejamento, a execução e o relato da revisão.

Planejamento

Na Tabela 3.1, apresentam-se as principais características do protocolo adotado na fase de planejamento. Realizou-se a revisão em 2022, e atualizou-se em 2023, considerando-se a análise de artigos científicos publicados em periódicos e conferências da área de Computação, bem como capítulos de livros, referentes a soluções para o gerenciamento de recursos em nuvem computacional. Incluíram-se trabalhos publicados em língua inglesa nos últimos nove anos, de 2015 a 2023.

Tabela 3.1: Características do protocolo da revisão sistemática realizada.

Realização/Revisão	2022 e 2023
Período das publicações	2015 a 2023
Tipo de publicação	artigos científicos publicados em periódicos/conferências e capítulos de livros
Idioma	inglês
Ferramenta	*Parsifal*
Repositório digital	*ACM Digital Library, IEEE Digital Library, ScienceDirect*

Para auxiliar a realização da revisão, utilizou-se uma ferramenta denominada *Parsifal*[1]. As buscas foram realizadas nos seguintes repositórios digitais:

- *ACM Digital Library* (`https://dl.acm.org/`),
- *IEEE Digital Library* utilizando *IEEEXplore* (`https://ieeexplore.ieee.org/Xplore/home.jsp`), e
- *ScienceDirect* (`https://www.sciencedirect.com/`).

Na Tabela 3.2, apresentam-se as palavras-chave utilizadas nas buscas especificadas por repositório digital.

Tabela 3.2: Palavras-chave utilizadas nos repositórios digitais.

Repositório	Palavras-chave
ACM Digital Library IEEE Digital Library	(“Cloud computing" OR “Cloud platforms" OR “Computer architecture" OR “Virtual machining" OR “virtualization" OR “Resource scheduling" OR “IaaS Cloud") AND (“Resource Balancing" OR “distributed processing" OR “Resource management" OR “Dynamic scheduling" OR “dynamic resource allocation" OR “resource allocation" OR “Machine Learning" OR “MV cost" OR “Metrification of MV" OR “Elasticity" OR “Predictive" OR “Reactive") AND (“Monitoring" OR “supervision" OR “monitoring" OR “measure" OR “go along" OR “inspect" OR “monitor" OR “observe" OR “supervise" OR “Efficient use of resources")
ScienceDirect	“Cloud computing"and “Monitoring"and “Resource management"and “dynamic provisioning"and “resource allocation"

Ainda na fase de planejamento, a fim de orientar a formulação das questões de pesquisa da revisão sistemática, há um consenso sobre a estruturação dos componentes do acrônimo PICO, em que cada letra representa um componente, conforme apresentado na

[1]Parsifal é uma ferramenta online desenvolvida para auxiliar pesquisadores a realizarem revisão sistemática de literatura, `https://parsif.al/`, (acessado em 17 de abril de 2020)

Tabela 3.3. **P** representa a população de interesse, **I** representa a intervenção de interesse, **C** representa a comparação com outras revisões, *surveys* e/ou *reviews*, e **O** representa os resultados ou *outcomes* [Methley et al., 2014].

Tabela 3.3: PICO definido na revisão sistemática.

✓	População	Soluções (estratégias, métodos, abordagens, heurísticas e/ou arquiteturas) para o gerenciamento e balanceamento de recursos em nuvem computacional
✓	Intervenção	Soluções com foco no monitoramento, predição, provisionamento e elasticidade de recursos
	Comparação	Comparar esta revisão sistemática com outras revisões sistemáticas, *surveys* e/ou *reviews*
✓	Resultado	Oportunidades, propostas e resultados das estratégias, métodos e soluções para o tratamento de recursos em nuvem computacional

Desta forma, a questão de pesquisa definida para a revisão sistemática foi:

> Quais as melhores estratégias, métodos, soluções, abordagens, heurísticas e/ou arquiteturas utilizadas para o gerenciamento e/ou balanceamento de recursos em nuvem computacional?

Execução e Relato da Revisão

Na execução da revisão de literatura foram definidos critérios de inclusão e exclusão dos trabalhos pesquisados. Como critérios de inclusão cita-se:

- devem ser trabalhos publicados e disponíveis integralmente nos repositórios digitais da *ACM Digital Library, IEEE Digital Library, ScienceDirect.*
- devem ser trabalhos recentes (publicados entre 2018-2023), publicados em periódicos/conferências reconhecidos e avaliados pela comunidade científica da área de Ciência da Computação.
- os trabalhos devem apresentar estratégias, métodos, soluções, abordagens, heurísticas e/ou arquiteturas utilizadas para o gerenciamento e/ou balanceamento de recursos em nuvem computacional.

Como critérios de exclusão cita-se:

- serão desconsiderados trabalhos que não detalhem experimentos práticos realizados para testar as suas hipóteses.
- trabalhos duplicados nos repositórios digitais da *ACM Digital Library, IEEE Digital Library, ScienceDirect.*

Na execução da revisão sistemática encontrou-se um total de 905 trabalhos, conforme a Tabela 3.4. Esses trabalhos apresentaram incidência nas bases da *ScienceDirect* (39.3%), na *IEEE Digital Library* (36%) e na *ACM Digital Library* (24.5%), conforme a Figura 3.1.

Tabela 3.4: Relação de trabalhos pesquisados e aceitos.

Repositório	**Pesquisados**	**Aceitos**
ScienceDirect	356	130
IEEE Digital Library	326	74
ACM Digital Library	223	75
Total	905	279

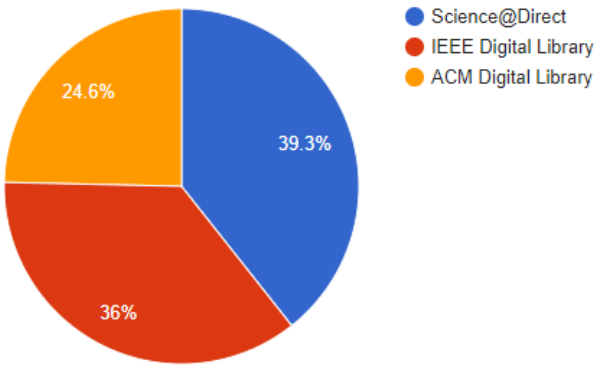

Figura 3.1: Trabalhos pesquisados classificados por repositório digital.

Para classificar e avaliar os trabalhos encontrados, definiram-se alguns critérios de qualidade. Cada pergunta permitiu três respostas com ponderações diferentes: peso 1 para SIM, peso $0,5$ para PARCIALMENTE e peso 0 para NÃO. Formularam-se quatro perguntas para avaliação de qualidade:

1. Há descrição arquitetural da proposta?
2. O trabalho foi validado em ambiente real de nuvem computacional?
3. Trata-se de um modelo automatizado?
4. Os experimentos são passíveis de replicação?

A relação dos trabalhos aceitos por repositório digital, após a aplicação do critério de qualidade descrito, com uma nota de corte variando de 3.5 a 4, totalizou 279, conforme a Tabela 3.4 e a Figura 3.2. Ao comparar os trabalhos aceitos, percebe-se que a maior incidência ocorreu no ano de 2018, com um total de mais de 200 artigos.

Na Figura 3.3, ilustra-se o processo adotado para a realização da revisão da literatura. A primeira etapa consiste em realizar uma busca abrangendo três repositórios digitais

(*ACM Digital Library, IEEE Digital Library, ScienceDirect*). Combinam-se os resultados e, posteriormente, há um processo de seleção, iniciado com uma pré-filtragem dos estudos, e refinado continuamente. Esse processo é dividido em três passos. No primeiro passo (Passo 1), selecionam-se os trabalhos com base na leitura dos resumos. No segundo passo (Passo 2), realiza-se a seleção dos textos restantes com base na introdução e conclusão dos trabalhos. Leem-se os textos restantes no terceiro passo (Passo 3) e, em caso de respondida alguma pergunta de pesquisa, registram-se as informações relevantes. Após a seleção, realiza-se um processo de *snowballing*, no qual leem-se as referências relevantes, encontradas nos estudos externos, com suas informações extraídas. Os 78 trabalhos resultantes da revisão de literatura, apresentado nesta figura, encontram-se listados na Tabela A.1 do Apêndice A.

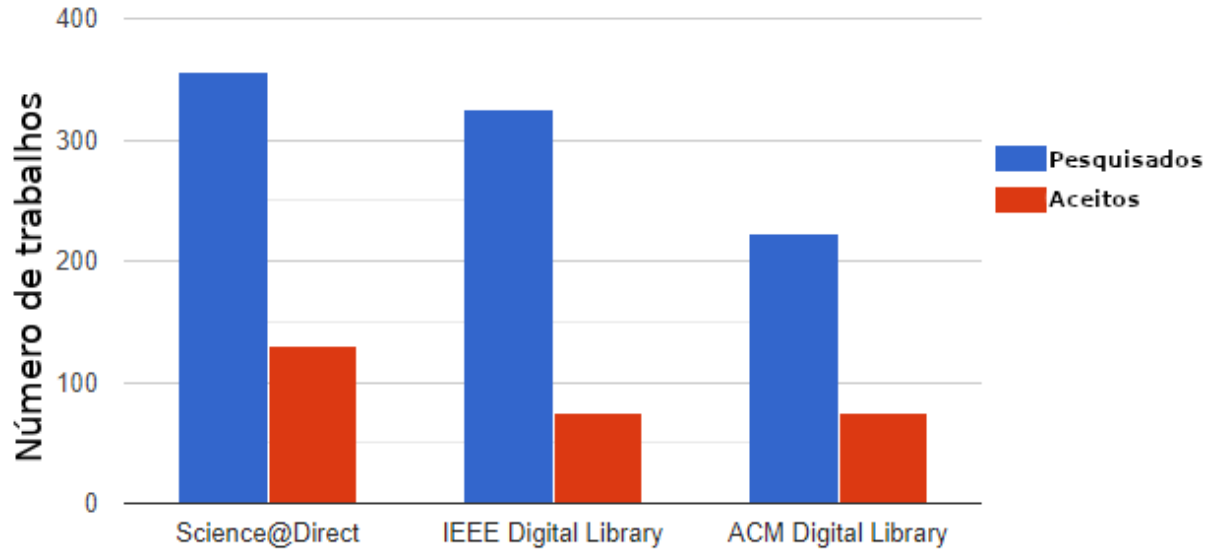

Figura 3.2: Trabalhos pesquisados e aceitos por repositório digital.

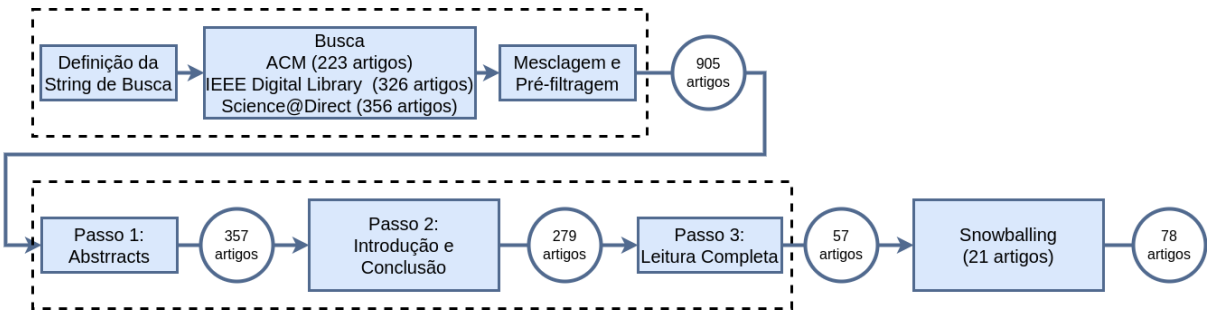

Figura 3.3: Método da execução da revisão de literatura.

Analisaram-se os trabalhos aceitos na revisão sistemática para verificar quais apresentavam maior relação com esta pesquisa. Nesse sentido, selecionaram-se sete trabalhos no período de 2018 a 2022. Posteriormente, incluiu-se trabalhos mais recentes publicados em 2023, totalizando dez trabalhos, conforme apresentado na Tabela 3.5 da Seção 3.2.

3.2 Trabalhos Correlatos

Nesta seção são apresentados os dez trabalhos mais relacionados a esta pesquisa após a realização da revisão sistemática da literatura. Na Tabela 3.5, comparam-se os trabalhos apresentados ao **M**ulti**A**gent **S**ystem **Cloud Plus** (MAS-Cloud+), considerando o modelo de predição, a independência dos provedores de nuvem computacional, as abordagens baseadas em SLA/QoS, o uso de agentes inteligentes com um modelo de provisionamento baseado em abordagens de raciocínio distintas (i.e., heurística, otimizado, meta-heurística), e o processo de validação em ambiente real ou simulado. Na sequência será apresentado um breve relato de cada trabalho.

Tabela 3.5: comparação com trabalhos relacionados

Referência	Modelo de Predição	Independente de Provedor	Baseado em SLA/QoS	Agentes Inteligentes	Plataforma Real	Modelo de Provisionamento		
						Heurístico	Otimização	Meta-heurístico
Funika et al. [2023]	✓	✓	✓	✓	✓	✓	✓	–
Kumar et al. [2023b]	–	✓	✓	✓	–	–	–	–
Kumar et al. [2023a]	–	–	✓	–	–	–	✓	–
Saif et al. [2022]	✓	–	✓	✓	–	✓	✓	–
Osypanka and Nawrocki [2022]	✓	–	✓	✓	–	✓	✓	–
Islam et al. [2022]	–	✓	✓	–	✓	✓	–	–
Asghari et al. [2020]	✓	–	–	✓	–	–	✓	✓
De la Prieta et al. [2020]	–	–	✓	✓	✓	✓	–	–
Jyoti and Shrimali [2020]	–	✓	✓	✓	–	✓	–	–
Guo and Shenoy [2018]	✓	–	✓	–	✓	✓	✓	–
MAS-Cloud+	✓	✓	✓	✓	✓	✓	✓	✓

Funika et al. [2023] propõem uma arquitetura para um sistema automatizado de provisionamento de recursos em nuvem usando otimização, mais especificamente o algoritmo de *Proximal Policy Optimization* (PPO) Schulman et al. [2017]. A proposta visa utilizar os avanços recentes em Aprendizado por Reforço - *Reinforcement Learning* (RL) para otimizar o custo de execução de um processo evolutivo intensivo em computação, automatizando a escalabilidade de recursos heterogêneos em um ambiente de nuvem. Para avaliar a arquitetura, os autores incluem experimentos com o gerenciamento autônomo de uma carga de trabalho de processo evolutivo executada no ambiente *Open AI Gym Cloud-Sim+* para treinar o sistema com um agente utilizando raciocínio baseado em RL.

É capturado o estado do ambiente para ajustar a política de controle para recursos heterogêneos. Compara-se o desempenho com a abordagem tradicional de gerenciamento, baseada em limiares automáticos, apresentando vantagens de provisionamento de recursos otimizados. Os autores fornecem detalhes do treinamento da política de gerenciamento usando o algoritmo de otimização PPO.

Kumar et al. [2023b] apresentam um *framework* denominado *Intelligent Recommender and Negotiation Agent Model* (IRNAM) utilizando agente inteligente com três fases para a recomendação e a negociação de serviços em nuvem. É utilizado um algoritmo de concessão analítica que analisa as escolhas do usuário e do CSP para encontrar um servidor confiável e seguro. É considerado a satisfação de QoS do usuário e do provedor antes de finalizar o SLA. As três fases incluem recomendação, negociação e agregação. Com o algoritmo de recomendação, um CSP adequado é recomendado. Na fase de negociação, consideram-se atributos prioritários para a negociação usando um algoritmo até que o valor agregado do usuário seja satisfeito. Na fase de agregação, analisa-se o grau de satisfação do usuário e do CSP. Se a oferta for aceita, estabelece-se o contrato de SLA e operacionalizam-se os serviços em nuvem. O IRNAM foi avaliado por simulação. Os autores concluíram que, em relação ao emprego da metodologia de agentes, os CSPs podem atender às demandas de usuários que requerem a entrega priorizada de serviço, a melhora da eficiência geral e o cumprimento do processo de entrega de serviços.

Kumar et al. [2023a] apresentam um novo algoritmo de gerenciamento de recursos em um ambiente de computação em nuvem denominado *Hybrid Cat Swarm BAT* (HCSBAT). O problema abordado é a alocação eficiente de recursos para cargas de trabalho, visando maximizar a utilização, e consequentemente, os lucros dos CSPs. Para isso, utilizaram-se meta-heurísticas como Algoritmo Genético (AG), *Cat Swarm Optimization* (CSO), *Particle Swarm Optimization* (PSO), Algoritmo Vagalume - *Firefly Algorithm* (FA) e o próprio Bat Yang [2014]. A solução proposta consiste na implementação do algoritmo, que se mostrou superior em termos de disponibilidade, escalabilidade e *throughput*, comparados ao AG, CSO e Bat. As vantagens da solução proposta incluíram a melhoria na utilização dos recursos e a maximização dos lucros dos CSPs. As desvantagens incluíram a complexidade do modelo e a necessidade de ajustes constantes para garantir sua eficiência.

Saif et al. [2022] buscam lidar com grandes demandas de recursos flutuantes, garantindo o funcionamento das operações comerciais em contêineres em várias nuvens. É proposto um *framework* de provisionamento de recursos com agentes para garantir a execução efetiva de processos comerciais em um ambiente de múltiplas nuvens com garantia de QoS. A solução permite o provisionamento elástico de recursos para resolver problemas de super e sub-provisionamento, atendendo aos objetivos do CSP e dos usuários, considerando QoS e SLA. Os autores utilizam rede neural artificial para prever as demandas

futuras de recursos, melhorando o desempenho e garantindo o provisionamento elástico de recursos. Agrupam-se os recursos para executar as cargas de trabalho intensivas em CPU e em I/O usando o algoritmo *K-means*. O trabalho é validado usando a plataforma *Cloudsim*, o que torna difícil a comparação com ambientes reais de nuvem pública.

O trabalho de Osypanka and Nawrocki [2022] aborda o provisionamento dinâmico de recursos em ambientes de nuvem para otimizar o uso dos recursos e reduzir os custos. A solução proposta utiliza algoritmos de aprendizado de máquina para prever a demanda futura e realizar o provisionamento dinâmico. Utilizam-se quatro modelos de predição: *Bayesian Linear*, *Decision Forest Regression*, *Boosted Decision Tree Regression* e *Neural Network Regression*. Empregam-se agentes inteligentes para gerenciar as MVs. Consideram-se todos os tipos de componentes (PaaS, IaaS e SaaS). As vantagens incluem maior eficiência no uso dos recursos, redução dos custos associados ao provisionamento inadequado e maior flexibilidade na escolha do provedor de nuvem. A validação usa testes *time-compressed*, que demonstram uma melhoria significativa na eficiência ao longo das execuções. No entanto, é necessário tempo para coletar dados históricos e ajustar periodicamente os modelos para garantir a precisão. Resultados demonstram uma redução de custos de recursos em 85% ao longo de 10 meses.

O problema abordado em Islam et al. [2022] refere-se ao agendamento de tarefas em um ambiente de nuvem híbrida. A solução visa minimizar violações de prazo das tarefas e o custo monetário de uso de MVs em *clusters*. Propõe-se um modelo de otimização para o agendamento baseado em SLA de tarefas *Spark*, incluindo dois algoritmos de agendamento de trabalhos. O primeiro é uma versão modificada da heurística *First-Fit* (FF) para resolver problemas de empacotamento de caixas. O segundo algoritmo, denominado *Greedy Iterative Optimization* (GIO), utiliza uma abordagem gulosa para encontrar o custo ótimo para cada tarefa na nuvem. A implementação utiliza a nuvem privada *Nectar* com o pacote *BigDataBench benchmarking* nos experimentos. Apesar da solução proposta não possuir como critérios de agendamento o desperdício de recursos, foi utilizado como um estudo comparativo neste trabalho (vide Seção 5.3).

O trabalho de Asghari et al. [2020] aborda o problema de otimização do consumo de energia em sistemas de computação em nuvem. A solução proposta é um algoritmo híbrido que combina técnicas de busca local e AGs para encontrar uma programação ótima que minimize o consumo de energia e o tempo de execução das tarefas. As vantagens da solução incluem a capacidade de reduzir os custos operacionais e melhorar a eficiência energética, enquanto as desvantagens incluem a complexidade computacional do algoritmo proposto. Os autores discutem estratégias adicionais, como monitoramento, provisionamento e gerenciamento dinâmico de MVs, usadas em conjunto com o algoritmo híbrido para melhorar a eficiência energética dos sistemas em nuvem. Os resultados revelaram

que o algoritmo proposto alcançou uma redução média no consumo de energia em 30% e uma redução média no tempo total de execução das tarefas em 25%. A solução apresenta espaço para melhorias futuras na complexidade computacional e na escalabilidade do modelo.

O trabalho de De la Prieta et al. [2020] utiliza abordagem baseada em agentes inteligentes para alocação de recursos em nuvem computacional. A solução utiliza SLA e QoS. A execução do sistema ocorreu em uma plataforma de nuvem privada, não sendo independente de provedor.[2] Os autores não apresentam o modelo de predição para o provisionamento de recursos de nuvem. O modelo de tomada de decisão dos agentes é Raciocínio Baseado em Casos - *Case-Based Reasoning* (CBR), exigindo experiências anteriores, ou seja, casos ocorridos no ambiente anteriormente e armazenados para produzir melhores resultados em cada alocação de recursos.

O problema apresentado por Jyoti and Shrimali [2020] relaciona-se à demanda do usuário quando as solicitações devem ser alocadas em MVs e os recursos distribuídos entre provedores. É apresentado a arquitetura *MultiAgent Deep Reinforcement Learning-Dynamic Resource Allocation* (MADRL-DRA) para recriar as solicitações dos clientes, prevendo os valores de recursos desperdiçados com base na demanda do cliente. A avaliação é simulada e inclui vários *data centers* e MVs. Cada classe de agente na arquitetura tem objetivos diferentes, como alocação de recursos e agendamento de tarefas. Para trabalhos futuros, os autores sugerem o estudo de monitoramento de recursos simulados, o uso de dados históricos para alocação de recursos e o agendamento de tarefas.

O problema abordado por Guo and Shenoy [2018] consiste na distribuição geográfica suportada pela eficiência da geo-elasticidade, a fim de atender às necessidades de aplicações que apresentam variações temporais e espaciais. Apresentam-se dois tipos de provisionamento: proativo e reativo. O provisionamento utiliza previsões por meio de um banco de dados histórico para orientar o algoritmo. O *GeoScale* monitora e gerencia MVs em várias nuvens por meio de *logs* de aplicativos, com informações de tempo de serviço, tempo de resposta com carimbos de data e hora. O modelo de predição de recursos com base em dados históricos não é detalhado. O *GeoScale* emprega o provisionamento reativo para fazer alterações no provisionamento de capacidade a longo prazo pelo algoritmo proativo — para lidar com aumentos inesperados na carga de trabalho ou corrigir erros nas previsões. Como otimização para o provisionamento geo-elástico é usado: pré-copiar a MV e escolher dinamicamente, pares de data centers para pré-cópia com base na largura de banda disponível. Resultados experimentais mostram uma melhora de 13% a 40% no tempo de resposta em comparação com outras técnicas de elasticidade.

[2]https://bisite.usal.es/

A arquitetura proposta nesta tese apresenta vantagens em relação aos trabalhos correlatos mencionados. Uma vantagem é a utilização de diferentes modelos de predição de tempo de execução e uso de recursos, RLM, contribuindo para melhorar o uso de agentes inteligentes para o gerenciamento otimizado de recursos, permitindo uma implementação extensível a novos módulos e serviços. A arquitetura utiliza modelos distintos de raciocínio nos agentes, aumentando a capacidade de adaptação a diversos contextos de nuvem. No entanto, assim como os trabalhos correlatos, a arquitetura também possui aspectos que podem ser aprimorados, como a complexidade computacional e a necessidade de histórico das aplicações para garantir a precisão do modelo de predição.

Capítulo 4

Proposta de Solução

Neste capítulo são apresentadas as respostas para as duas questões de pesquisa descritas na Seção 1.2:

- QP1 – Como desenvolver uma arquitetura que gerencie recursos na computação em nuvem de maneira autônoma e otimizada, considerando as necessidades e preferências do usuário para reduzir o tempo de execução, o custo e o desperdício de recursos?
- QP2 – Quais modelos de tomada de decisão são adequados para gerenciar os recursos na computação em nuvem de forma autônoma e otimizada?

Desta forma, o conteúdo deste capítulo inclui: na Seção 4.1, o modelo conceitual com as definições relacionadas aos agentes e a caracterização do ambiente de computação em nuvem; na Seção 4.2, o modelo arquitetural, incluindo aspectos de software; na Seção 4.3, os modelos de raciocínio para tomada de decisão dos agentes; na Seção 4.4, apresenta-se a escolha automática dos modelos de raciocínio, incluindo predição e provisionamento; e na Seção 4.5, os aspectos tecnológicos do desenvolvimento da solução.

Em relação ao desenvolvimento da arquitetura para o gerenciamento de recursos na nuvem computacional, referente à QP1, a arquitetura multiagente desenvolvida neste trabalho foi denominada MAS-Cloud+. O principal objetivo da arquitetura consiste na minimização do tempo de execução e custo, independente da aplicação, por meio da seleção autônoma da melhor configuração de MV, considerando as preferências do usuário, incluindo tempo de execução e custo, em múltiplos provedores. Os serviços de gerenciamento de recursos disponíveis no MAS-Cloud+ são detalhados nas seções de monitoramento (Seção 4.2.1), de predição (Seção 4.2.2), e de provisionamento de MVs (Seção 4.2.3).

Ainda considerando o desenvolvimento da arquitetura com distintos modelos de raciocínio dos agentes, referente à QP2, os modelos propostos para a gestão otimizada de

recursos das MVs são apresentados na Seção 4.3, incluindo heurístico (Seção 4.3.1), otimizado (Seção 4.3.2) e meta-heurístico (Seção 4.3.3).

4.1 Modelo Conceitual

A abordagem baseada em agentes no gerenciamento de recursos em nuvem aborda a complexidade dos problemas envolvidos na alocação e provisionamento de recursos. Problemas como a atribuição eficiente de recursos e o balanceamento de carga são desafiadores do ponto de vista computacional [Buyya et al., 2008]. Podem-se classificar alguns desses problemas como NP-difíceis. Tais problemas requerem aplicar heurísticas, meta-heurísticas e algoritmos aproximados para encontrar soluções em tempo satisfatório [Youseff et al., 2008].

O MAS-Cloud+ visa lidar com esses desafios, integrando uma arquitetura multiagente que permite a coordenação e a comunicação entre os agentes responsáveis pelo gerenciamento de recursos em nuvem. Essa abordagem estratégica busca otimizar a alocação e o provisionamento de recursos, considerando as demandas de necessidades do usuário e as restrições dos recursos disponíveis.

Um elemento central do modelo conceitual é a capacidade de prever e provisionar, eficientemente, recursos com base nas demandas do usuário. Esse aspecto crucial do *design* do sistema permite um uso eficiente dos recursos, garantindo que as MVs sejam alocadas conforme a necessidade das cargas de trabalho. Outro conceito fundamental refere-se ao foco na modularidade e na extensibilidade. Projetou-se o sistema para acomodar e integrar novos modelos, agentes, serviços e provedores, permitindo que este evolua e se adapte às mudanças nas demandas do usuário e às tendências emergentes da nuvem.

Os conceitos fundamentais que norteiam a arquitetura do MAS-Cloud+ envolvem abordagens estratégicas para criar uma arquitetura para o gerenciamento de recursos de MVs em computação em nuvem. Por uma série de componentes interconectados, trabalhando sinergicamente, o sistema visa proporcionar flexibilidade, controle e extensividade.

No *core* do MAS-Cloud+, há uma estrutura multiagente, em que cada agente desempenha um papel específico para o funcionamento global do sistema. Os agentes são elementos-chave que facilitam a operacionalidade e a manutenção de um ambiente robusto. Cada agente tem uma função designada para o gerenciamento de recursos na nuvem, incluindo monitoramento, predição e provisionamento de MVs.

Na ideia conceitual do projeto, a independência do provedor é um princípio-chave no modelo. Por meio da comunicação através da Interface de Programação de Aplicações - *Application Programming Interface* (API), o sistema poderá operar com diferentes prove-

dores de serviços de nuvem, proporcionando flexibilidade e permitindo que o sistema seja adaptado a diferentes contextos e requisitos.

Nesse projeto a eficiência da solução é definida pela habilidade da arquitetura em gerenciar recursos em múltiplos provedores de nuvem de forma otimizada, minimizando o tempo de execução, custo e desperdício de recursos. Isso inclui a seleção automática e eficiente da MV, bem como o cumprimento efetivo da SLA.

Representando o fluxo de trabalho do projeto, na Figura 4.1, ilustra-se o processo executado pelo MAS-Cloud+. A execução do processo inicia-se com a submissão de uma aplicação pelo usuário, sendo esta a carga de trabalho considerada nas etapas de tomada de decisão subsequentes. Em seguida, realiza-se a predição do tempo de execução e dos recursos necessários para a execução da carga de trabalho. Posteriormente, ocorre o provisionamento da MV, permitindo a alocação de recursos no provedor. Realiza-se a execução da aplicação no provedor (por exemplo, *Amazon Web Services* (AWS) *Amazon Elastic Compute Cloud* (EC2)), seguida pelo início do monitoramento da MV. Nas seções subsequentes detalha-se cada módulo da arquitetura do MAS-Cloud+.

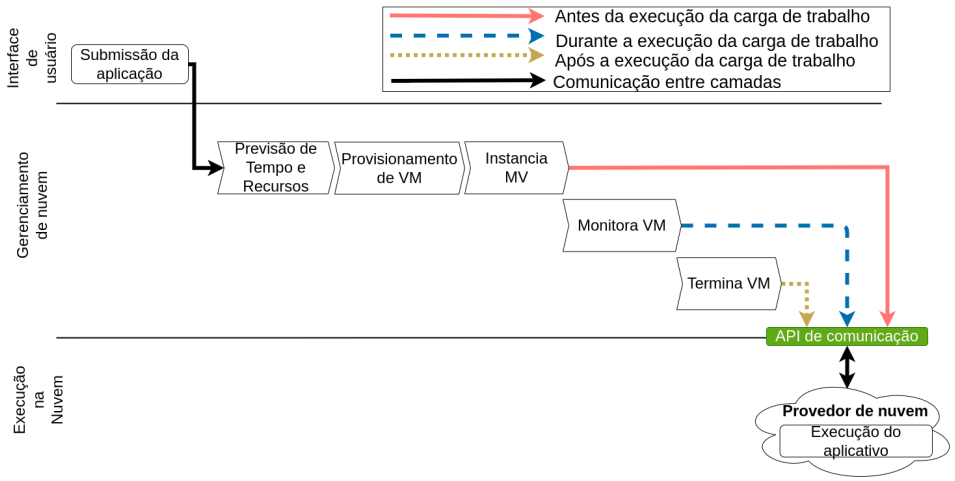

Figura 4.1: Fluxo de execução do MAS-Cloud+.

4.2 Arquitetura MAS-Cloud+

A Figura 4.2 ilustra a arquitetura do MAS-Cloud+, composta por três camadas distintas. Na camada inferior, a execução da carga de trabalho ocorre na MV do provedor de nuvem. A camada intermediária abriga o gerenciamento de recursos em um servidor Web via API *Representational State Transfer* (REST). Nessa camada, estão implementados os agentes responsáveis pelo gerenciamento de recursos na nuvem, incluindo monitoramento, predição e provisionamento de MVs, além das bases de armazenamento e de conhecimento.

A camada superior engloba a interface do usuário, acessível por dispositivos pessoais que proporcionam acesso ao sistema pelo consumidor de nuvem.

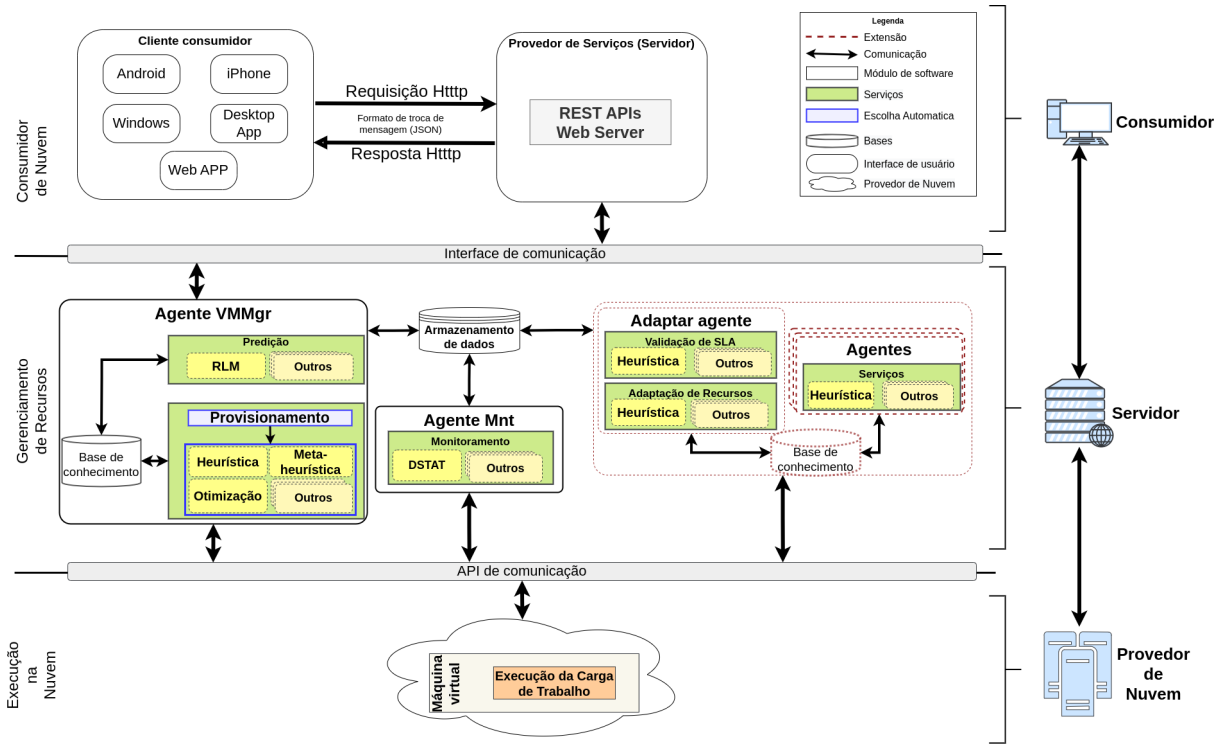

Figura 4.2: Arquitetura do MAS-Cloud+ com os três níveis e seus respectivos agentes, serviços e comunicação.

O nível de execução inclui MVs executando cargas de trabalho submetidas pelo usuário que interagem com o *host* do provedor. Podem-se incorporar provedores de nuvem privados ou públicos (e.g., AWS e *Google Cloud*) para ter MVs instanciadas.

A respeito do nível de gerenciamento, incluem-se serviços para predição, provisionamento, monitoramento, validação de SLA e adaptação de recursos (da esquerda para a direita). Implementam-se os serviços pelos agentes de Gerente de Máquina Virtual - *Virtual Machine Management* (VMMgr), Agente de Monitoramento - *Monitoring Agent* (Mnt) e Agente de Adaptação - *Adaptation Agent* (Adapt). As informações de execução da aplicação são armazenadas em arquivos *Comma Separated Values* (CSV) para a tomada de decisão dos agentes.

O nível de interface do usuário fornece a interação com o MAS-Cloud+ por meio da submissão de cargas de trabalho usando uma API REST, permitindo independência de provedor. A arquitetura é modular e extensível, viabilizando a absorção de novos modelos de raciocínio, agentes, serviços e provedores, conforme marcado com linhas pontilhadas vermelhas. Um módulo de extensão futura no agente Adapt pode fornecer validação de SLA em tempo de execução usando métodos proativos, como no *framework* de Provisi-

onamento de Recursos Proativos de SLA (PRP-RM-SLA), proposto por Nadeem et al. [2023].

A arquitetura MAS-Cloud+ implementa a API utilizando os métodos apresentados na Tabela 4.1, conforme os padrões *REST* definidos por Fielding et al. [1999], Fielding [2000]. Implementam-se os métodos por meio das operações *Create-Read-Update-Delete* (CRUD), incluindo *CREATE* ⇒ *POST*, *READ* ⇒ *GET*, *UPDATE* ⇒ *PUT* e *DELETE* ⇒ *DELETE*, para lidar com os recursos das MVs. Incluem-se as seguintes variáveis utilizadas como entrada para os métodos: *UserId*, constitui o identificador único do usuário que acessa o sistema; *CloudProvider*, que indica o provedor de nuvem selecionado para a instanciação da MVs; *DecisionModel*, que determina o modelo de tomada de decisão utilizado para o provisionamento, que pode ser otimizado, heurística ou meta-heurística; *TimeVariable* representa a prioridade com base em restrições de tempo de execução; *ResourceUsageVariable* reflete a prioridade percentual de uso de recursos, como uso de CPU e memória; *PriceVariable* representa o custo financeiro da execução; *SLATime*, *SLAResource* e *SLAPrice* se referem às garantias de nível de serviço em relação ao tempo de execução, recursos e preço, respectivamente; *OSimage* consiste no sistema operacional selecionado para a instância da MV; *SubmitionId* é a identificação da aplicação submetida pelo usuário, servindo para identificar a base de dados com as informações da aplicação; com o *ApplicationCategory*, o usuário pode informar a categoria da aplicação; *UpdateStatusVM* atualiza o status da MV; e *Command* recebe os comandos de entrada do usuário executados nas MVs [1].

Na Tabela 4.1 são apresentados os métodos da API. No método *GET /RequestVM*, quando uma MV já foi criada para o usuário, o acesso à MVserá restaurado. No método POST /RequestVM, se a MVfor criada com sucesso, o usuário terá acesso a esta. No método *PUT /RequestVM*, o usuário pode atualizar o status da MV, e suspender a execução da MV. No método *DELETE /RequestVM*, o usuário pode atualizar o *status* da MV, podendo deletar a execução da MV. No método POST */commandsTerminal*, se o usuário estiver logado e a MVestiver instanciada, o comando enviado como entrada será executado na MV. Em retorno, o resultado da execução será entregue ao usuário Fielding et al. [1999], Fielding [2000].

Conforme apresentado na Tabela 4.2, o MAS-Cloud+ utiliza a variável de entrada *ApplicationCategory* para identificar a categoria da aplicação e selecionar as configurações iniciais de CPU e memória, com base nos limites máximos e mínimos. É importante observar que, para garantir o processo de categorização, as informações relativas às categorias das aplicações são armazenadas junto aos agentes em uma base de dados CSV comparti-

[1] A API MAS-Cloud+ está disponível em `https://github.com/aldo2102/mas-cloud-plus-API`

lhada entre os usuários. Desta forma, o MAS-Cloud+ mantém atualizada e acessível às categorias de aplicações com suas respectivas configurações.

Tabela 4.1: Métodos da API.

API	Rota	Método	Entrada	Saída	Descrição
Gerenciamento de MV	*/RequestVM*	*GET*	- *UserId*	Se sucesso - Acesso à VM Senão - Mensagem de erro	Caso uma MVjá tenha sido criada para o usuário, o acesso à MVserá restaurado.
	/RequestVM	POST	- *CloudProvider* - *DecisionModel* - *TimeVariable* - *ResourceUsageVariable* - *PriceVariable* - *SLATime* - *SLAResource* - *SLAPrice* - *OSimage* - *SubmitionId* - *ApplicationCategory* - *UserId*	Se sucesso - Acesso à VM Senão - Mensagem de erro	Se a MVfor criada com sucesso, o usuário terá acesso à mesma.
	/RequestVM	PUT	- *UpdateStatusVM* - UserId	Se MVexistir - Suspender VM Senão - Mensagem de erro	O usuário pode atualizar o status da MV, sendo capaz de suspender a execução da VM.
	/RequestVM	DELETE	- *UpdateStatusVM* - *UserId*	Se MVexistir - Excluir VM Senão - Mensagem de erro	O usuário pode atualizar o status da MV, sendo capaz de excluir a execução da VM.
Acesso à MV	*/commandsTerminal*	POST	- *Command* - *UserId*	Se sucesso - Saída do Terminal Senão - Mensagem de erro	Se o usuário estiver logado e a MV estiver instanciada, o comando enviado como entrada será executado na VM. Ao retornar, o resultado da execução será entregue.

Tabela 4.2: Categorização de aplicações e limites de recursos ([Gupta et al., 2019, Joe, 2022, KarlErickson, 2022, Rautiainen and Marschall, 2020, Singh et al., 2020]).

ID	**Aplicação**	**CPU (Min-Max)**	**Memória (Min-Max)**
1	Serviço de *e-commerce*	4 - 8	16 - 32
2	Alinhamento de sequência de DNA	8 - 16	32 - 64
3	Intensivo em CPU	16 - 32	16 - 32
4	Intensivo em memória	4 - 8	64 - 128
5	Sites da Web	2 - 4	8 - 16
6	Computação tecnológica	8 - 16	32 - 64
7	*Software Endeavour*	4 - 8	16 - 32
8	Teste de *performance*	8 - 16	32 - 64
9	Processamento de transações *online*	8 - 16	32 - 64
10	Serviços financeiros centrais	8 - 16	32 - 64
11	Serviços de armazenamento e *backup*	4 - 8	16 - 32
12	Aplicações de produtividade	2 - 4	8 - 16
13	Desenvolvimento e teste de software/projeto	4 - 8	16 - 32
14	Orientado a gráficos	8 - 16	32 - 64
15	Aplicações críticas da Internet	8 - 16	32 - 64
16	Serviços de computação móvel	2 - 4	8 - 16

As categorias utilizadas neste trabalho baseiam-se em trabalhos anteriores [Gupta et al., 2019, Joe, 2022, KarlErickson, 2022, Rautiainen and Marschall, 2020, Singh et al., 2020]. Utiliza-se essa estratégia, principalmente, para novas aplicações, otimizando a alocação de MV mesmo sem dados históricos. Esse processo permite o uso eficiente

de recurso para melhorar o desempenho. Nos casos em que o usuário não especifica uma categoria para a aplicação, o sistema seleciona progressivamente começando com a máquina de menor capacidade de recursos de CPU e memória, aumentando gradualmente o poder computacional.

4.2.1 Monitoramento

O agente Mnt monitora o uso dos recursos de MVs, incluindo (CPU) e (Memória de Acesso Aleatório (RAM)). O módulo de monitoramento considera o uso de recursos, desempenho e custo de execução. Armazenam-se os resultados para criar uma base histórica de uso para atender ao raciocínio dos agentes.

O agente Mnt é do tipo reativo simples (vide Seção 2.3.1). Este registra os dados capturados em uma estrutura vetorial em tempo de execução e salva em um arquivo CSV ao término da execução da carga de trabalho. A Ferramenta para Gerar Estatísticas de Recursos de Sistema (DSTAT)[2] monitora e obtém as informações das MVs. Adota-se o DSTAT porque fornece as informações necessárias de forma organizada, além de ser personalizável para capturar as informações desejadas, permitindo exportar dados estatísticos de tempo de execução.

O comando para capturar os dados no DSTAT é *dstat -cmnd –noheaders –output stats.csv*. O parâmetro *cmnd* indica que a captura terá a CPU e a memória. Os dados são salvos em um arquivo CSV externo chamado stats.csv. Armazenam-se os dados de execução em estruturas de dados na memória das MVs para serem lidos e processados com agilidade durante a execução da aplicação.

4.2.2 Predição

Nesta seção, apresenta-se o modelo de predição de tempo de execução e uso de recursos implementado na arquitetura do MAS-Cloud+, utilizando a RLM.

Os modelos de predição podem utilizar diversas abordagens, como aprendizado de máquina Teylo et al. [2017], Zahoor et al. [2018] e estatística Harrell [2015], Kumar and Rao [2020]. Um método comumente utilizado em estatística é a regressão linear, que estima um valor com base em dados de entrada Mandal et al. [2020], Sri and Balaji [2019], Zharikov et al. [2020], Ziraba and David [2018].

O MAS-Cloud+ utiliza a RLM para estimativa de recursos por meio do perfil de aplicação, conforme incluído na taxonomia e levantamento de dados em Qu et al. [2018], ou de um esquema sensível à aplicação, como citado em esquemas de classificação de MV em Masdari et al. [2020].

[2]http://dag.wiee.rs/home-made/dstat/

O problema de predição pode ser modelado como um problema linear, em determinadas situações, em que se busca encontrar uma função linear que relaciona as características da MV (como VCPU e memória) com o tempo de execução e custo. Considerando uma lista de MVs com diferentes configurações de VCPU e memória, o objetivo é encontrar uma função linear que minimize a diferença entre os valores reais e os valores previstos pelo modelo.

Ressalta-se que, embora a abordagem linear seja comumente utilizada para modelar o problema de predição do tempo de execução, há situações em que a relação entre as características da MV e o tempo de execução pode ser não linear. Nesses casos, é necessário explorar outras técnicas, como usando aprendizado de máquina, a fim de capturar as complexidades do problema.

De acordo com Qu et al. [2018], no contexto de estimativa de recursos, desempenho de aplicação e carga de trabalho para nuvem computacional, o uso de métodos baseados em regressão linear mostra-se adequado com o perfilamento de aplicação (*application profiling*). À medida que mais dados se tornam disponíveis, os resultados previstos tornam-se mais precisos, com a vantagem de convergir, rapidamente, mesmo com um número menor de amostras de dados.

Regressão Linear Múltipla

No MAS-Cloud+ se redefinem as equações de tempo de execução e CPU, propostas em Ralha et al. [2019], atualizando Quantidade de Memória na MV (QM) e Quantidade de CPU na MV (QC), que representa, respectivamente, a quantidade de memória e de VCPU, com RLM dos dados capturados da execução da aplicação (vide RLM na Seção 2.2.1). Com RLM foi possível prever adequadamente o uso de recursos para escolher as instâncias de MV que melhor atendem aos requisitos da aplicação, conforme as necessidades do usuário. Nas equações 4.1, 4.2 e 4.3, podem-se ajustar os parâmetros com base em dados obtidos em execuções iniciais da aplicação, considerando tempo de execução, custo e uso de recursos. O tempo é representado por Tx, a CPU usada por Cx, e a memória por Mx, com os coeficientes do RLM (por exemplo, T1, T2, T3, C1, C2, C3, M1, M2 e M3). Representa-se o número de CPUs por QC e a quantidade de memória por QM.

$$Tempo = T1 + T2 * QM + T3 * QC \tag{4.1}$$

$$Usodecpu = C1 + C2 * QM + C3 * QC \tag{4.2}$$

$$UsodeMemória = M1 + M2 * QM + M3 * QC \tag{4.3}$$

Neste trabalho, utilizaram-se essas equações para todas as combinações de MVs dos provedores. Para exemplificar o uso do RLM para predição de tempo de execução em uma

instância AWS EC2, considera-se uma máquina *p3.2xlarge* com 8 VCPUs e 61 GiB de RAM. Para calcular o tempo de execução esperado para uma carga de trabalho, utiliza-se a Equação 4.1: $Time = T1+T2*61+T3*8$. Para calcular a porcentagem de uso da CPU, utiliza-se a Equação 4.2: $CPUUsage = C1+C2*61+C3*8$. Para calcular a porcentagem de uso de memória, a Equação 4.3: $MemoryUsage = M1 + M2 * 61 + M3 * 8$. Após realizar os cálculos de predição, o agente VMMgr utiliza distintos modelos de raciocínio para selecionar a melhor configuração de MV para executar a aplicação do usuário (Seção 4.3).

4.2.3 Provisionamento de Máquinas Virtuais

O agente VMMgr desempenha o papel de coordenador geral na arquitetura do MAS-Cloud+, facilitando a comunicação entre a interface do usuário e a camada de execução na nuvem. O VMMgr realiza predições e provisiona recursos nos provedores de nuvem, assegurando que uma MV seja instanciada para cada solicitação do usuário, considerando o tempo de execução, custo e desperdício de recursos.

O agente VMMgr realiza a instanciação das MVs com base em dados históricos armazenados e nos acordos de SLA. Os valores limite do SLA para garantir a QoS são baseados nas médias de tempo de execução, custo e uso de recursos das execuções iniciais da aplicação (Seção 5.1). O VMMgr garante a continuidade da execução da aplicação, identificando falhas nas MVs e reiniciando-as, visando minimizar a taxa de violações de SLA. A carga de trabalho é executada em MVs com taxa de violação de SLA inferior ao limite estabelecido pelo usuário. Utilizam-se as métricas adaptadas de Singh et al. [2017] para calcular a taxa de violação do SLA, conforme as Equações 4.4 a 4.7, onde txT é a taxa de violação do SLA de tempo de execução, T é o tempo estimado da carga de trabalho do usuário e T_{SLA} representa o limite superior de tempo de execução pré-definido pelo SLA. O $\bar{R}$ é a estimativa média do uso de recursos da carga de trabalho n, N é o total de cargas de trabalho, e R_n é a média do uso de recursos da carga de trabalho n. O txR é a taxa de violação do SLA de uso de recursos. Além disso, R_{SLA} representa o limite superior de utilização de recursos pré-definido pelo SLA. txC representa a taxa de violação do SLA. Adicionalmente, C é o custo estimado da carga de trabalho, em que C_{SLA} é a restrição de orçamento definida pelo usuário para o valor máximo a ser pago. O objetivo é minimizar a soma do custo de alocação, tempo de execução e utilização desperdiçada de recursos para cada carga de trabalho submetida pelo usuário no intervalo t.

$$txT = \frac{T}{T_{SLA}} \tag{4.4}$$

$$\bar{R} = \sum_{n=1}^{N} (\frac{R_n}{N}) \tag{4.5}$$

$$txR = \frac{\bar{R}}{R_{SLA}} \tag{4.6}$$

$$txC = \frac{C}{C_{SLA}} \tag{4.7}$$

Para ilustrar o uso da métrica, apresenta-se a Tabela 4.3, na qual a primeira coluna inclui os resultados da execução da carga de trabalho considerando o tempo de execução (T), custo (C) e uso de recursos (R). Conforme apresentado na segunda coluna da tabela, o usuário tem as restrições de que o tempo de execução não deve exceder 1800 segundos (T_{SLA}), o custo de execução não pode exceder 50 dólares (C_{SLA}) e o desperdício de recursos não pode ser superior a 50% (R_{SLA}). A terceira coluna apresenta a taxa de violação do SLA considerando o tempo de execução (txT), custo (txC) e uso de recursos (txR). Por exemplo, na primeira linha da tabela, há uma violação do SLA de tempo de execução ($txT = 1,11$). O agente VMMgr, usando os três modelos de raciocínio (ou seja, heurístico, otimizado, meta-heurístico), escolherá uma MV que evite violar qualquer uma dessas restrições.

Tabela 4.3: Ilustração de uso da taxa de violação do SLA.

T	**T_{SLA}**	**txT**
2000	1800	**1,11**
C	C_{SLA}	**txC**
35	50	**0,7**
R	R_{SLA}	**txR**
40	50	**0,8**

4.3 Modelos de Raciocínio

O VMMgr utiliza modelos distintos de raciocínio para o provisionamento de recursos na nuvem. Nesta seção são detalhados os modelos: heurístico (Seção 4.3.1), de otimização com programação linear (Seção 4.3.2) e meta-heurístico (Seção 4.3.3).

Em relação aos elementos de entrada para os modelos distintos de raciocínio de provisionamento são utilizadas os valores de SLA e as prioridades atribuídas em termos de tempo de execução, custo e minimização do desperdício de recurso, fornecidos pelo usuário, sendo que a soma das prioridades é sempre igual a um. Do provedor são obtidas

informações referentes às configurações das MVs disponíveis com a precificação. Também são utilizados pelo VMMgr os valores de predição para tempo de execução, custo e uso de recursos calculados pelo modelo de predição anterior ao provisionamento.

4.3.1 Heurístico

A tomada de decisão do agente VMMgr utiliza heurística baseado em regras proposicionais com uma BC e um motor de inferência. Visto que as regras impactam na decisão do agente, é possível analisar o comportamento da aplicação para as regras adequadas. Houve necessidade de um motor de inferência para integrar a plataforma JADE (Seção 2.3). Como em Ralha et al. [2019], utilizou-se o motor de inferência Drools Browne and Johnson [2009] para viabilizar a submissão de diferentes aplicações sem ajustar as regras de inferência.

Os Algoritmos 4.1 e 4.2, juntamente às Figuras 4.3 e 4.4, apresentam as regras de inferência de provisionamento para selecionar as melhores MVs utilizadas pelo agente VMMgr no serviço de provisionamento do MAS-Cloud+. A primeira regra cria a lista de MVs candidatas, e a segunda seleciona a MV adequada.

Utilizam-se os dados das instâncias de MVdos provedores pelo MAS-Cloud+ por meio da API de comunicação do provedor, conforme a Seção 4.2, Figura 4.2. Esses dados incluem nome da instância, núcleos de vCPU, memória (GiB) e preço (USD por hora). No Algoritmo 4.1, a primeira MV disponível é inserida na $candidate\_VMs\_List$ (Linha 7). Utilizam-se os dados da primeira MV para comparar a necessidade da aplicação calculada pelo método de predição RLM de tempo de execução, custo, uso de CPU e memória, calculados pelas Equações 4.1, 4.2 e 4.3, respectivamente (Linhas 10-13). Além disso, tempo de execução, custo, memória e uso de CPU não podem violar o limiar de SLA estabelecido como entrada pelo usuário (Linhas 14-17). O laço *while* (Linhas 9-18) compara todas as MVs disponíveis em $available\_VMs$, verificando as instâncias adequadas para incluir na $candidate\_VMs\_List$. Essa estratégia simples evita cenários impossíveis. Na prática é possível ajustar a implementação para otimizar o desempenho com muitas MVs usando uma abordagem paralela, mas preservando a lógica central do algoritmo.

Como descrito no Algoritmo 4.1, durante o processo de seleção das MVs, algumas condições são necessárias. O tempo de execução e o custo da MVdevem ser menores do que as MVs já listadas como candidatas. O uso médio de CPU e memória deve ser maior do que a média geral de todas as MVs disponibilizadas pelo provedor, calculado conforme os dados obtidos pelo RLM. O uso de recursos de CPU e memória deve ser menor do que aqueles estabelecidos pelo acordo de SLA. Essa estratégia ajuda a garantir que os candidatos a MVtenham o melhor tempo de execução, custo e menor desperdício.

A primeira regra na Figura 4.3 tem como entrada o limiar de SLA e os dados obtidos com a RLM apresentados nas Equações 4.1, 4.2 e 4.3, que calculam o tempo médio de execução, uso de CPU e memória. Essa primeira regra combina os dados históricos dos usuários com as configurações das MVs fornecidas pelo provedor de nuvem.

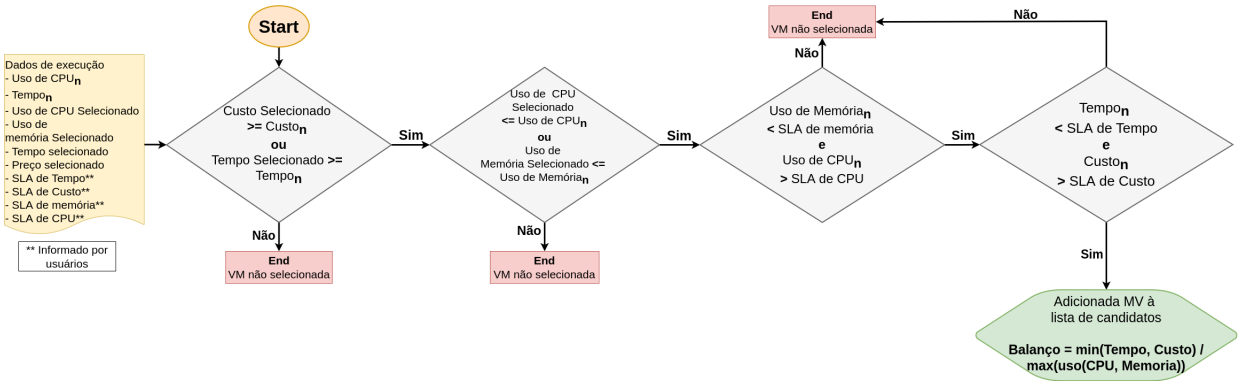

Figura 4.3: A primeira regra de inferência de provisionamento usada para criar a lista de MVs candidatas, conforme o Algoritmo 4.1.

Algoritmo 4.1: Regra de inferência para criar a lista de VMs candidatas.

```
1  Entrada: available_VMs
2  Saída: candidate_VMs_List
3
4  função Regra de inferência 1 para provisionamento retorna
5  candidate_VMs_List
6
7  Adicionar available_VMs.FirstVM ao candidate_VMs_List
8
9  enquanto available_VMs.next
10   se (Tempo Selecionado >= Tempo_n OU
11       Custo Selecionado >= Custo_n)
12     se (Uso de CPU Selecionado <= UsodeCPU_n OU
13         Uso de Memória Selecionado <= UsodeMemória_n)
14       se (Tempo_n <= Tempo SLA E
15           Custo_n <= Custo SLA E
16           UsodeMemória_n <= Memória SLA E
17           UsodeCPU_n <= PU SLA)
18         Adicionar available_VMs_n ao candidate_VMs_List
19 retorna candidate_VMs_List
```

A *candidate_VMs_List* é a entrada para o Algoritmo 4.2 (Linha 1), recebida do Algoritmo 4.1, atribuindo a primeira MV ao *VM_Selected* (Linha 6). O laço enquanto

(Linhas 9-18) compara todas as MVs na $candidate\_VMs\_List$ para escolher a melhor instância, balanceando tempo de execução, custo, uso de CPU, uso de memória e uso de recursos, calculado pela equação $\frac{\min(\text{Tempo, Custo})}{\max(\text{uso(CPU, Memória)})}$. O resultado desse algoritmo é a MV selecionada para executar a aplicação.

Algoritmo 4.2: Regra de inferência para selecionar a melhor VM.

```
1  Entrada: candidate_VMs_List
2  Saída: VM_Selected
3
4  função Regra de inferência 2 para provisionamento retorna VM_Selected
5
6  VM_Selected = candidate_VMs_List[0]
7
8  enquanto candidate_VMs_List.next
9    se (Balanceamento_n >= Melhor Balanceamento)
10     se (Tempo_n <= Tempo Selecionado)
11       se (Custo_n <= Custo Selecionado)
12         VM_Selected = candidate_VMs_List_n
13 retorna VM_Selected
```

A segunda regra na Figura 4.4 tem como entrada a lista de MVs candidatas. Essa regra verifica quais têm melhor balanceamento, o melhor desempenho e o melhor custo, retornando uma única MV em que será executado.

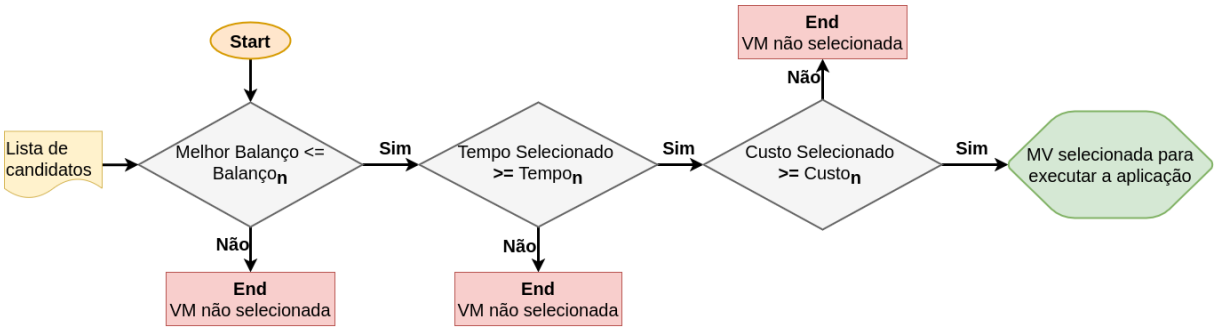

Figura 4.4: Inferência para a realização da segunda etapa do processo de provisionamento de MV pelo VMMgr, conforme o Algoritmo 4.2.

4.3.2 Otimização

A otimização combinatória é um dos modelos de raciocínio da arquitetura MAS-Cloud+ incorporada ao agente VMMgr (Seção 2.4.3). O problema de determinar um mapeamento X_{nm} entre a carga de trabalho n e uma configuração de MV.m, representado como

um programa linear inteiro, é formulado como nas Equações 4.8 a 4.13. Nas equações, considera-se que há m MVs na nuvem e a submissão de n cargas de trabalho. Na validação experimental, o usuário submete uma carga de trabalho por vez ($n = 1$).

$$minimize(\sum_{n}^{N}(X_{nm}A_{nm})) \tag{4.8}$$

$$A_{nm} = (D_{nm}\alpha_d) + (C_{nm}\alpha_c) + (T_{nm}\alpha_t) \tag{4.9}$$

As Equações 4.8 e 4.9 estão sujeitas às restrições:

$$\sum_{m=1}^{M} X_{nm} = 1, \forall n \in 1...N \tag{4.10}$$

$$\sum_{n=1}^{N}\sum_{m=1}^{M} T_{nm} * X_{nm} * R_{nu} \leq T_u, \forall u \in 1...U \tag{4.11}$$

$$\sum_{n=1}^{N}\sum_{m=1}^{M} D_{nm} * X_{nm} * R_{nu} \leq D_u, \forall u \in 1...U \tag{4.12}$$

$$\sum_{n=1}^{N}\sum_{m=1}^{M} C_{nm} * X_{nm} * R_{nu} \leq C_u, \forall u \in 1...U \tag{4.13}$$

Na Tabela 4.4 apresenta-se a notação completa do modelo de otimização, em que X_{nm} é a variável de decisão binária é A_{nm}, sujeito às restrições apresentadas pelas Equações 4.10, 4.11, 4.12 e 4.13. A Equação 4.9 define A_{nm} como uma combinação linear de D_{nm}, C_{nm} e T_{nm}, com coeficientes α_d, α_c e α_t, respectivamente. A_{nm} é a soma das variáveis de tempo de execução, custo e desperdício de recursos, da carga de trabalho n na MV.m. D_{nm} é o uso estimado de recursos da carga de trabalho n. D_u representa o limite superior de utilização de recursos pré-definido pelo SLA. C_{nm} é o custo estimado da carga de trabalho n. C_u representa o limite superior definido pelo usuário. T_{nm} é o tempo estimado de execução da carga de trabalho n pelo usuário. T_u representa o limite superior de tempo de execução pré-definido pelo SLA. $R_{nu} = NxU$. R_{nu} é uma matriz de associação entre tarefas e seus usuários. Se a tarefa n pertence ao usuário u, é igual a 1 e 0 caso contrário. Na Equação 4.9, α_d, α_c e α_t são parâmetros de interesse e prioridade fornecidos pelo usuário.

Com o modelo de otimização, busca-se minimizar a soma do custo de alocação, tempo de execução e uso de recursos desperdiçados para cada carga de trabalho submetida pelo usuário por intervalo de tempo t, sujeito a quatro restrições (Equações 4.10, 4.11, 4.12 e 4.13). A restrição 4.10 afirma que cada solicitação do usuário deve alocar uma única MV.

Tabela 4.4: Notação do modelo de otimização.

Notação	Descrição
n	Carga de trabalho enviada pelo usuário
m	máquina virtual m
u	usuário
N	Quantidade de carga de trabalho
M	Número total de MVs disponíveis
U	Total de usuários
X_{nm}	Variáveis binárias de decisão
D_{nm}	uso de recursos na máquina virtual m para a carga de trabalho do usuário n
C_{nm}	Custo estimado da máquina virtual m de carga de trabalho n
T_{nm}	Tempo estimado de carga de trabalho n da máquina virtual m
A_{nm}	$D_{nm} + C_{nm} + T_{nm} = A_{nm}$
R_{un}	Matriz de associação entre tarefas e seus usuários
T_u, D_u e C_u	Variáveis de tempo de execução, uso de recursos e custo acordados com SLA
α_d, α_c e α_t	Pesos indicados pelo usuário para cada critério de minimização

As restrições 4.11, 4.12 e 4.13 definem limites, em que T_u é o tempo, D_u é a porcentagem de recursos desperdiçados e C_u é o custo. O usuário fornece esses valores considerando o SLA.

4.3.3 Meta-heurístico

Um dos modelos de raciocínio da arquitetura MAS-Cloud+ incorporado ao agente VMMgr é o meta-heurístico, que utiliza GRASP para provisionamento de recursos de MVs. A implementação do GRASP possui duas etapas (Algoritmo 2.1): a construção (Algoritmo 2.2) e a busca local (Algoritmo 2.3).

Apresenta-se a função objetivo que retorna a qualidade da solução na Equação 4.14. A função de busca minimiza a soma de tempo de execução, custo e recursos desperdiçados para cada carga de trabalho submetida pelo usuário. Na Tabela 4.5, apresenta-se a notação GRASP, em que as variáveis Q_x representam os valores de SLA para tempo de execução, custo e recursos desperdiçados. T_x, C_x e D_x são os valores estimados de tempo de execução, custo e desperdício para uma MV.x. Finalmente, b e B são os valores ótimos encontrados pela busca local e o melhor valor encontrado pelo GRASP, respectivamente. α_x representa os pesos especificados pelo usuário para cada critério de minimização.

$$F(x) = \alpha_t * (max[0, Q_t - T_x]) + \alpha_c * (max[0, Q_c - C_x]) + \alpha_d * (max[0, Q_d - D_x]) \quad (4.14)$$

Tabela 4.5: Notação das variáveis do GRASP.

Notação	Descrição
Qt	QoS de tempo de execução que deverá ser garantido pelo SLA
Qc	QoS de custo que deverá ser garantido pelo SLA
Qd	QoS de desperdício de recursos que deverá ser garantido pelo SLA
Tx	Tempo de execução estimado para MV x
Cx	Custo estimado para MV x
Dx	Desperdício de recursos estimado para MV x
b	Melhores valores na atual busca local
B	Melhor valor para a função objetivo
α_x	Pesos especificados pelo usuário para cada critério de minimização

Para os experimentos realizados neste trabalho, as Equações 4.9 a 4.14 consideraram 0,33 para α_x, uma vez que apenas os valores de SLA serão considerados.

4.4 Escolha Automática dos Modelos de Raciocínio

A seleção automática dos modelos de raciocínio na arquitetura MAS-Cloud+ desempenha um papel fundamental para o provisionamento de recursos em ambientes de nuvem. Esse processo envolve a análise e a comparação dos modelos de provisionamento, considerando e selecionando entre abordagens heurística, meta-heurística e otimizada. Essa abordagem automatizada permite a identificação dos modelos mais adequados para cada cenário, garantindo a eficiência no gerenciamento dos recursos na nuvem.

4.4.1 Modelo de Provisionamento

A seleção adequada de modelos de provisionamento de MV é fundamental para garantir a eficiência e a qualidade do desempenho em ambientes de computação em nuvem. Utilizando um modelo automático, é possível escolher entre abordagens de provisionamento como heurísticas, meta-heurísticas e otimização.

As heurísticas oferecem soluções rápidas, mas podem não ser ótimas para cenários complexos. As meta-heurísticas exploram um espaço de busca mais amplo, buscando soluções de melhor qualidade, porém com maior tempo computacional. Os métodos de otimização formulam o problema matematicamente em busca de soluções ótimas, mas

podem exigir mais recursos computacionais. A escolha do modelo apropriado depende da complexidade e do tamanho do conjunto de dados. É importante considerar o tempo de execução, o custo e a eficiência na seleção das configurações de MV.

Na Figura 4.5, ilustram-se as variações de tempo na seleção das configurações de MV para cada modelo. O modelo heurístico apresenta um tempo constante. O modelo meta-heurístico indica o crescimento linear e o otimizado um crescimento exponencial. Essas informações auxiliam a escolha adequada do modelo, considerando a complexidade do conjunto de dados.

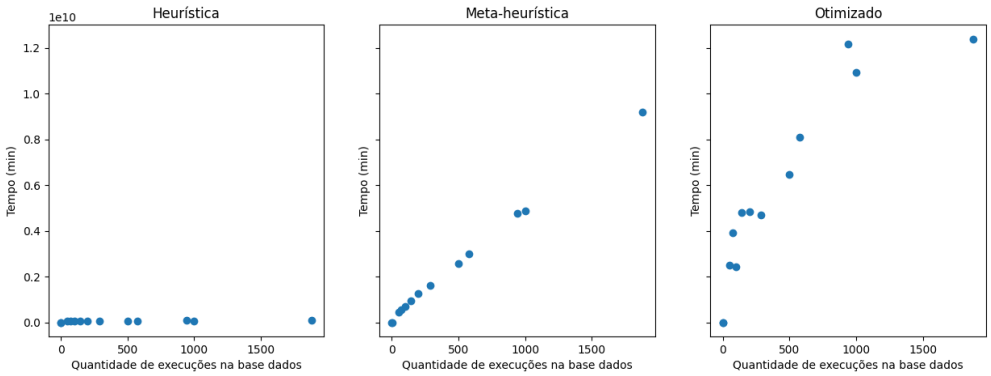

Figura 4.5: Tempo de escolha de cada modelo de raciocínio conforme a quantidade histórico de registros.

O algoritmo de seleção (Algoritmo 1) descreve um método baseado no Custo Benefício (CB) da execução. Inicialmente, para cada um dos modelos de raciocínio disponível (Linha 1), calcula-se a média do tempo de execução e do custo de execução (Linha 2), além do tempo de escolha e o CB (Linha 3). Após estas avaliações, os modelos são ordenados com base no tempo total (soma do tempo de execução com o tempo de escolha), no custo e no custo-benefício (Linha 5). O próximo passo envolve a seleção do melhor modelo, considerando esses três critérios (Linhas 6-13). Este processo de avaliação e seleção garante uma escolha balanceada no tempo de execução e custo do modelo, apropriado para o provisionamento.

4.5 Aspectos Tecnológicos

A implementação da arquitetura MAS-Cloud+ empregou diversas tecnologias avançadas, cada uma desempenhando um papel específico. Na sequência será apresentado cada uma.

As principais linguagens de programação utilizadas foram *Java* e *Python*. Utilizou-se *Java* para a implementação da arquitetura MAS-Cloud+ com o *middleware* JADE.

Algoritmo 1 Cálculo do método com melhor custo benefício.

Require: Base de dados

1: **para** cada modelo [heuristico, otimizado, metaheuristico] **faça**
2: Calcula média de tempo_execução e custo de execução
3: Calcula tempo_escolha e custo_beneficio
4: **fim para**
5: **ordenar** modelos por tempo_total(tempo_execução + tempo_escolha), custo e custo_beneficio
6: **selecionar** melhor modelo com base em tempo_total, custo e custo_beneficio
7: **se** O melhor método é o heurístico **então**
8: executa o metodo_heuristico
9: **senão se** O melhor método é o MetaHeurístico **então**
10: executa o metodo_metaheuristico
11: **senão se** O melhor método é o otimizado **então**
12: executa o metodo_otimizado
13: **fim se**

Conforme apresentado na Seção 2.3.3, o JADE facilita a implementação de SMA, segundo as especificações FIPA para agentes de software interoperáveis. Assim sendo, esta foi a linguagem usada para a implementação e o gerenciamento dos agentes do sistema [Bellifemine et al., 2007].

Utilizou-se a linguagem Python para *scripts* do método de otimização e predição, incluindo manipulação de dados e interação com as bibliotecas do *OR-Tools* e de regressão linear. A biblioteca de código aberto *OR-Tools*, foi usada para modelar o problema de otimização combinatória [Kruk, 2018].

Foi usado uma *API REST* para a implementação do raciocínio heurístico e metaheurístico — que exige bom desempenho do sistema.

Para monitorar as MVs durante a execução das aplicações na nuvem e coletar dados sobre o uso de recursos foi usado a ferramenta DSTAT[3], que permite gerar estatísticas de recursos do sistema Malik et al. [2017].

Para o Gerenciamento de Regras de Negócios - *Business Rules Management System* (BRMS) foi usado o *Drools* implementado em Java. O mecanismo de gerenciamento de regras de inferência fornece a tomada de decisão heurística para o agente VMMgr [Browne and Johnson, 2009].

A ferramenta *Vagrant*[4] foi utilizada para a construção e gestão de ambientes de MV. Empregou-se esta para a criação da API de comunicação do *core* do MAS-Cloud+ com os provedores de nuvem Hashimoto [2013].

[3]https://linux.die.net/man/1/dstat
[4]https://www.vagrantup.com

Algumas bibliotecas de regressão linear em Python foram utilizadas, incluindo Scikit-learn[5], Statsmodels[6], e Numpy[7], para construir e treinar modelos de regressão linear.

Disponibilizam-se a implementação do *core* do MAS-Cloud+ e a API nos seguintes repositórios de código aberto:
`https://gitlab.com/InfoKnow/CloudFogComputing/aldomendes-mas-cloudplus` e
`https://github.com/aldo2102/mas-cloud-plus-API`.

[5]https://scikit-learn.org
[6]https://www.statsmodels.org
[7]https://numpy.org

Capítulo 5

Análise Experimental

Neste capítulo é apresentada a resposta à seguinte questão de pesquisa:

- QP3 – Quais são as métricas adequadas para avaliar a eficiência da proposta arquitetural de gerenciamento de recursos na nuvem, independente de aplicação e para múltiplos provedores, incluindo os modelos de predição e provisionamento de MVs?

O conteúdo deste capítulo inclui detalhes sobre os experimentos realizados para validar a arquitetura MAS-Cloud+. Na Seção 5.1 são fornecidas as especificações dos experimentos (*setup* experimental), incluindo o processo de execução (Seção 5.1.1), a parametrização (Seção 5.1.2) e a caracterização da carga de trabalho (Seção 5.1.3). Descrevem-se dois experimentos em domínios distintos, sendo um na área de Bioinformática, com alinhamento de sequências de DNA (Seção 5.2) e um estudo comparativo com solução do estado da arte em *Big Data* Islam et al. [2022], usando o *Apache Spark Benchmark* com três aplicações, *WordCount* (intensivo em CPU), *Sort* (intensivo em memória) e *PageRank* (intensivo em rede) (Seção 5.3). As discussões de resultados experimentais são apresentadas na Seção 5.4.

Considerando à QP3, para a predição foram utilizadas as métricas MSE, R^2 e MAE quantificando erros e diferenças entre valores previstos e observados. Para o provisionamento, a eficiência foi avaliada pela taxa de violação de SLA, considerando tempo de execução, custo e uso de recursos, além do tempo para escolha da MV e a relação custo-benefício para cada modelo de raciocínio do MAS-Cloud+, conforme apresentado nas Seções 5.2.5 e 5.3. Ressalta-se que tais avaliações são realizadas independentemente de aplicação com possibilidade de uso para novos módulos de servições de provisionamento e predição, a serem implementados no futuro, assegurando um modelo de avaliação uniforme.

5.1 *Setup* Experimental

Nesta tese, considera-se que a metodologia adotada seja exploratória, experimental, com enfoque de análise quantitativo e qualitativo, no tema de gerenciamento de recursos em nuvem computacional. Portanto, coletaram-se e analisaram-se dados sobre o uso de recursos, alocados para cada aplicação submetida à execução pelo usuário, para análise quantitativa do tempo de execução, custo e desperdício de recursos. Além disso, foi analisada nos experimentos a taxa de violação de SLA.

Adicionalmente, é importante ressaltar que, nos experimentos realizados, quando se refere ao tempo de execução, é o tempo que levou entre a submissão e a conclusão da aplicação. O tempo de escolha, por sua vez, é o tempo que os modelos de raciocínio levaram para escolher a MV mais adequada para a execução da aplicação. O tempo total é a soma do tempo de execução e o tempo de escolha, representando assim o tempo total necessário para completar o ciclo de execução da aplicação na infraestrutura de nuvem.

Desta forma, consideram-se os seguintes aspectos para os experimentos:

- objetivo – analisar os módulos arquiteturais de monitoramento, predição e provisionamento do MAS-Cloud+, com três modelos de raciocínio distintos para a tomada de decisão dos agentes, usando aplicações em diferentes domínios, Bioinformática e *Big Data*. Também objetiva-se avaliar o gerenciamento de MVs em múltiplos provedores de nuvem (AWS e *Google Cloud*).

- cenário real de nuvem com múltiplos provedores – AWS EC2 com diferentes configurações de MV, possibilitando realizar e comparar experimentos com distintas capacidades computacionais e custos, permitindo realizar e comparar experimentos em ambiente heterogêneo, baseando-se na variação de VCPU, memória (GiB) e preço (USD/hora) das instâncias da AWS na aplicação de Bioinformática, conforme apresentado na Tabela 5.1. No estudo comparativo, a Tabela 5.2 apresenta instâncias AWS EC2 e do *Google Cloud* na Tabela 5.3, escolhidas como base comparativa para as instâncias de MV utilizadas no trabalho de Islam et al. [2022].

- aplicações utilizadas – para avaliar os modelos de raciocínio distintos (heurístico, otimizado e meta-heurístico) do MAS-Cloud+, considerando gerenciamento de recursos na nuvem, foram utilizados duas aplicações em domínios distintos. A primeira na Bioinformática, em que a seleção de instâncias concentram-se na execução do alinhamento de sequências de DNA, usando o *MASA-OpenMP*, com código disponível no GitHub Sandes et al. [2016].[1] Foram realizadas 30 execuções para cada modelo de raciocínio, considerando o tempo de execução e o custo médio. Para o

[1]https://github.com/edanssandes/MASA-OpenMP

estudo comparativo, utilizou-se o experimento de escalonamento baseado em SLA de trabalhos do *Spark* na nuvem híbrida apresentado por Islam et al. [2022]. Para comparar a implementação do MAS-Cloud+ com o trabalho de Islam et al. [2022], realizaram-se 100 execuções de cada aplicação (*WordCount*, *Sort* e *PageRank*) para os três modelos de raciocínio, computando a média do tempo de execução e o custo.

- estratégia de validação – analisam-se dados médios ao final de cada execução, considerando percentual de recursos, como CPU e memória, tempo de execução e custo para cada MV. O agente VMMgr utiliza o modelo de RLM com esses dados, para validar os resultados de predição e provisionamento, tanto na aplicação de Bioinformática, como no estudo comparativo.
- métrica de avaliação de resultados – para medir a eficiência no provisionamento, utilizou-se a taxa de violação de SLA, conforme apresentado por Singh et al. [2020], considerando tempo de execução, custo e uso de recursos, além do tempo para escolha da MV. Para avaliar o método de predição (RLM), foram utilizados as métricas MSE, R^2 e MAE. Para o estudo comparativo, utilizou-se o custo total, custo acumulado, percentual de cumprimento do prazo entre os algoritmos de escalonamento (quanto maior, melhor) e a duração média do trabalho entre os algoritmos de escalonamento para diferentes tipos de trabalhos (quanto menor, melhor).

Tabela 5.1: Instâncias da AWS EC2 utilizadas na aplicação de Bioinformática.

Instância	**vCPU**	**Memória (GiB)**	**Preço (USD/hora)**
t2.micro	1	1	$0.0116
m4.large	2	8	$0.1000
r4.xlarge	4	30.5	$0.2660
p3.2xlarge	8	61	$3.0600
m5.4xlarge	16	64	$0.7680
h1.8xlarge	32	128	$2.0960
d2.8xlarge	36	244	$5.5200
m4.10xlarge	40	160	$2.0000
c6in.12xlarge	48	96	$3.5760
p2.16xlarge	64	732	$14.400
c5.18xlarge	72	144	$4.9920
c5a.24xlarge	96	192	$4.6080

Obs.: apesar de algumas instâncias possuírem aceleradores de GPU, utilizou-se somente a CPU.

5.1.1 Processo de Execução

Os experimentos incluíram avaliação dos três modelos de raciocínio dos agentes, permitindo analisar os resultados de cada módulo e serviço oferecido na arquitetura MAS-

Cloud+. Na Figura 5.1 é apresentado o diagrama de fluxo de processo de execução da solução, com as atividades executadas.

Tabela 5.2: Instâncias da AWS EC2 usadas no estudo comparativo com Islam et al. [2022].

Instância	**vCPU**	**Memória (GiB)**	**Preço (USD/hora)**
t2.xlarge	4	16	$0.24
t2.2xlarge	8	32	$0.48
m5zn.3xlarge	12	48	$0.72

Tabela 5.3: Instâncias do *Google Cloud* usadas no estudo comparativo com Islam et al. [2022].

Instância	**vCPU**	**Memória (GiB)**	**Preço (USD/hora)**
e2-standard-4	4	16	$0.13
e2-standard-8	8	32	$0.26
custom-12-49152	12	48	$0.40

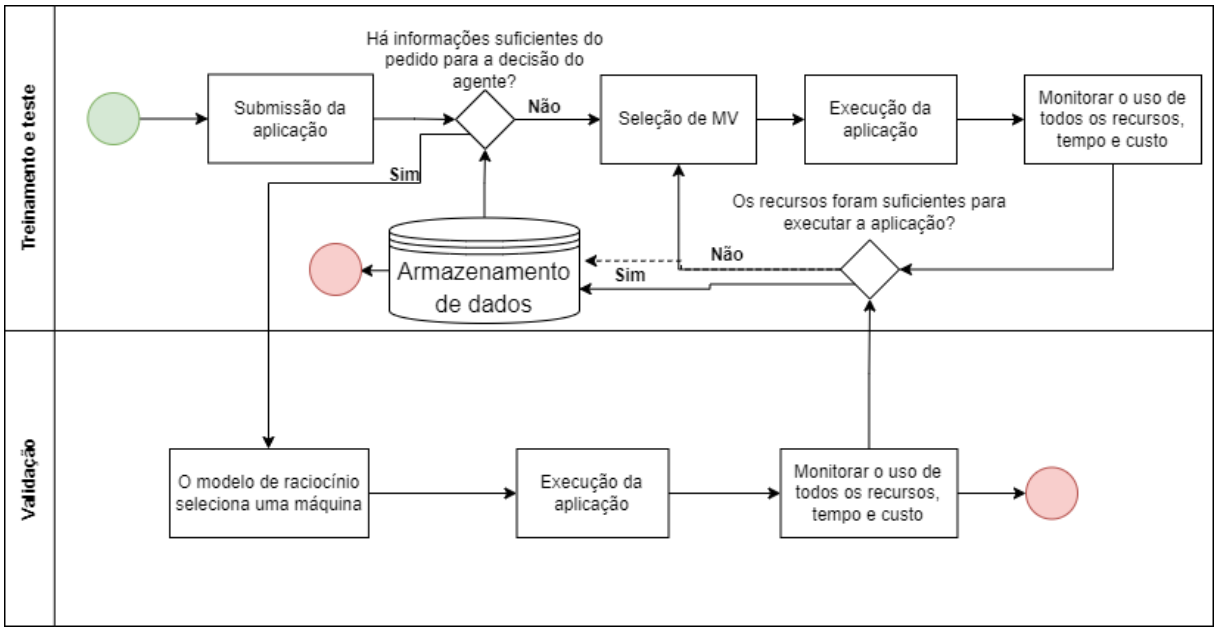

Figura 5.1: Diagrama de processos da fase experimental da execução das aplicações.

O passo a passo da execução dos experimentos inclui:

1. Uma MV é solicitada para executar a aplicação;
2. Se o armazenamento de dados não tiver informações suficientes para o agente VMMgr executar a predição e o provisionamento:

(a) a MV é selecionada progressivamente, começando com a menor disponível e aumentando gradualmente o poder computacional entre as disponíveis pelo provedor;

(b) a aplicação é executada na máquina instanciada;

(c) os dados de execução são monitorados pelo agente Mnt com frequência de um segundo;

(d) ao final da execução, atualizar o arquivo CSV no armazenamento de dados.

3. Se o armazenamento de dados tiver informações suficientes para o agente VMMgr:

(a) as configurações da MV são escolhidas pelo modelo de raciocínio ativo no agente indicado pelo usuário, usando o método de Gerenciamento da API MV com a entrada *DecisionModel* ou escolhido de maneira automática, que pode ser heurístico, de otimização e meta-heurístico;

(b) a aplicação é executada na máquina instanciada;

(c) os dados de execução são monitorados pelo agente Mnt;

(d) ao final da execução, os dados são atualizados no armazenamento de dados.

5.1.2 Parametrização

Parametrizaram-se os experimentos realizados nesta tese, considerando a requisição de instanciação de MVs em múltiplos provedores de nuvem. Como possíveis cenários, o usuário: (i) faz uma requisição, a MV é instanciada no provedor e a carga de trabalho é executada; (ii) faz uma requisição, a MV não é instanciada no provedor; (iii) faz uma requisição, a MV é instanciada e a carga de trabalho não é executada; (iv) não consegue fazer uma requisição.

Definiram-se os requisitos desejados de desempenho: (i) a MV deve ser instanciada com a quantidade de recursos estipulados pelo VMMgr; (ii) o usuário deve ser capaz de executar a carga de trabalho na MV instanciada; (iii) os recursos alocados precisam estar disponíveis no provedor. Realizou-se a avaliação dos requisitos por meio de modelagem analítica e experimentação do protótipo implementado.

5.1.3 Caracterização da Carga de Trabalho

Para caracterizar a carga de trabalho a ser utilizada pelo provisionamento de recursos, utiliza-se o monitoramento das MVs com a ferramenta *Dstat*. O *Dstat* é capaz de monitorar diversos recursos de uma MV, detalhados na Tabela 5.4. Ao final de cada execução, calcula-se a média de uso de cada recurso listado e o desempenho da aplicação.

Tabela 5.4: Dados monitorados das pelo MVs Dstat.

CPU				Memória				Rede		HD	
usr	**sys**	**idl**	**wait**	**used**	**free**	**buff**	**cach**	**recv**	**send**	**read**	**writ**

No contexto de monitoramento das MVs, utilizam-se as seguintes métricas:

- uso de CPU por processos de usuário (*usr*), sistema (*sys*), bem como o número de processos ociosos (*idl*) e aguardando (*wait*);
- memória: quantidade de memória usada (*used*), memória livre (*free*), armazenada em buffer (*buff*), armazenada em cache (*cach*);
- rede (*network*): número total de bytes recebidos (*recv*) e enviados (*send*) pelas interfaces de rede;
- disco rígido (HD): número total de operações de leitura (*read*) e gravação (*writ*).

Os dados do HD e da rede são coletados por serem *default* do Dstat, podendo ser utilizados em uma implementação futura de serviços no MAS-Cloud+.

5.2 Aplicação de Bioinformática

A aplicação de Bioinformática alinha sequências de DNA utilizando a ferramenta *MASA-OpenMP*, a qual faz uso dos algoritmos *Smith-Waterman/Needleman-Wunsch* e *Myers-Miller* Sandes et al. [2016]. *MASA-OpenMP* é uma extensão da arquitetura *Multi-Platform Architecture for Sequence Aligner* (MASA), que utiliza o *framework OpenMP* para reduzir o tempo de execução na CPU. A característica de desempenho dessa aplicação é intensiva em computação e memória, mas, como o acesso à memória é mais lento do que a computação, a característica de intensidade de memória é dominante. A entrada para a aplicação *MASA-OpenMP* refere-se a dois arquivos de sequência de DNA a serem alinhados. Para exemplificar o tipo de computação, consideram-se as sequências do *SARS-CoV-2*, em que serão realizados 30.000 x 30.000 x 3 cálculos de posição na matriz. A complexidade é quadrática, e calculam-se três matrizes, onde a cada cálculo é atualizado uma posição na matriz.

5.2.1 Descrição

Para os experimentos realizados nesta pesquisa, utilizou-se a ferramenta *MASA-OpenMP* com duas sequências de DNA (Sequências de DNA 1 e 2) de diferentes tamanhos (10K, 30K, 50K, 150K, 500K e 1M), contemplando seis alinhamentos distintos, conforme apresentado na Tabela 5.5 .

Tabela 5.5: Sequencias de DNA submetidas ao *MASA-OpenMP*.

Tamanho	**Sequência de DNA 1**	**Sequência de DNA 2**
10k	*HIV-1 isolate MB2059 from Kenya*	*HIV-1 isolate SF33 from USA*
30k	*SARS-CoV-2 UK*	*Syndrome coronavirus 2 isolate SARS-CoV-2/human /USA/AL-UAB-GX413/2021*
50k	*Allomyces macrogynus mitochondrion*	*Chaetosphaeridium globosum mitochondrial DNA*
150k	*Human herpesvirus 6B*	*Human herpesvirus 4 complete wild*
500k	*Rhizobium sp. NGR234 plasmid pNGR234a*	*Agrobacterium fabrum str. C58 plasmid At*
1M	Chlamydia muridarum Nigg	*Chlamydia trachomatis A/HAR-13*

Ao utilizar a nuvem da AWS/EC2, há disponibilidade de diversas configurações de MVs, permitindo a realização e comparação de experimentos em diferentes cenários. Para a seleção das MVs, elencaram-se configurações específicas que apresentam uma maior variabilidade de recursos, as quais estavam disponíveis no Projeto *BioCloudConselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)/AWS*, conforme apresentado na Tabela 5.1.

5.2.2 Resultados da Escolha Aleatória

Considera-se que escolhas inadequadas de MVs na computação em nuvem podem levar a diferentes cenários: (a) Cenário 1 – uma MV com maior capacidade de processamento e custo do que o necessário; (b) Cenário 2 – uma MV de baixo custo que pode apresentar desempenho insatisfatório; (c) Cenário 3 – uma MV que possui mais recursos do que o necessário (por exemplo, núcleos, memória), resultando em desperdício.

Considerando esses três cenários, as instâncias AWS da Tabela 5.1 e as cargas de trabalho apresentadas na Tabela 5.5, realizaram-se experimentos para comparar as escolhas aleatórias dos usuários com 1, 8 e 96 VCPUs e os modelos de provisionamento do agente VMMgr, conforme instâncias apresentadas na Tabela 5.7, resultando nas Figuras 5.2 e 5.3. Como os usuários de nuvem podem ter pouca experiência na escolha de MVs, as escolhas de instâncias podem ser realmente próxima do aleatório.

As escolhas de MVs feitas aleatoriamente pelo usuário apresentam problemas ao considerar algumas cargas de trabalho, uma vez que essas escolhas podem não observar as características da aplicação. Utilizaram-se diferentes cargas de trabalho para comparar o desempenho e os custos dos modelos de provisionamento de escolhas aleatórias e do MAS-

Cloud+ (modelos heurístico, otimização e meta-heurístico). Na Figura 5.2 apresentam-se as menores cargas de trabalho (10k, 30k, 50k), enquanto na Figura 5.3 as maiores (150k, 500k, 1M). Considerando MVs com 1 e 8 vCPUs, o desempenho do MAS-Cloud+ foi superior em 44% na execução de cargas totais, sendo inferior, apenas, com 96 vCPUs. No entanto, a arquitetura MAS-Cloud+ não considera, apenas, o tempo de execução, mas também o custo. Considerando as mesmas cargas de trabalho, o MAS-Cloud+ é 74% mais eficiente em termos de custo em execuções totais do que máquinas com 8 e 96 vCPUs, com um custo mais alto com apenas uma vCPU. Os resultados indicam que a arquitetura visa encontrar o melhor balanceamento entre tempo de execução e custo.

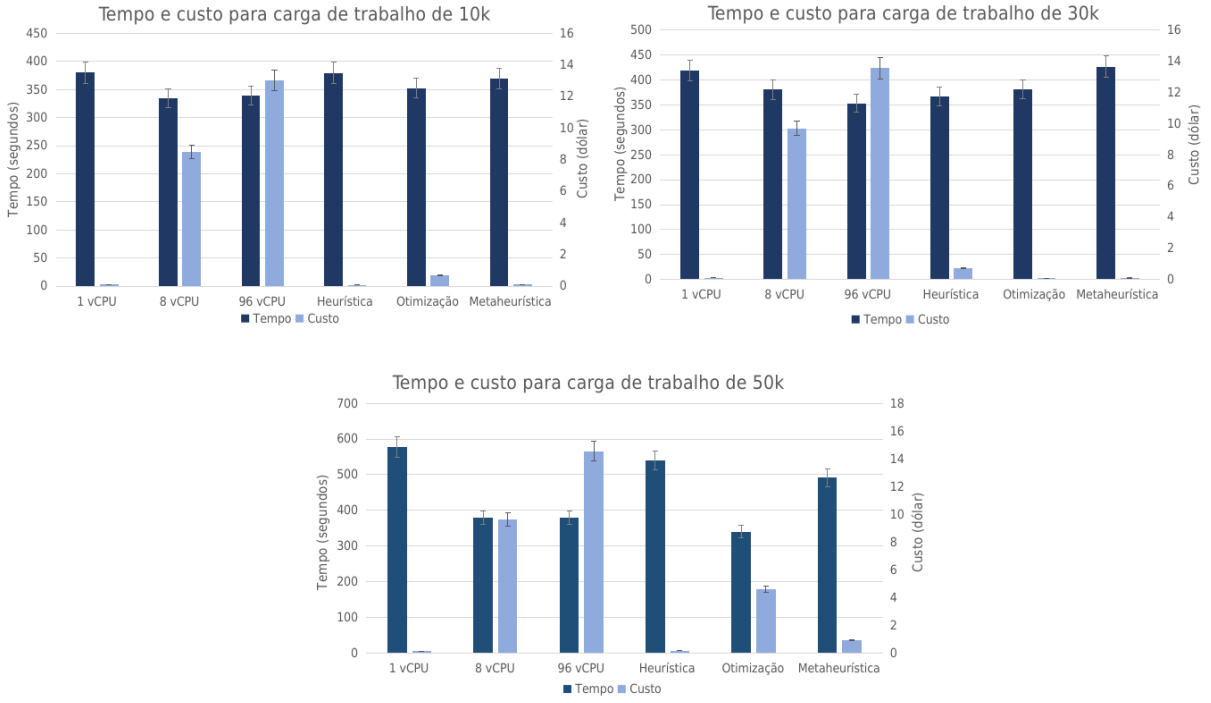

Figura 5.2: Comparação de tempo de execução e custo entre escolha aleatória e MAS-Cloud+, com cargas de tamanho 10k, 30k, 50k. As três barras à esquerda apresentam a escolha aleatória em diferentes instâncias (1, 8 e 96 VCPUs), enquanto as demais as escolhas do MAS-Cloud+, com os três modelos de raciocínio (heurístico, otimização, meta-heurístico).

Na Figura 5.3, em relação as cargas de trabalho de 500k e 1M com 1 VCPU, o tempo de execução apresentado é igual a 9000 segundos, tendo em vista que essas cargas não obtiveram conclusão durante a execução apresentando falhas por falta de recurso de memória.

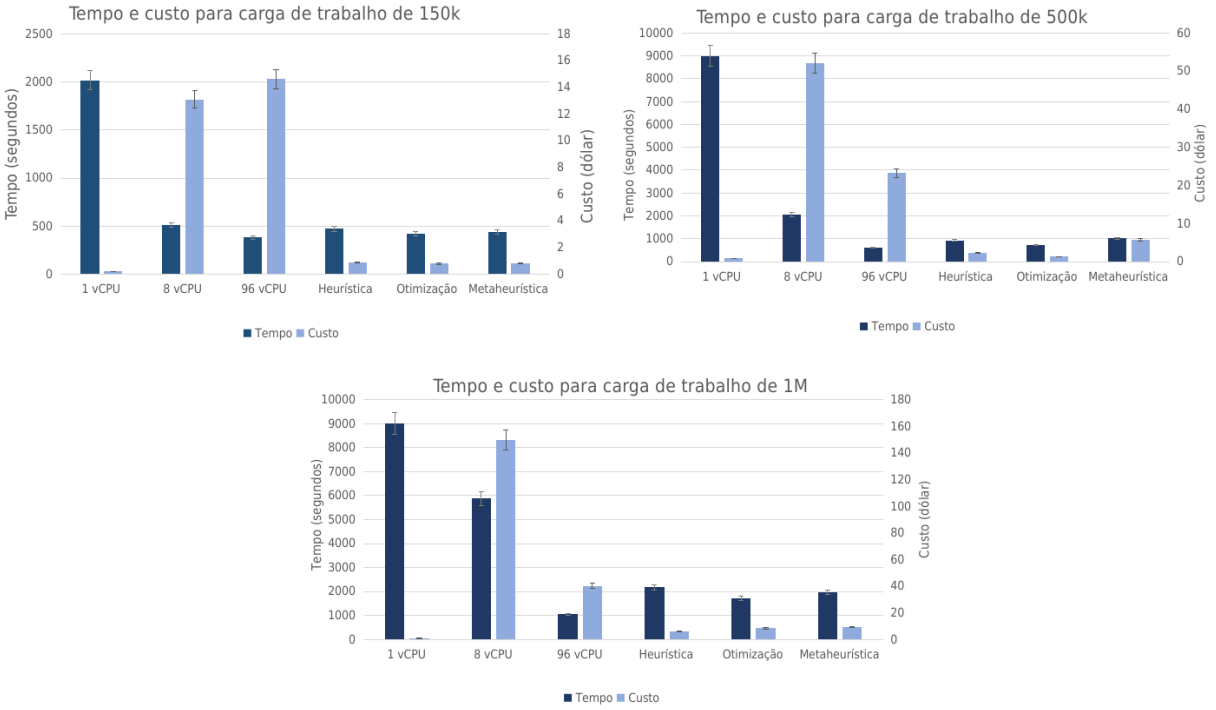

Figura 5.3: Comparação de tempo de execução e custo entre escolha aleatória e MAS-Cloud+, com cargas de tamanho 150k, 500k, 1M. As três barras à esquerda apresentam a escolha aleatória em diferentes instâncias (1, 8 e 96 VCPUs), enquanto as demais as escolhas do MAS-Cloud+, com os três modelos de raciocínio (heurístico, otimização, metaheurístico).

5.2.3 Resultados com Categorização da Aplicação

Uma vez que o MAS-Cloud+ adota uma abordagem independente de aplicação, faz-se necessário realizar execuções iniciais para compreender o comportamento da aplicação *MASA-OpenMP*. Na Figura 5.4 apresentam-se o tempo de execução e o custo destas execuções. O método para a criação do histórico da aplicação, apresentado em [Ralha et al., 2019], inclui a necessidade de 13 execuções para realizar uma predição adequada.

Neste trabalho foi comparado o MAS-Cloud+, utilizando as 13 execuções e a categorização da aplicação com especificação da variável de entrada *ApplicationCategory*, conforme apresentado na Tabela 4.2, Seção 4.2 de detalhamento da arquitetura. Com a categorização, faz-se necessário cinco execuções iniciais da aplicação para executar o modelos de predição (RLM). No modelo de [Ralha et al., 2019], o maior custo ocorria com a sequência de DNA de 1M, com uma média de $62,16$ minutos e USD $\$58,20$ (incluindo as 13 execuções iniciais). No modelo com categorização, o maior custo também ocorre com a sequência de DNA de 1M, mas com uma média de $27,05$ minutos e USD $\$38,34$ (incluindo as 5 execuções iniciais). Considerando a sequência de DNA de 1M, com uso

da categorização da aplicação, houve uma redução de $43,5\%$ no tempo de execução e $\approx 65,87\%$ no custo.

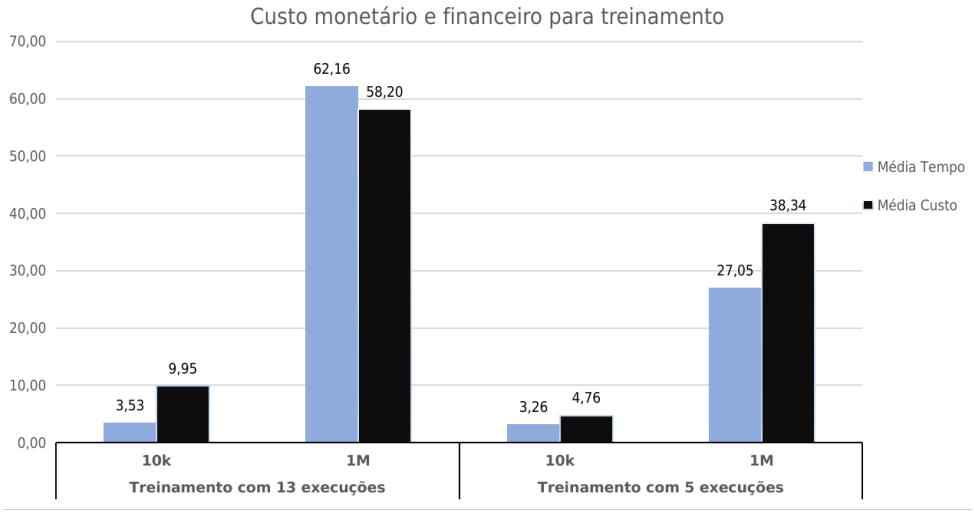

Figura 5.4: Tempo de execução e custo para criação do histórico da aplicação no MAS-Cloud+, com e sem categorização da aplicação.

5.2.4 Resultados e Análise de Predição

Para melhor expressar as variáveis da Equação 4.1 do modelo RLM (Seção 4.2.2), aplicou-se uma transformação logarítmica, conforme a Equação 5.1. Houve necessidade de transformar os valores das variáveis de volta para o sistema decimal, a fim de que as regras do VMMgr interpretem os valores, consoante a Equação 5.1. Na Tabela 5.6, apresentam-se os valores comparativos entre os modelos com e sem transformação logarítmica.

$$TimeLog = \log_{10}(T1) + \log_{10}(T2) * \log_{10}(QM) \ + \log_{10}(T3) * \log_{10}(QC)$$

$$TempoEstimado = 10^{TimeLog} \tag{5.1}$$

A Figura 5.5 ilustra gráficos de variância do modelo de RLM com 65 observações da média de tempo de predição com e sem transformação logarítmica. Nos histogramas de tempo de predição, o eixo x apresenta o tempo variando de zero a 50 minutos (a) e de zero a 4 minutos (c), e o eixo y com a contagem de frequência de zero a 50. Os gráficos quantil-quantil exibem os quantis teóricos no eixo x, variando de -2 a 2, e os quantis amostrais no eixo y, mudando de zero a 50 (b) e de 0,5 a 4,0 (d). Os dados distribuídos aparecem aproximadamente como uma linha reta nos gráficos Q-Q, embora as extremidades comecem a se desviar da linha reta. A transformação logarítmica dos dados visa abordar possíveis problemas de heteroscedasticidade e não linearidade nos

resíduos do modelo de regressão linear. Note-se que as transformações nos gráficos Q-Q mantêm as propriedades estatísticas, justificando seu uso.

Tabela 5.6: Comparação do R^2 para transformações de variáveis da RLM.

RLM	**R^2 com transformação logarítmica**	**R^2 sem transformação logarítmica**
Tempo de execução	98.65%	94.22%
Uso da CPU	96.85%	93.43%
Uso de memória	84.28%	93.74%

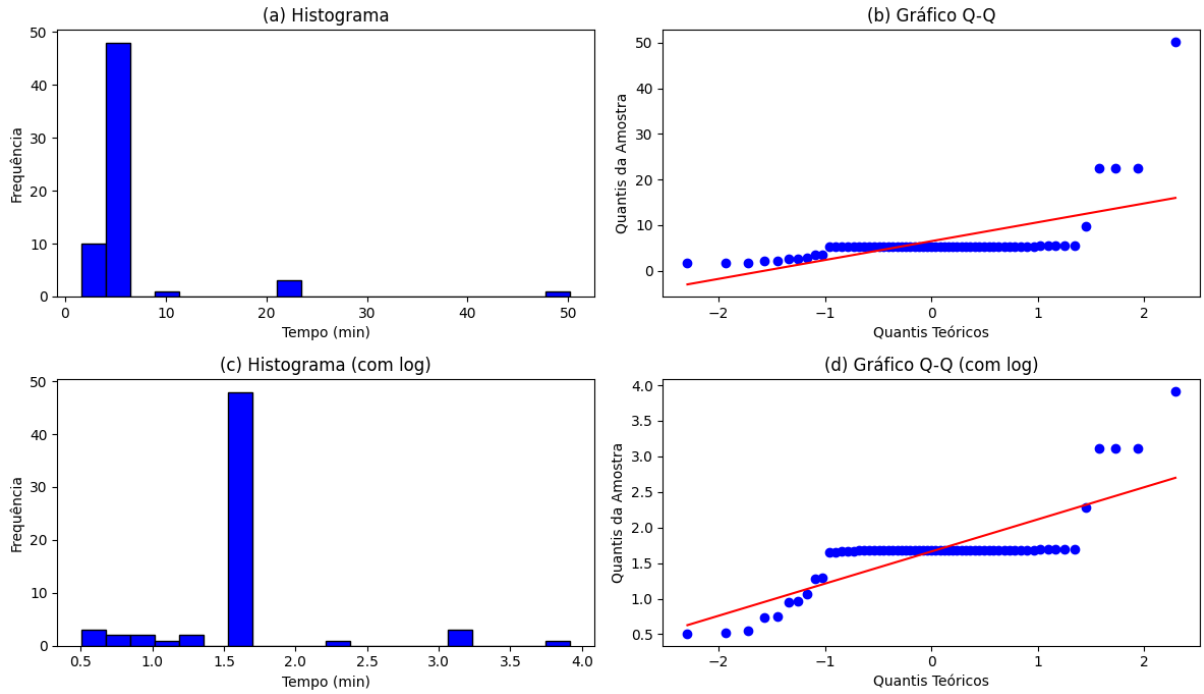

Figura 5.5: Distribuição dos dados de uso médio de tempo de execução.

Realizaram-se análises residuais para verificar a precisão do modelo de RLM. Na Figura 5.6, apresentam-se técnicas de análise de erros comparando valores reais e previstos. Na Figura 5.6 (a), mostra-se a ausência de relação entre as observações, que estão distribuídas de forma aleatória. Na Figura 5.6 (b), dispõem-se as observações sobre a linha de distribuição normal, com pequenas variações no início e no término das caudas, mostrando que a regressão é precisa para valores pequenos a médios. Os últimos três gráficos da Figura 5.6 (c, d, e), revelam os resíduos distribuídos de forma aleatória sobre o eixo x (linha vermelha), indicando que os resíduos se distribuem aleatoriamente, entre as variáveis quantitativas de memória e CPU. Assim, valida-se o modelo de RLM, uma vez que se distribuem os erros de maneira independente, com uma média próxima de zero.

Para demonstrar os valores dos resultados da análise de erros, são utilizado os dados de 10k e 1M da sequencia de DNA da Bioinformática. Os tamanhos distintos facilitam

a visualização do comportamento preditivo com cargas de trabalho variadas. A análise dos dados fornece informações significativas sobre o desempenho da . Os valores de R^2 mantêm-se geralmente elevados, oscilando entre $0,96$ e $0,99$, indicando uma excelente capacidade de explicar a variação dos dados (Figura 5.7). O MSE apresenta-se baixo, variando entre $0,006$ e $0,06$, e o MAE também apresenta valores baixos, de $0,04$ a $0,20$ (Figura 5.8). Esses resultados demonstram um ajuste eficiente do modelo aos dados observados.

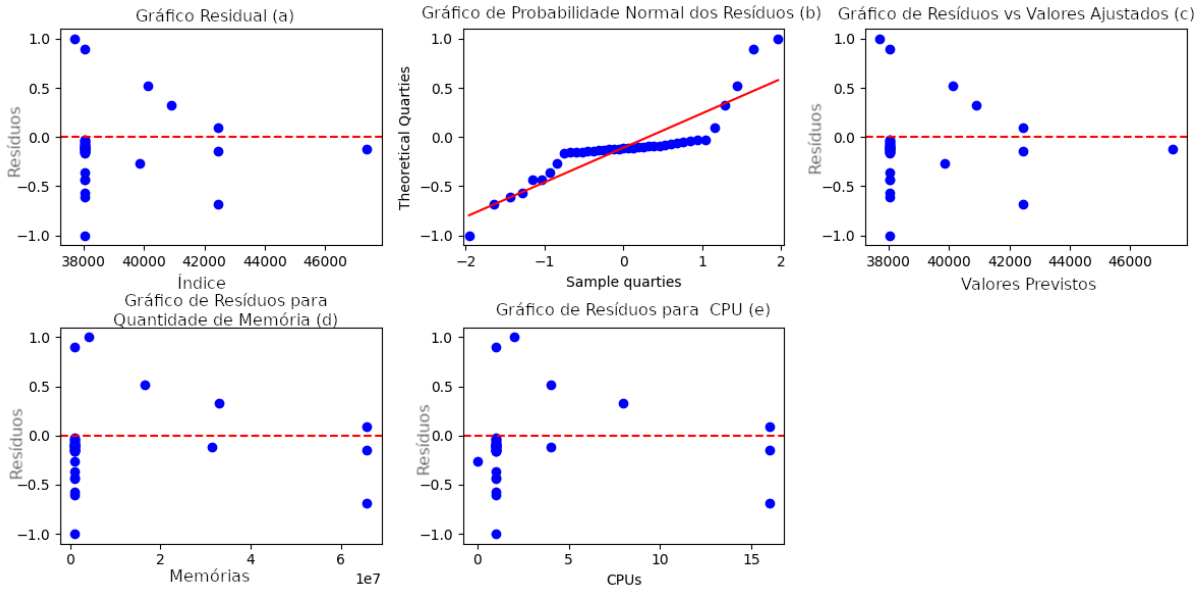

Figura 5.6: Análise residual do modelo de RLM.

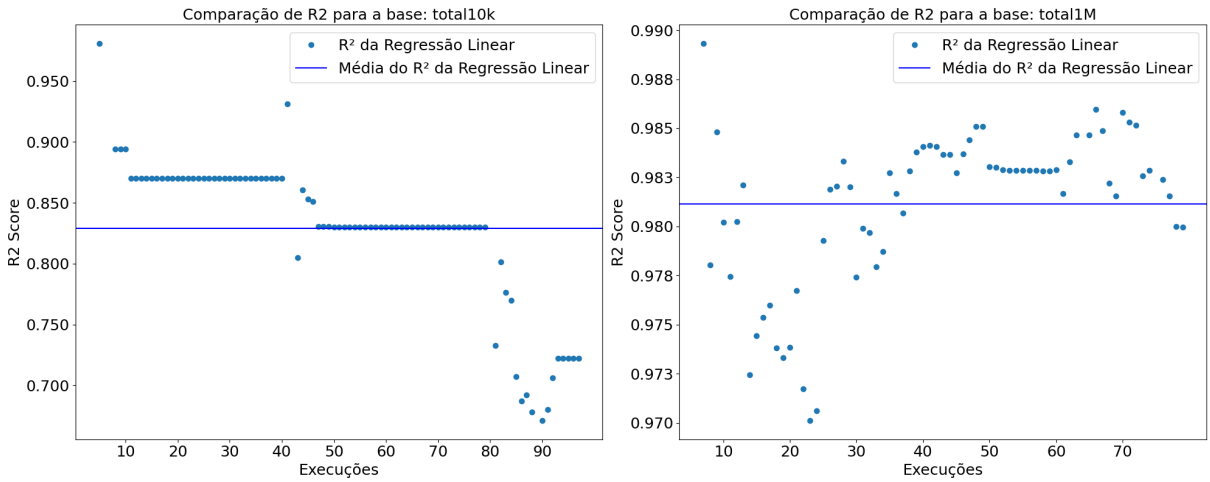

Figura 5.7: Variações dos valores de R^2 para a RLM.

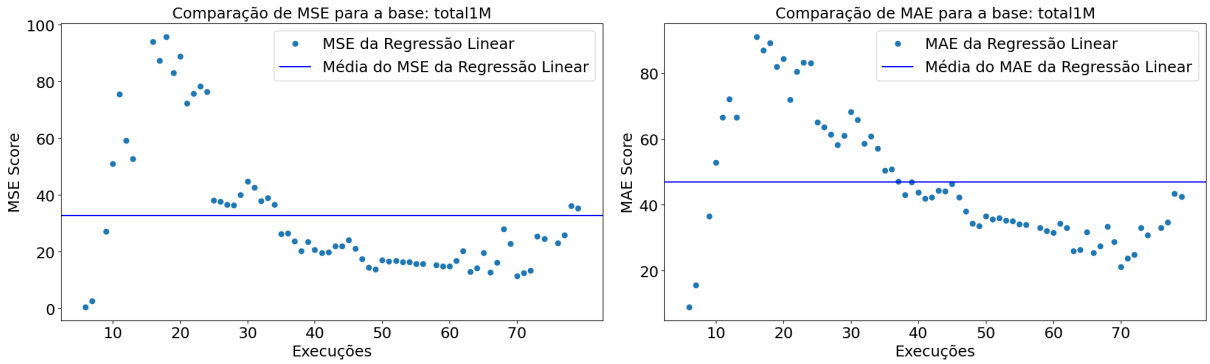

Figura 5.8: Variações dos valores de MSE e MAE para RLM.

Analisando os resultados em termos percentuais, observa-se que a RLM exibe um desempenho consistente, com R^2 médio aproximado de 97% e MSE médio em torno de $0,03$. O MAE médio, situa-se próximo de $0,13$. Estes valores refletem uma capacidade robusta de predição dos dados observados.

A análise das métricas para os modelos da RLM aplicados às bases de 10k e 1M, conforme a Figura 5.8, revela diferenças notáveis em termos de desempenho. Nas sequências de DNA de 10k, o erro MSE é de aproximadamente $14,68$, enquanto para as sequências de DNA de 1M é de cerca de $0,000014$. Similarmente, o MAE também diminui, de $14,95$ das sequências de DNA de 10k, para $0,0259$ nas sequências de 1M. Esta redução no MSE e MAE, sugere uma precisão maior nas predições para aplicações que requerem um tempo maior de execução, uma vez que na base de sequências de 10k, o tempo de execução é reduzido, dificultando o ajuste do modelo de predição.

Estas métricas normalizadas reforçam a conclusão de que o desempenho do modelo de melhora significativamente com um volume maior de dados, demonstrando o valor de se ter um grande conjunto de dados para treinar modelos de aprendizado de máquina, resultando em previsões mais precisas e um ajuste de modelo mais confiável.

Ao analisar a Figura 5.9, que compara os valores preditos, usando a RLM, com os valores reais, pode-se observar que a linha de regressão linear representada em vermelho se ajusta aos pontos de dados reais. Inclusive, com mais precisão com sequencias de 1M do que com 10k.

Considerando-se esses aspectos, conclui-se que a RLM se adapta as base de dados utilizadas neste trabalho, demonstrando uma taxa de explicação elevada com erros médios baixos e um bom desempenho geral. Contudo, é crucial destacar que os resultados podem sofrer variações em função do conjunto de dados específico e da natureza do problema analisado. Neste caso, outras aplicações deveriam ser experimentadas para aprofundamento de investigação do modelo preditivo utilizado.

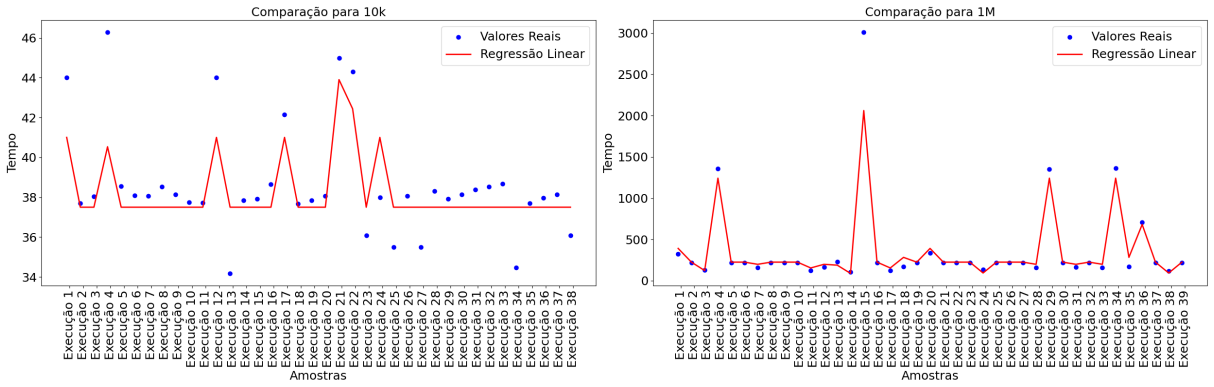

Figura 5.9: Comparação dos valores reais e o modelo de predição com RLM para o *MASA-OpenMP* (10k e 1M).

5.2.5 Resultados e Análise de Provisionamento

Neste trabalho avaliaram-se as escolhas do agente VMMgr na fase de provisionamento da arquitetura MAS-Cloud+, com relação às configurações das MVs. Como mencionado na Seção 4.2, as MVs escolhidas devem minimizar as violações de SLA. Dessa forma, analisaram e compararam-se as taxas de violação do SLA entre os modelos de raciocínio do VMMgr. Utilizaram-se as Equações 4.4 a 4.7 para avaliar as escolhas de MVs, para executar a ferramenta *MASA-OpenMP*, com sequencias de DNA de diversos tamanhos. O tempo de escolha das MVs usado nos modelos de raciocínio não faz parte dos resultados da Tabela 5.7. As instâncias de MVs apresentadas na tabela, revelam que a meta-heurística GRASP, tem variações notáveis nas configurações de instâncias para cada carga de trabalho.

Tabela 5.7: MVs escolhidas por cada modelo de raciocínio.

Heurístico					
10k	30K	50k	150k	500k	1M
t2.micro	r4.xlarge	t2.micro	r4.xlarge	m5.4xlarge	m5.4xlarge
				r5ad.8xlarge	
Otimização					
10k	30K	50k	150k	500k	1M
r4.xlarge	r5ad.8xlarge	t2.micro	r4.xlarge	r5ad.8xlarge	r5n.12xlarge
Meta-heurístico					
10k	30K	50k	150k	500k	1M
t2.micro	t2.micro	m4.large	r4.xlarge	m5.4xlarge	m4.10xlarge
	m4.large	r4.xlarge	m5.4xlarge	m4.10xlarge	m5.24xlarge
	r4.xlarge				
	m5.4xlarge				

Na Tabela 5.8 apresentam-se os valores limites acordados para a execução de cada carga de trabalho. Na coluna Utilização de Recursos (%), exibe-se a soma das percentagens de uso de memória e CPU das MVs. O Tempo de execução (ms) em milissegundos, indica o máximo que o usuário deseja que a aplicação termine a execução. O Preço (USD/hora) apresenta o limite de valor que o usuário planeja pagar pela execução. Definem-se os limites de SLA, utilizados na aplicação de Bioinformática, calculando-se a média das 13 execuções iniciais, conforme a Seção 5.1. Utilizam-se esses valores de limiar de SLA, ao final das execuções da aplicação, para calcular a taxa de violação e qualificar as escolhas para cada modelo de raciocínio.

Tabela 5.8: Limites de SLA usados nos experimentos da aplicação *MASA-OpenMP*.

Cargas de trabalho	**Utilização de Recursos (%)**	**Tempo (ms)**	**Preço (USD/hora)**
10k	183	67250	0.17
30k	198	49316	0.57
50k	146	56409	0.18
150k	142	72745	0.12
500k	108	386717	20.0
1M	98	494892	33.0

Nas Figuras 5.10, 5.11 e 5.12, o Eixo X apresenta as cargas de trabalho submetidas e o Eixo Y os valores das taxas de violação de SLA. Em relação às barras, a barra vermelha representa o limite da taxa de violação de SLA, acordado com o usuário. Representa-se a taxa de violação da meta-heurística pela barra verde. A coluna azul representa a taxa de violação do modelo de otimização, e a barra preta representa o modelo heurístico.

Ressalta-se que as taxas de violação de SLA demonstram uma proximidade entre os três modelos de raciocínio. Assim, há um balanceamento entre os modelos de escolha do agente. O modelo de otimização, em comparação com a escolha aleatória ou a não otimização, possui, em média, tempos de execução abaixo de 2.000 segundos ($\approx$ 33 minutos). Utilizam-se as Equações 5.2 e 5.3 para verificar o desempenho e o custo dos três modelos de raciocínio (heurístico, otimização e meta-heurístico). Essas equações calculam a diferença entre os modelos de raciocínio considerando o tempo médio e o custo de 30 execuções, em que AVG é a média, E é a execução, N é o número total de execuções, e x e y representam o modelo calculado considerando tempo de execução ou custo.

$$AVG_x = \frac{\sum_{x=1}^{N} E_x}{N} \tag{5.2}$$

$$diferença = \frac{AVG_x}{AVG_y} \tag{5.3}$$

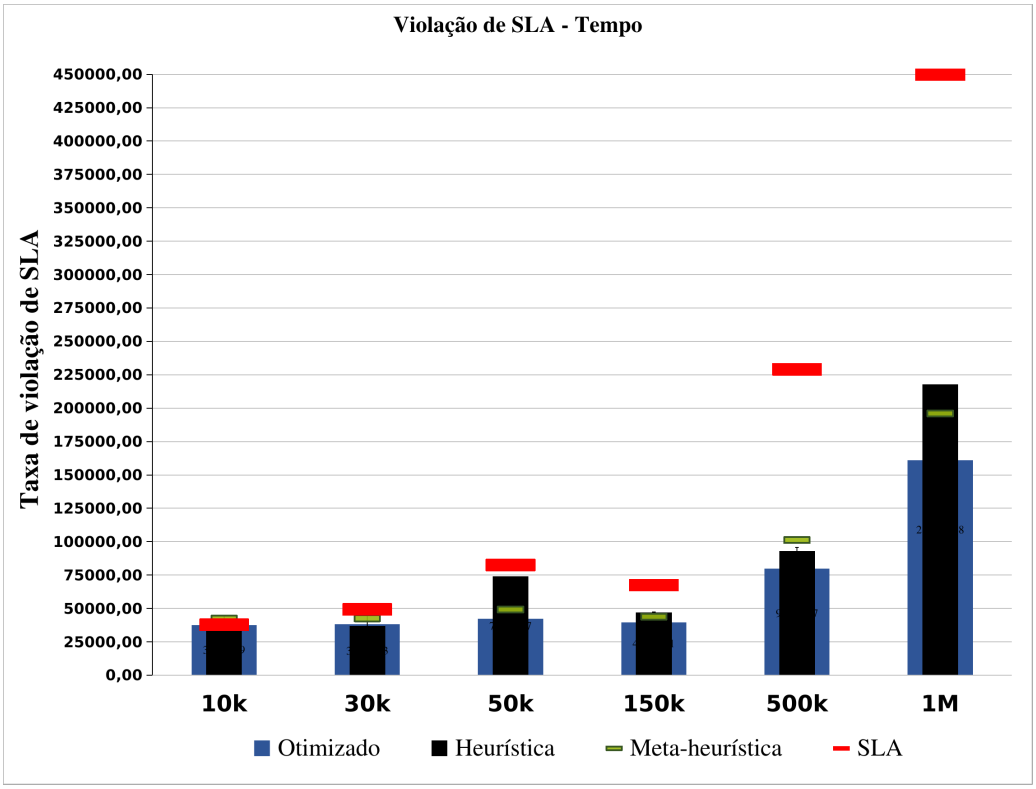

Figura 5.10: Taxas de violação do tempo de SLA nos resultados de provisionamento.

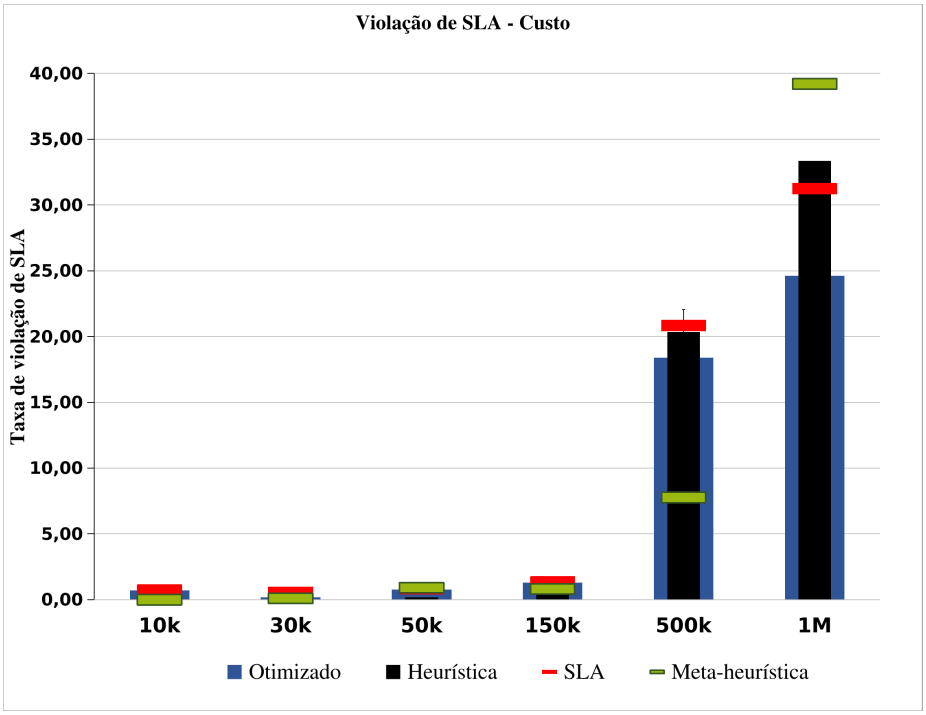

Figura 5.11: Taxas de violação do custo de SLA nos resultados de provisionamento.

Ao avaliar as execuções do protótipo da arquitetura MAS-Cloud+ na AWS EC2 com os valores mostrados na Tabela 5.9, obtiveram-se os seguintes resultados médios:

- a otimização apresenta desempenho $23,50\%$ superior à heurística e $19,27\%$ superior à meta-heurística;
- a heurística possui custo $66,39\%$ superior ao da otimização e $62,34\%$ superior à meta-heurística;
- a meta-heurística possui desempenho $3,55\%$ superior à heurística, no entanto, a heurística possui custo $62,34\%$ inferior à meta-heurística.

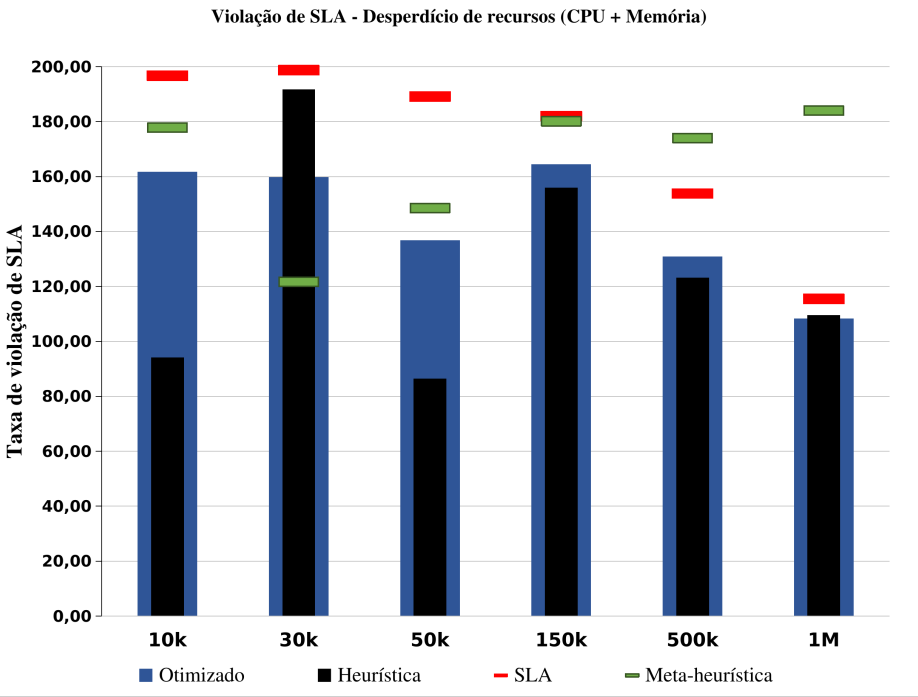

Figura 5.12: Taxas de violação de desperdício de recursos (CPU e memória) de SLA nos resultados de provisionamento.

Tabela 5.9: Comparação dos resultados, (a) médias totais de tempo de execução e custo para todas as cargas de trabalho, e (b) diferenças entre os modelos de raciocínio.

a	**heurístico**	**Otimização**	**Meta-heurístico**
Tempo médio	810,91s	656,56s	783,09s
Custo médio	$1,76	$2,65	$2,82
b	**heurístico/ Otimização**	**Meta-heurístico/ Otimização**	**heurístico/ Meta-heurístico**
Tempo	**23,50%**	**19,27%**	**3,55%**
Custo	**66,39%**	**6,51%**	**62,34%**

5.2.6 Resultado Escolha Automática

Foram realizados experimentos para automatizar a seleção dos três modelos de raciocínio (heurístico, otimização e meta-heurístico) no MAS-Cloud+, considerando o tempo de execução e o tempo de escolha da MV em cada modelo. Embora cada modelo busque balancear o tempo de execução e custo, a seleção do modelo baseou-se no tempo total e no custo da execução. Calculou-se um índice de custo-benefício considerando os tempos totais e custos, e o modelo escolheu o melhor modelo com base nesse índice. Os resultados obtidos na análise comparativa dos modelos, revela diferenças significativas em termos de custo, tempo total e relação custo-benefício. Assim, são apresentadas nas Figuras 5.13 (para 10k) e 5.14 (para 1M), exemplos das escolhas, submetidos ao *MASA-OpenMP*. Para uma melhor compreensão, enfocou-se a comparação entre o modelo automatizado e os demais modelos utilizados individualmente.

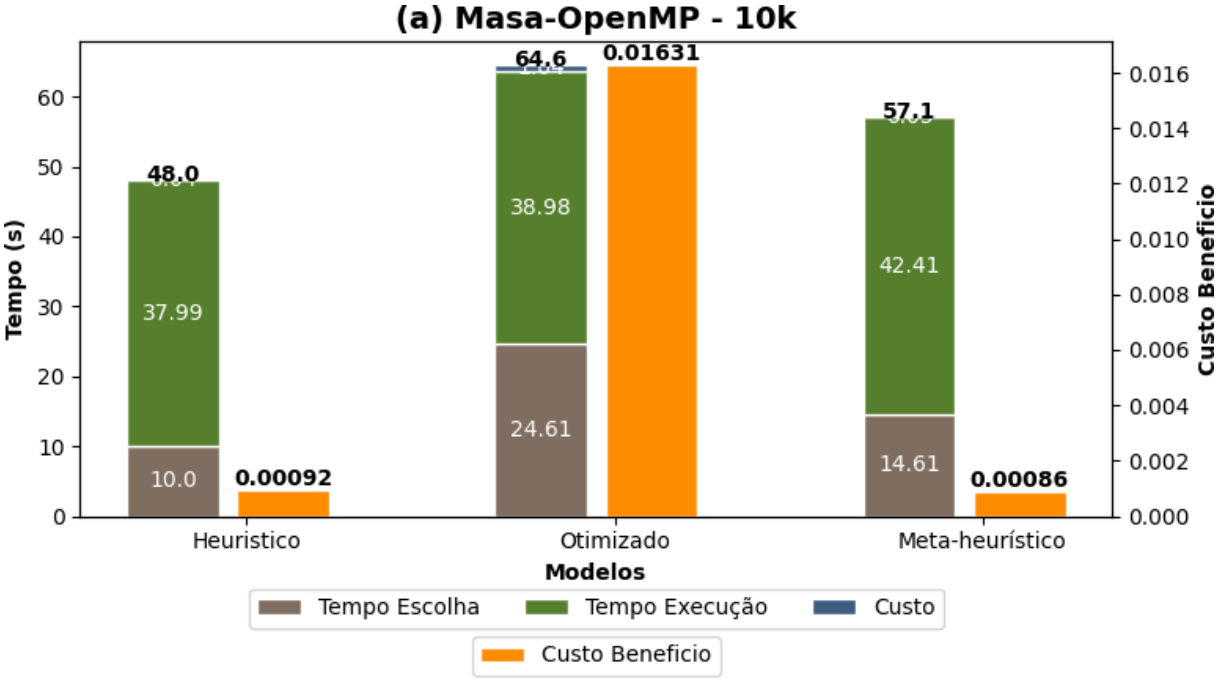

Figura 5.13: Comparação dos modelos de raciocínio distintos na Bioinformática (10k).

Na Figura 5.15 apresenta-se uma análise comparativa de execuções com dois tamanhos diferentes de sequências de DNA (utilizou-se 10k e 1M), submetidas pelo usuário em diferentes momentos e, então, elencam-se os modelos de raciocínio do agente. Cada agrupamento de três barras nos gráficos representa uma submissão do usuário (Execução), e o modelo escolhido refere-se àquele que possui o melhor custo benefício para o usuário. Na figura, a barra azul é o tempo total (escolha mais execução) do modelo heurístico, a barra cinza é o tempo total (escolha mais execução) da otimização, e a barra rosa é o tempo

total (escolha mais execução) do modelo meta-heurístico. O triângulo preto refere-se ao custo benefício calculado de cada modelo, em cada execução, o triângulo verde, o melhor custo benefício. Observa-se que diferentes execuções podem se beneficiar de modelos distintos. Quando o tempo de execução é semelhante para os três modelos de raciocínio do agente e com um custo financeiro inferior, o tempo de escolha se torna mais relevante, pois terá um impacto maior no custo benefício final, dando uma melhor experiência para o usuário.

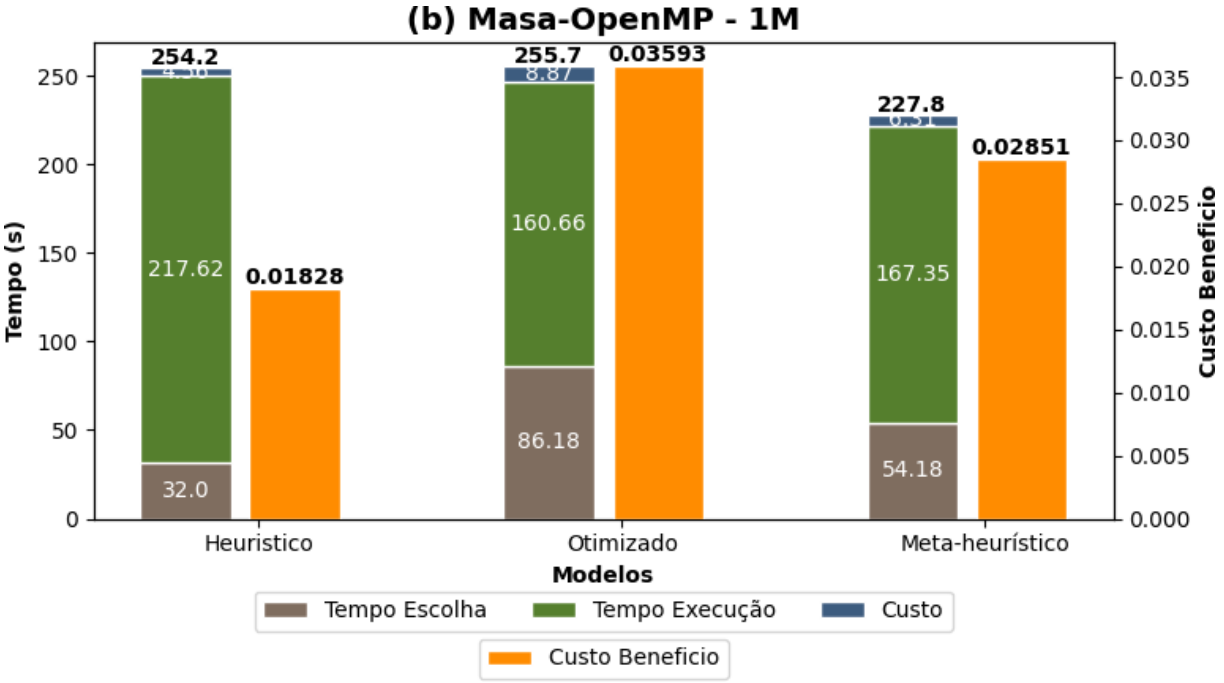

Figura 5.14: Comparação dos modelos de raciocínio distintos na Bioinformática (1M).

Na Figura 5.16 de 10k, os resultados para dez execuções também apontam para um melhor desempenho do modelo automatizado em comparação aos modelos de raciocínio heurístico, otimização e meta-heurístico. O modelo automatizado registrou a menor média de custo, com $\$0,005$, enquanto o modelo heurístico apresentou o custo mais alto, alcançando $\$0,07$. O otimizado registrou uma média de custo de $\$0,89$; enquanto o meta-heurística alcançou uma média de custo de $\$0,97$. Da mesma forma, o modelo automatizado mostrou uma vantagem significativa em relação a média do tempo de execução de 41,62 segundos, enquanto os outros modelos apresentaram médias entre 35 e 37 segundos.

Em termos, apresentado na Figura 5.16 de relação custo-benefício, o modelo automatizado apresentou uma relação custo-benefício extremamente baixa, registrando um valor de $0,001$. Por outro lado, a heurística registrou uma relação custo-benefício de $0,02$; enquanto o otimizado e a meta-heurística registraram valores de $0,25$ e $0,26$, respectiva-

mente. Isso significa que o modelo automatizado obteve uma melhora de 95% na relação custo-benefício da heurística, 99, 6% na relação custo-benefício do otimizado e 99, 615% na relação custo-benefício da meta-heurística.

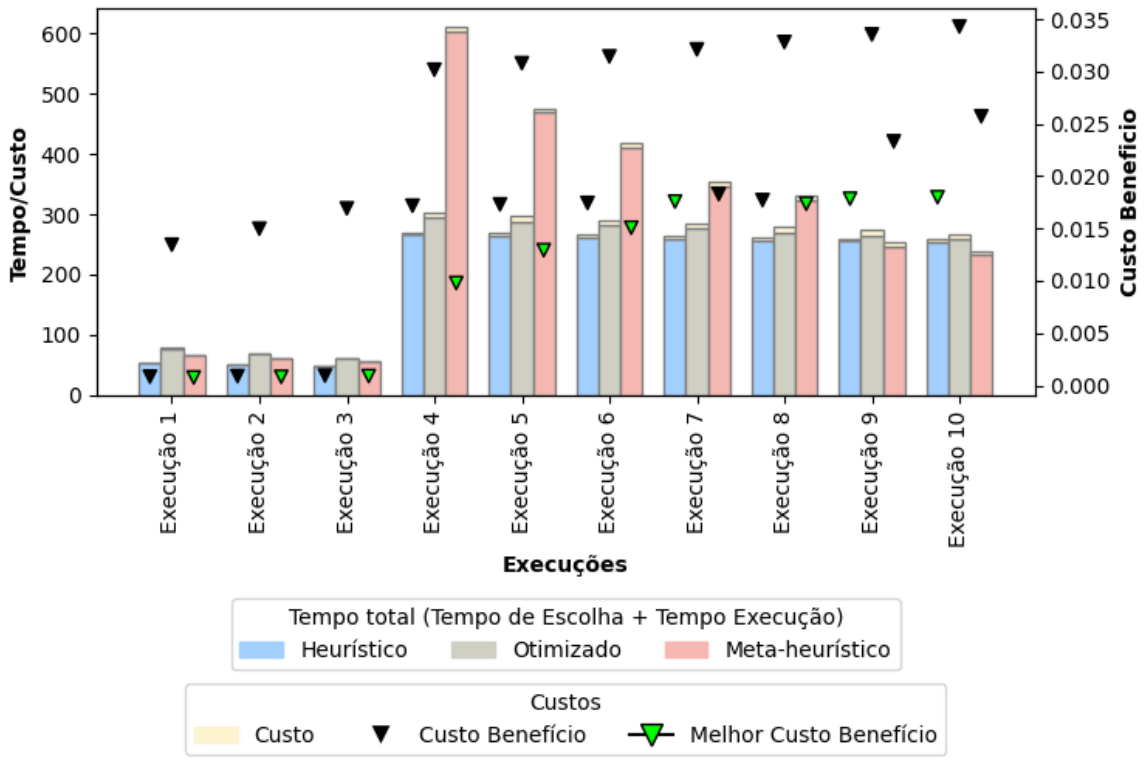

Figura 5.15: Comparação de custo acumulado de dez execuções utilizando dois tamanhos diferentes de sequências de DNA (10k e 1M) com MAS-Cloud+ na AWS EC2.

Ao avaliar a base de dados de 1M (Figura 5.17), nota-se que o modelo automatizado apresentou uma média de custo de $4, 85; enquanto a heurística registrou uma média de custo superior, alcançando $5, 56. Por sua vez, o otimizado mostrou uma média de custo mais elevada, atingindo $7, 33. No entanto, o modelo automatizado não superou tanto o otimizado quanto a heurística em termos de média do tempo de execução. Enquanto o modelo automatizado obteve uma média de 230 segundos, a heurística e o modelo otimizado registraram médias de 217, 5 e 151, 63 segundos, respectivamente.

A relação custo-benefício também deve ser considerada nessa análise. O modelo automatizado apresentou uma relação custo-benefício de 0,21, indicando uma eficiência significativa na utilização de recursos em comparação aos outros modelos. A heurística também registrou uma relação custo-benefício de 0,26, demonstrando ligeiramente maior. No entanto, o modelo otimizado mostrou uma relação custo-benefício mais elevada, alcançando

0,48. Isso significa que o modelo automatizado obteve um resultado 56, 25% melhor em termos de relação custo-benefício em comparação ao modelo otimizado.

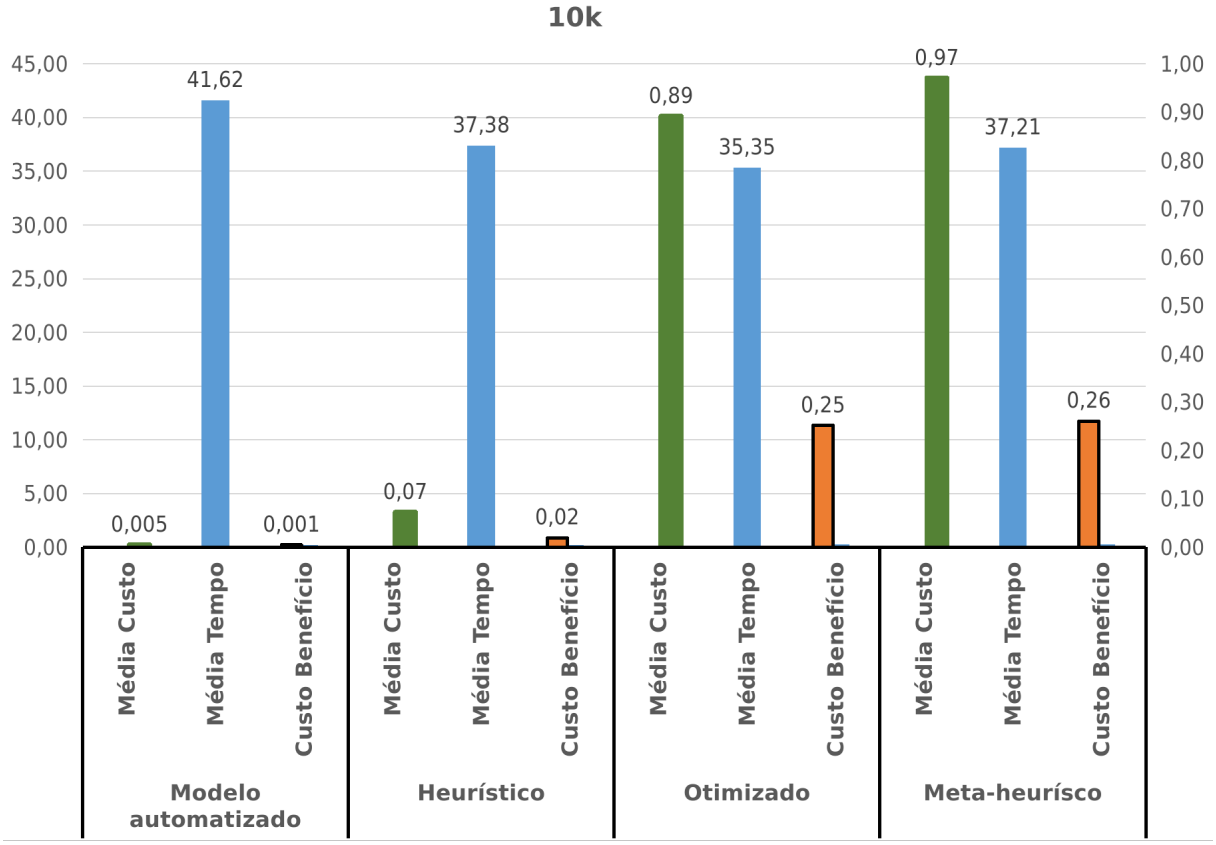

Figura 5.16: Comparação do modelo de escolha automática (10k).

Em conclusão, os resultados obtidos revelam que o modelo automatizado, que combina a abordagem heurística, otimizada e meta-heurística, apresenta um desempenho superior em termos de tempo de execução, custo e relação custo-benefício, quando comparado aos modelos individuais. Essa combinação estratégica permite obter resultados mais eficientes, com uma melhor utilização de recursos e uma maior vantagem econômica.

5.3 Aplicação com *Apache Spark Benchmark*

Um estudo comparativo com o trabalho de Islam et al. [2022] foi realizado, em que os autores desenvolveram experimentos em plataformas de nuvem híbrida real, além de experimentos simulados. Os experimentos executados nas plataformas de nuvem avaliam o desempenho de três tipos de aplicações, usando *BigDataBench* Wang et al. [2014]. Neste trabalho foram utilizadas as mesmas aplicações com *WordCount* (intensivo em CPU),

Sort (intensivo em memória) e *PageRank* (intensivo em rede), misturados aleatoriamente para gerar a carga de trabalho.

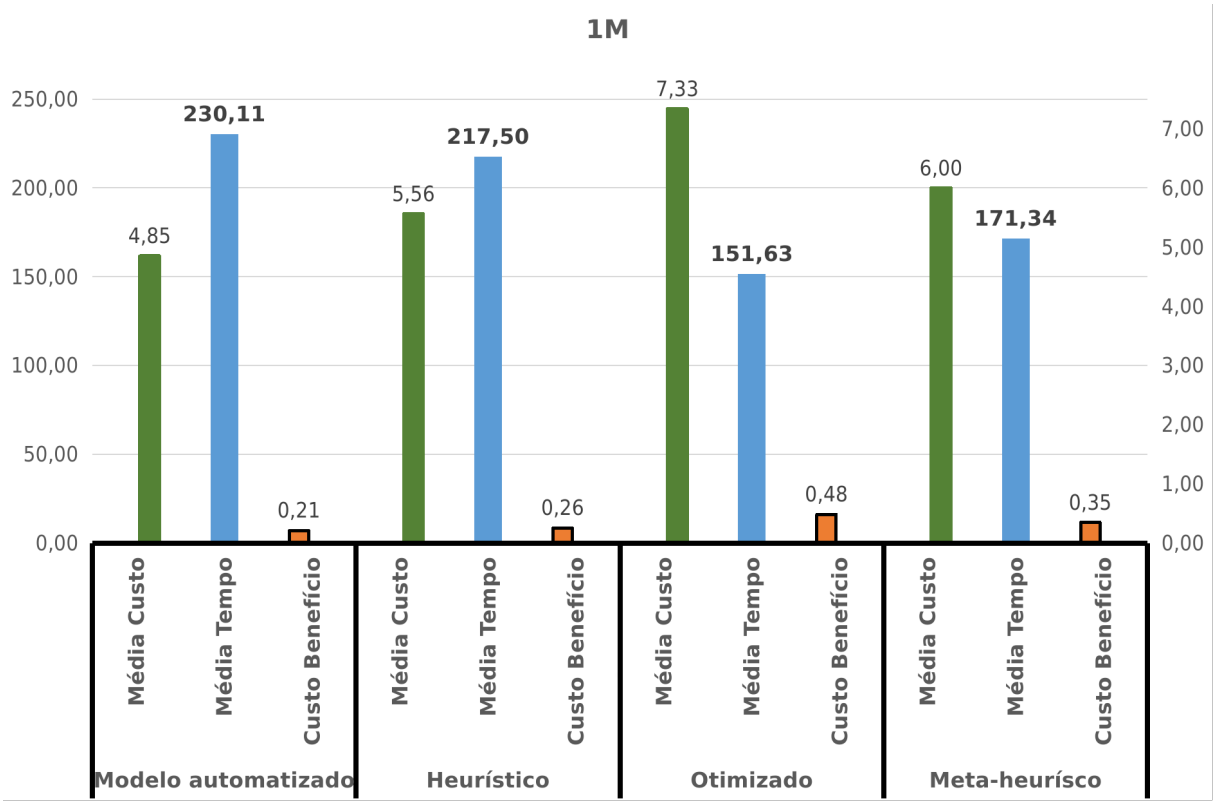

Figura 5.17: Comparação do modelo de escolha automática (1M).

O trabalho de Islam et al. [2022] utilizou quatro algoritmos de escalonamento: *First In, First Out* (FIFO), *First In First Out Consolidate* (FIFO-C), *Adaptive-Scheduling-with-QoS-Satisfaction* (AsQ) Wang et al. [2013], e Programação Linear Mista Inteira - *Mixed-Integer Linear Programming* (MILP). Além disso, os autores propuseram dois algoritmos de escalonamento: FF e GIO. Comparou-se o MAS-Cloud+ com os resultados dos algoritmos de escalonamento propostos, calculando a média de 100 execuções. Os resultados executados na AWS EC2 e na *Google Cloud* estão nas Seções 5.3.1 e 5.3.2, respectivamente.

Na análise qualitativa do experimento comparativo, valores abaixo de 30% são atribuídos como qualidade baixa, entre 30% e 50% como qualidade moderada, acima de 50% como média, e acima de 80% como alta qualidade. Essa atribuição de valores de qualidade visa oferecer uma comparação qualitativa dos resultados, permitindo uma avaliação padronizada do desempenho obtido.

5.3.1 Resultados Comparativos do AWS EC2

Avaliou-se o modelo de provisionamento do MAS-Cloud+ com as configurações escolhidas para as MVs, em comparação ao estudo de Islam et al. [2022], utilizando as mesmas instâncias da AWS apresentadas na Tabela 5.2. É essencial notar que as MVs selecionadas devem minimizar as violações do SLA, balanceando o menor tempo de execução, custo e desperdício de recurso.

Na Figura 5.18 mostra-se o tempo médio de execução do MAS-Cloud+ com os três modelos de raciocínio para as aplicações *WordCount*, *Sort* e *PageRank*. A duração média do trabalho do *WordCount* é de 85,5 segundos usando heurística, 84,2 segundos com meta-heurística e 83,2 segundos com otimização. A duração média do trabalho do *Sort* é de 272,5 segundos usando heurística, 252 segundos com meta-heurística e 262,3 segundos com otimização. A duração média do trabalho do *PageRank* é de 91,7 segundos usando heurística, 95,2 segundos com meta-heurística e 90,1 segundos com otimização (média aproximada de 93 segundos). O tempo médio de execução dos algoritmos de Islam et al. [2022] é de aproximadamente 220 segundos para o *PageRank*, representando uma redução de, aproximadamente, 58% para os modelos do MAS-Cloud+. Esses resultados são explicáveis, pois utilizaram-se as MVs para a execução do *WordCount* e o *PageRank* com base nos três tipos de instâncias da Tabela 5.2; enquanto o *Sort* utilizou, apenas, as instâncias mais baratas (t2.xlarge e t2.2xlarge, com 4 e 8 vCPUs, respectivamente).

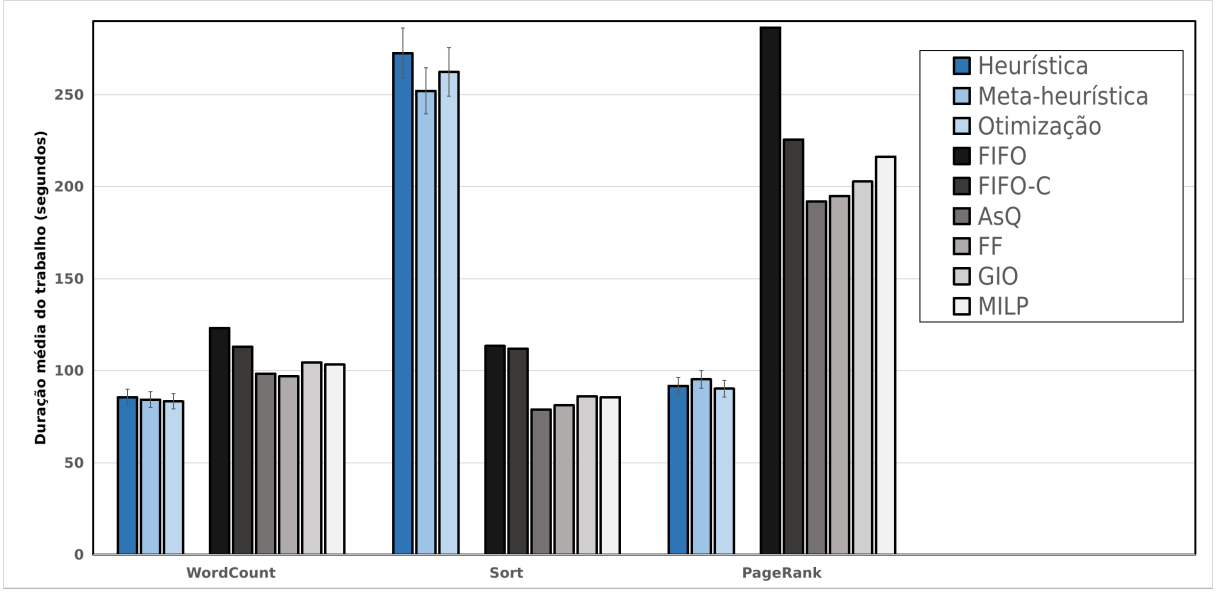

Figura 5.18: Tempo médio de execução na AWS EC2 com os modelos de raciocínio do MAS-Cloud+ (heurístico, meta-heurístico, otimização) e o trabalho de Islam et al. [2022] com algoritmos FIFO, FIFO-C, AsQ, FF, GIO e MILP utilizando três tipos de aplicação do *BigDataBench* – *WordCount*, *Sort* e *PageRank*.

Na Figura 5.19, ilustra-se o custo médio de execução das aplicações *WordCount*, *Sort* e *PageRank* em conjunto, em que os modelos do MAS-Cloud+ estão com USD $0,83, $0,76, e $0,74, para heurística, meta-heurística e otimização, respectivamente (média de USD $0,77). Os resultados de Islam et al. [2022], com os algoritmos FIFO, FIFO-C, AsQ, FF, GIO e MILP apresentam USD $2,47, $2,28, $2,05, $1,83, $1,6, e $1,86, respectivamente (média de USD $2,01). Portanto, os resultados do MAS-Cloud+ apresentam uma redução de custo médio de, aproximadamente, 58%.

O estudo comparativo revela que os modelos de raciocínio do MAS-Cloud+ (ou seja, heurístico, otimização, meta-heurístico) consistentemente buscam balancear tempo de execução, custo e desperdício, indicando resultados promissores em relação aos resultados do estudo comparativo com trabalho de Islam et al. [2022].

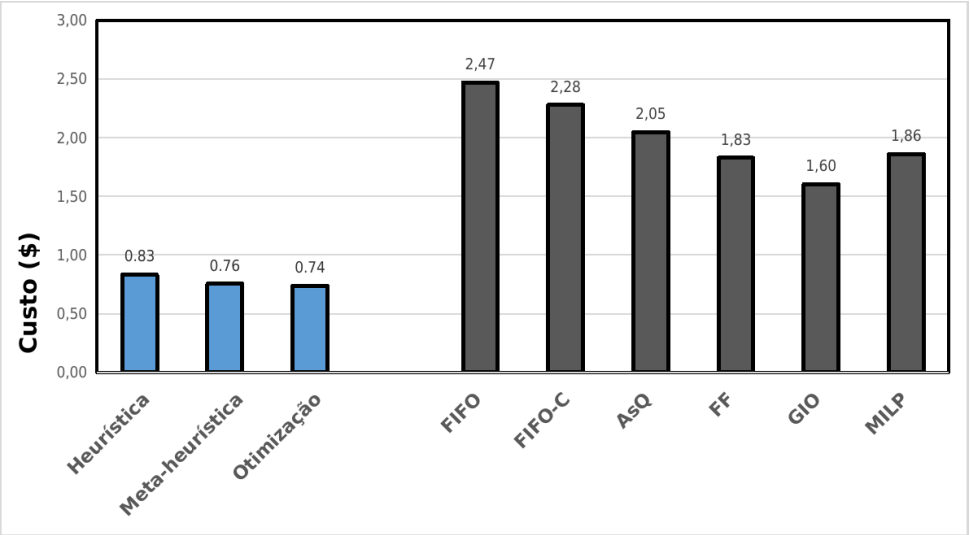

Figura 5.19: Custo médio de execução na AWS EC2 com os modelos de raciocínio do MAS-Cloud+ (heurístico, meta-heurístico, otimização) e o trabalho de Islam et al. [2022] com algoritmos FIFO, FIFO-C, AsQ, FF, GIO e MILP usando três tipos de aplicação do *BigDataBench* juntas (*WordCount*, *Sort* e *PageRank*).

5.3.2 Resultados Comparativos do *Google Cloud*

Os experimentos realizados na AWS foram todos repetidos no *Google Cloud*, como objetivo de avaliar os resultados do MAS-Cloud+ em diferentes plataformas de nuvem. Executaram-se as mesmas aplicações com as configurações de nuvem do Google apresentadas na Tabela 5.3, sendo estas instâncias iguais em vCPU e memória (GiB) as da Tabela 5.2 (AWS), variando somente nome e preço (USD/hora).

Na Figura 5.20, apresenta-se o tempo médio de execução do MAS-Cloud+ (barras azuis) com os três modelos de raciocínio para as aplicações *WordCount*, *Sort* e *PageRank*.

A duração média do trabalho do *WordCount* é de 85, 50 segundos usando heurística, 84, 26 segundos com meta-heurística e 83, 29 segundos com otimização. Para o *Sort*, a duração média é de 272, 57 segundos usando heurística, 252, 07 segundos com meta-heurística e 262, 34 segundos com otimização. Para o *PageRank*, é de 91, 76 segundos usando heurística, 95, 25 segundos com meta-heurística e 90, 17 segundos com otimização. As aplicações *WordCount* e *PageRank* ficam abaixo de 100 segundos.

Os resultados de tempo médio de execução do trabalho de Islam et al. [2022] no *Google Cloud* estão na Figura 5.20 (barras cinzas). O tempo médio do Islam et al. [2022] é de 127, 38 segundos para *WordCount*, 49, 28 segundos para *Sort* e 135, 04 segundos para *PageRank*, utilizando os algoritmos FIFO, FIFO-C, AsQ, FF, GIO e MILP. Observou-se que o MAS-Cloud+ apresenta uma redução de aproximadamente 32,88% no tempo de execução para *WordCount* e 33,23% para *PageRank*, mas um aumento de 532,30% *Sort*.

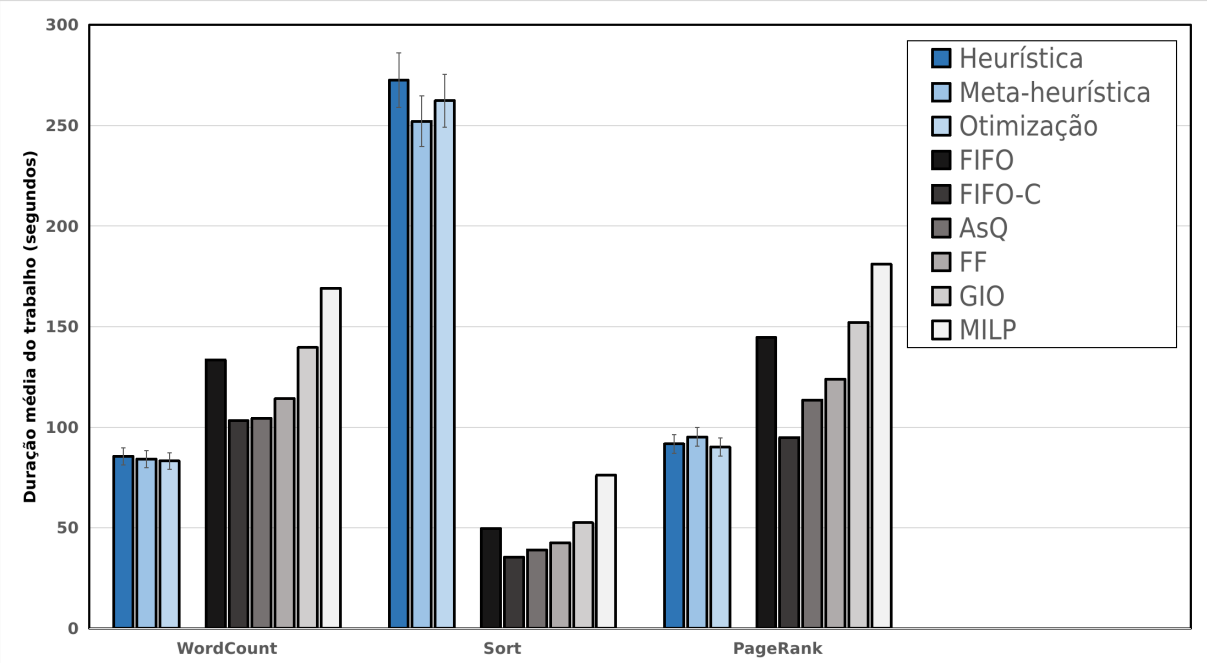

Figura 5.20: Comparação de tempo médio de execução do MAS-Cloud+ e Islam et al. [2022] na *Google Cloud*.

Na Figura 5.21, mostra-se o custo médio de execução das aplicações *WordCount*, *Sort* e *PageRank* juntas, em que o modelo MAS-Cloud+ tem um custo médio de USD $1, 02 para o *WordCount*, USD $0, 26 para o *Sort*, e USD $0, 06 para o *PageRank*. O modelo heurístico apresenta um custo de USD $2, 28 para *WordCount*, USD $0, 66 para *Sort* e USD $0, 07 para *PageRank*. O modelo meta-heurístico tem um custo de USD $0, 72 para *WordCount*, USD $0, 56 para *Sort* e USD $0, 05 para *PageRank*. O modelo de otimização apresenta um custo de USD $1, 31 para *WordCount*, USD $0, 06 para Sort e também

USD $\$0,06$ para *PageRank*. Já o modelo proposto por Islam et al. [2022], com algoritmos FIFO, FIFO-C, AsQ, FF, GIO e MILP, apresenta um custo médio de USD $\$0,60$ para *WordCount*, USD $\$0,52$ para *Sort* e USD $\$0,54$ para *PageRank*. Observa-se que o modelo heurístico apresenta um custo mais elevado em comparação aos demais modelos.

No âmbito da comparação de custos entre os modelos do MAS-Cloud+ (heurístico, meta-heurístico e otimização) e os algoritmos de Islam et al. [2022] (FIFO, FIFO-C, AsQ, FF, GIO e MILP), observa-se uma disparidade notável. O modelo heurístico do MAS-Cloud+ registra o custo mais elevado com USD $\$3,02$; enquanto o modelo meta-heurístico apresenta o menor custo com USD $\$1,33$. Em contraste, o FIFO-C tem o mais caro com USD $\$4,10$ e o MILP o mais barato com USD $\$0,93$. A média geral dos custos para os modelos MAS-Cloud+ é de aproximadamente USD $\$1,92$. Para o Islam et al. [2022] é de, aproximadamente, USD $\$2,25$. Isso revela que, em média, os modelos MAS-Cloud+ são cerca de $\$14,7\%$ mais baratos.

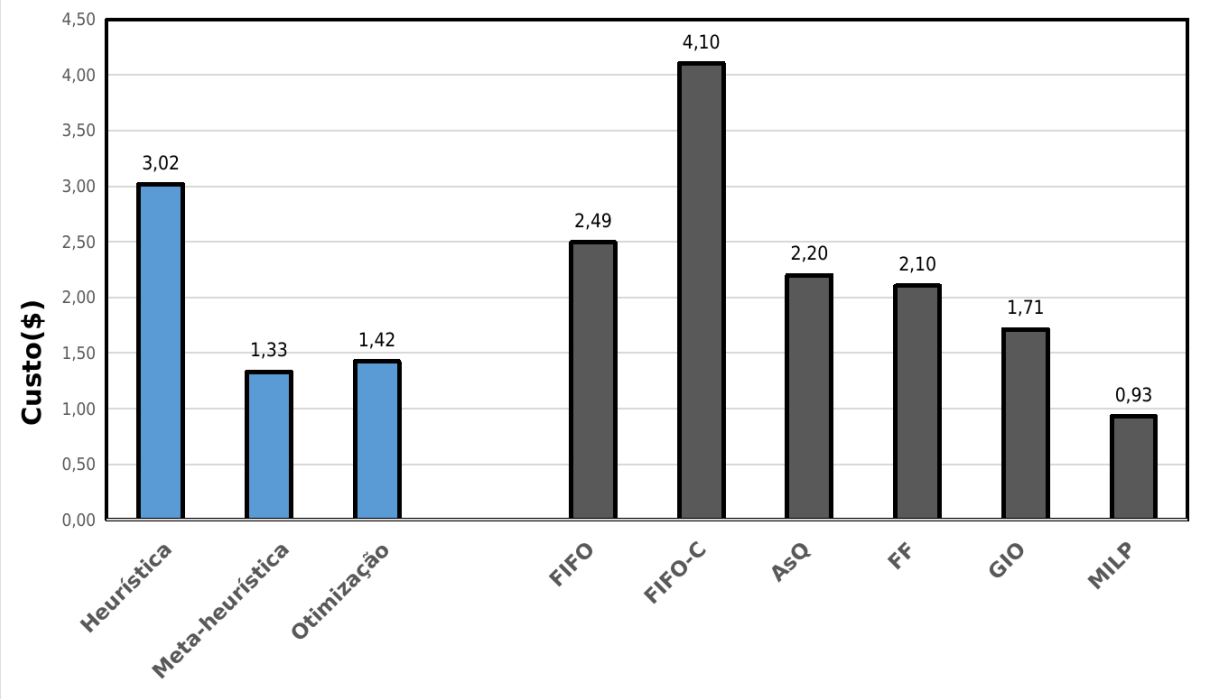

Figura 5.21: Comparação do custo médio de execução do MAS-Cloud+ e Islam et al. [2022] na *Google Cloud*.

O estudo comparativo revela que os modelos de raciocínio do MAS-Cloud+ (heurístico, otimizado, meta-heurístico), consistentemente, balanceiam o tempo de execução, o custo e o desperdício, indicando resultados promissores em relação ao trabalho de Islam et al. [2022].

5.3.3 Resultados da Escolha Automática

Os experimentos realizados para a escolha automática do modelo de raciocínio utilizaram uma base histórica de aplicações, considerando o *Apache Spark Benchmark*, com um total de 1500 execuções. Compararam-se os resultados obtidos com o estudo de Islam et al. [2022], proporcionando um referencial com o estado da arte. A análise do tempo de escolha em cada modelo é de suma importância, como evidenciado pela Figura 5.22. Essas figuras representam o tempo de escolha em diferentes modelos e o custo, ressaltando a relevância de considerar esse fator durante o processo de seleção do raciocínio.

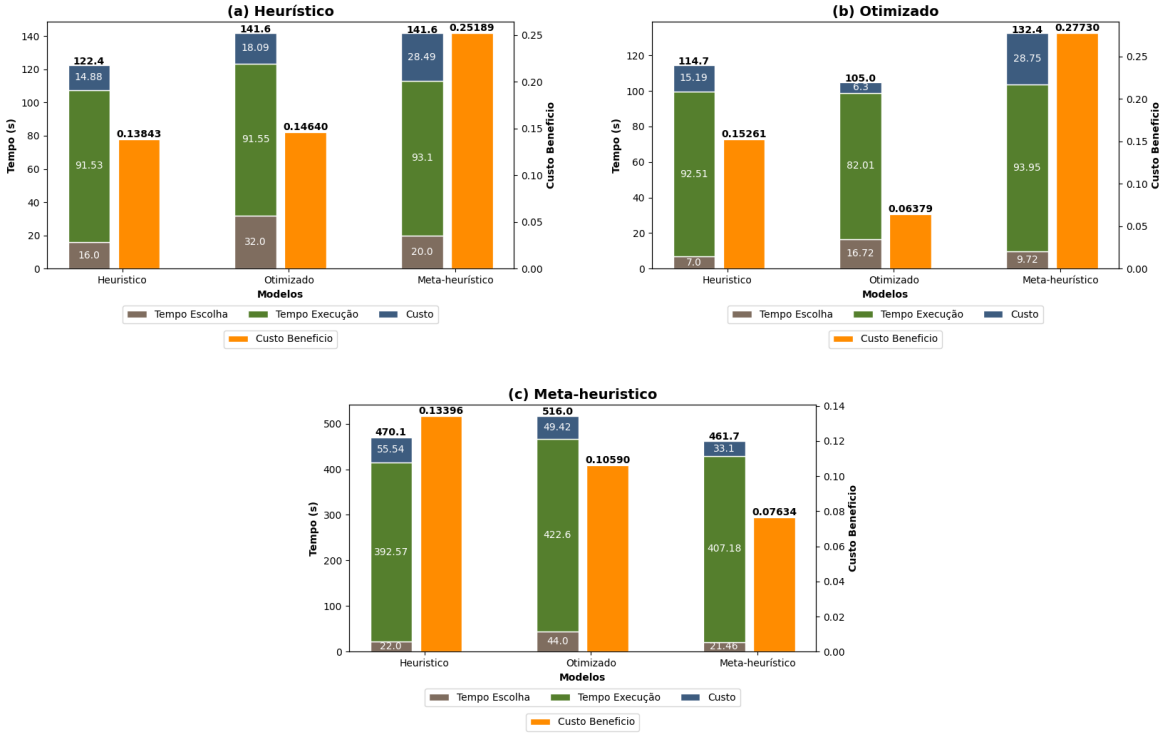

Figura 5.22: Escolhas dos modelos de provisionamento para execução das aplicações: (a) *Sort* foi selecionado o modelo heurístico, (b) *WordCount* foi selecionado o modelo otimizado, e *PageRank* foi selecionado o modelo meta-heurístico.

Para ressaltar a importância da análise do tempo de escolha em cada modelo, apresenta-se a Figura 5.22, em que utilizam-se os valores para calcular o índice de custo-benefício. Na Figura 5.22 (a) apresenta-se um exemplo em que o modelo heurístico é selecionado para executar a aplicação *Sort*. Isso ocorre porque a relação entre o tempo de execução e o custo resulta em um menor índice de custo-benefício do que a soma dos tempos dos outros dois modelos (heurístico: 16 segundos para escolha e 91 segundos para execução, totalizando $107,5$ segundos e um custo de $\$14,9$, resultando em um índice de custo-benefício de $0,14$). Na Figura 5.22 (b) para a execução da aplicação *WordCount*, utiliza-se o modelo

otimizado, que apresenta um tempo total menor em comparação à soma do tempo de escolha e execução (16 segundos para escolha e 82 segundos de execução, totalizando 98 segundos e um custo de $\$6,3$, resultando em um índice de custo-benefício de $0,07$). Por fim, na Figura 5.22 (c) elencou-se o modelo meta-heurístico para executar a aplicação *PageRank* ($21,5$ segundos para escolha e $407,2$ segundos para execução, totalizando $428,7$ segundos e um custo de $\$33,1$, resultando em um índice de custo-benefício de $0,08$). Esses resultados evidenciam a importância de analisar não apenas o tempo de execução da aplicação, mas também o tempo de escolha da MV e o custo das execuções. Isso ocorre porque, em alguns casos, o modelo de execução pode se tornar inviável para o usuário quando considera o tempo total, incluindo a escolha e a execução, bem como o custo envolvido.

Na Figura 5.23 expõe-se uma análise comparativa de diferentes aplicações (*PageRank*, *Sort*, *WordCount*) submetidas pelo usuário. O modelo escolhido é aquele que oferece o melhor custo benefício, considerando o tempo de execução e o custo financeiro. Observa-se que cada aplicação pode se beneficiar de modelos diferentes. Quando o tempo de execução é semelhante entre os modelos de raciocínio, e o custo financeiro é inferior, o tempo de escolha se torna mais relevante, resultando em um melhor custo benefício e uma experiência aprimorada para o usuário.

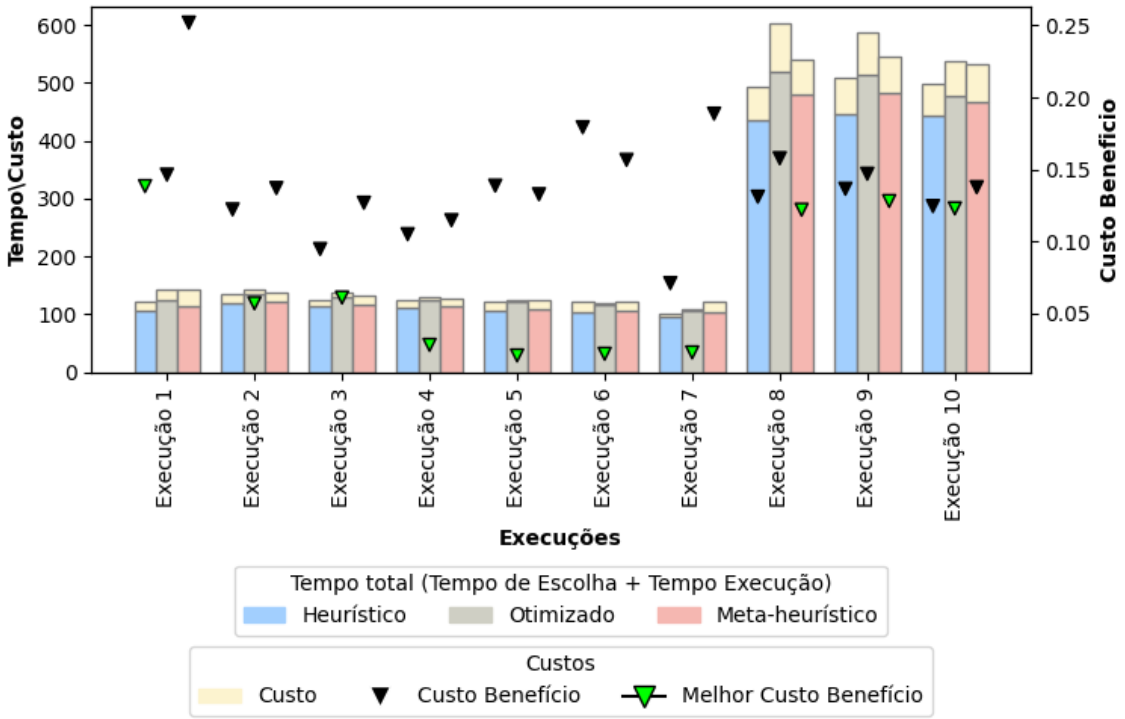

Figura 5.23: Comparação de custo acumulado de 10 execuções utilizando diferentes aplicações com MAS-Cloud+ na AWS EC2.

A utilização do modelo automático permite analisar e comparar fatores relacionados ao provisionamento, considerando carga de trabalho, demanda de recursos, restrições de desempenho e metas de qualidade de serviço. Uma vez que heurísticas oferecem soluções rápidas, porém podem não encontrar a solução ótima em cenários complexos. Já as Meta-heurísticas têm potencial para encontrar soluções de melhor qualidade, mas exigem mais tempo de execução. A otimização garante resultados precisos, porém e computacionalmente intensivo, podendo levar um tempo maior para encontrar a solução. Um modelo automático que selecione entre essas abordagens é essencial para adaptar-se às demandas do ambiente em nuvem, resultando em melhor alocação de recursos, redução de custos e satisfação dos usuários finais. No entanto, é importante considerar o tempo de seleção dos modelos, uma vez que modelos meta-heurísticos e otimizados podem demandar mais tempo à medida que a base de dados aumenta. A escolha adequada do modelo considera a complexidade e o tamanho do conjunto de dados é fundamental para a eficiência e escalabilidade do sistema. A análise do tempo de escolha e execução de cada modelo é crucial, pois o tempo total pode se tornar inviável em certos casos.

5.4 Discussão dos Resultados

Os experimentos realizados avaliaram a eficiência do MAS-Cloud+ em comparação com outras abordagens, usando estudos de Bioinformática e referências como Islam et al. [2022]. Foram analisados três modelos de gerenciamento: otimização, heurístico e meta-heurístico, cada um com desempenhos distintos:

- o modelo de otimização obteve o melhor desempenho em termos de tempo de execução, abaixo de 33 minutos para todas as cargas de trabalho testadas, demonstrando a capacidade do modelo em encontrar soluções eficientes para a alocação de recursos;
- o heurístico, se destacou pela economia de custos, alcançando economias de até $66,39\%$ em comparação com o modelo de otimização. Essa abordagem mostra-se vantajosa para usuários que priorizam a redução de custo;
- o meta-heurístico apresentou resultados próximos aos do heurístico em relação ao custo, porém com um desempenho um pouco melhor em termos de tempo de execução. Essa abordagem combina características de eficiência no tempo e no custo, tornando-se uma opção intermediária;
- a escolha automatizada dos modelos de raciocínio com base no custo-benefício mostrou-se crucial para a eficiência geral do sistema, otimizando a alocação de recursos conforme as especificidades de cada tarefa.

Ressalta-se que os resultados podem variar dependendo do contexto específico e das necessidades do problema em questão. A escolha do modelo de raciocínio mais adequado deve considerar não apenas os resultados numéricos, mas também outros fatores, como interpretabilidade, simplicidade e requisitos específicos do domínio.

No estudo da Bioinformática, os resultados revelaram que o MAS-Cloud+ realizou escolhas eficientes de configurações de MVs. Dependendo das prioridades do usuário, seja tempo de execução ou custo, é possível selecionar o modelo de raciocínio mais apropriado. O modelo heurístico se destaca pela economia de custos, a otimização apresenta os melhores tempos de execução e o meta-heurística combina um desempenho razoável em ambos os aspectos.

Comparativamente aos algoritmos de Islam et al. [2022], o MAS-Cloud+ obteve melhores tempos de execução e reduções de custo significativas em ambientes como AWS EC2 e *Google Cloud*. Para aplicações como *WordCount* e *PageRank*, houve reduções de tempo de execução e de custos. No entanto, vale ressaltar que o MAS-Cloud+ apresentou desempenho inferior para o *Sort*, uma vez que foi priorizado a redução de custo em detrimento do desempenho.

Considerando todos esses fatores, que a abordagem MAS-Cloud+ apresenta resultados promissores no gerenciamento otimizado de recursos em ambientes de nuvem. A integração de agentes de raciocínio dedutivo, otimização combinatória, monitoramento de nuvem e provisionamento de recursos por meio de uma abordagem multiagente demonstrou a redução de custos e no aprimoramento do tempo de execução, resultando em vantagens significativas em termos de tempo de execução e custos. A implementação do MAS-Cloud+ otimiza a alocação de recursos em ambientes de nuvem, promovendo tempos de execução mais eficientes e redução de custos, o que é promissor para melhorar a satisfação do usuário final. Além disso, essas descobertas podem servir de base para estudos futuros que explorem ainda mais os benefícios do MAS-Cloud+ em cenários de nuvem.

Capítulo 6

Conclusões

Esta tese apresentou o desenvolvimento e a validação da arquitetura multiagente denominada MAS-Cloud+, com modelos de raciocínio distintos, capaz de gerenciar de maneira otimizada os recursos em ambientes de nuvem. A arquitetura propõe um gerenciamento autônomo e eficiente dos recursos de nuvem, independente de aplicação, para múltiplos provedores, considerando os requisitos do usuário, com serviços de monitoramento, predição, e provisionamento de MVs.

O modelo de predição com RLM apresentou um desempenho consistente utilizando métricas de desempenho, como coeficiente de determinação (R^2), MSE e MAE. A regressão linear obteve uma boa taxa de acerto, pequenos erros médios e um bom desempenho geral. Os experimentos realizados em domínios de conhecimento distintas indicam que MAS-Cloud+ é uma proposta promissora para gerenciamento de recursos eficiente na nuvem computacional.

A utilização do modelo automático permite analisar e comparar fatores relacionados ao provisionamento, considerando carga de trabalho, demanda de recursos, restrições de desempenho de qualidade de serviço. Heurísticas oferecem soluções rápidas, porém podem não encontrar a solução ótima em cenários complexos. Meta-heurísticas têm potencial para encontrar soluções de melhor qualidade, mas exigem maior esforço computacional. Métodos de otimização garantem resultados precisos, porém são computacionalmente intensivos. Um modelo automático que selecione entre essas abordagens é essencial para adaptar-se às demandas do ambiente em nuvem, resultando em melhor alocação de recursos, redução de custos e satisfação dos usuários finais. No entanto, é importante considerar o tempo de seleção dos modelos, uma vez que modelos meta-heurísticas e otimizados podem demandar mais tempo à medida que a base de dados aumenta. A escolha adequada do modelo considerando a complexidade e o tamanho do conjunto de dados é fundamental para a eficiência e escalabilidade do sistema. A análise do tempo de escolha e execução dos modelos é crucial, pois o tempo total pode se tornar inviável em certos casos.

Limitações

A arquitetura proposta neste trabalho apresenta algumas limitações, que representam áreas de interesse para melhorias e pesquisas futuras.

- a arquitetura multiagente desenvolvida não inclui um mecanismo de tolerância a falhas, essencial para garantir a continuidade dos serviços de nuvem em caso de interrupções;
- faz-se necessário um volume de dados históricos para que o agente gerente de MV (VMMgr) realize previsões de recursos apropriadas, dada a abordagem ser independente da aplicação;
- a necessidade de testar a arquitetura com uma variedade maior de aplicações se torna relevante, especialmente em ambientes de nuvem com cargas de trabalho altamente variáveis;
- a arquitetura emprega um modelo linear de otimização e predição baseada em RLM, que embora úteis, podem não ser adequados para todas as aplicações, não capturando a complexidade de cenários dinâmicos.

Trabalhos futuros

Sugere-se, para trabalhos futuros, investigação sobre como os agentes do MAS-Cloud+ podem escolher, autonomamente, o melhor modelo de raciocínio para aplicações genéricas, considerando as preferências específicas dos usuários. Sugere-se também, a aplicação de modelos de predição além da regressão linear e redes neurais. Além disso, podem-se avaliar diferentes módulos de provisão, monitoramento, adaptação de recursos e validação de SLA utilizando outros tipos de agentes e modelos de raciocínio com aprendizado profundo para atender às recentes abordagens de autoescalabilidade proativa com RL, análise de séries temporais e esquemas bio-inspirados. Como o MAS-Cloud+ é independente do provedor, deseja-se desenvolver uma versão *multi-cloud*.

A implementação adicional do desenvolvimento de uma API de acesso para os principais provedores de nuvem para usar a camada de gestão do MAS-Cloud+ com os serviços de predição, provisão e monitoramento (agentes VMMgr & Mnt) seria interessante. Experimentos utilizando equações que priorizem separadamente o custo e o desempenho das cargas de trabalho poderiam atender a prioridades específicas dos usuários, ampliando a gama de opções para provisionamento de MVs. Experimentos complementares incluiriam maior cobertura na oferta de instâncias na nuvem com diferentes cargas de trabalho incluindo Computação de Alto Desempenho - *High Performance Computing* (HPC), usando

a Unidade de Processamento Gráfico - *Graphics Processing Unit* (GPU), misturando instâncias de CPU e GPU, aplicações de comércio eletrônico, recursos de rede como Acesso Direto à Memória Remota - *Remote Direct Memory Access* (RDMA) e *Elastic Fabric Adapter* (EFA), funções sem servidor e contêineres para implantação de aplicações, melhorando o desempenho para desenvolvedores e clientes.

Considerando esses resultados e perspectivas futuras, a arquitetura MAS-Cloud+ apresenta um grande potencial para melhorar o gerenciamento de recursos em ambientes de nuvem, oferecendo eficiência, redução de custos e otimização do tempo de execução. Com o contínuo avanço da computação em nuvem e a necessidade crescente de otimização de recursos, o MAS-Cloud+ se posiciona como uma solução promissora para enfrentar os desafios presentes nesse ambiente em constante evolução.

Referências

Mahbuba Afrin, Jiong Jin, Akhlaqur Rahman, Ashfaqur Rahman, Jiafu Wan, and Ekram Hossain. Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2021. 2

Mahmoud Al-Ayyoub, Yaser Jararweh, Mustafa Daraghmeh, and Qutaibah Althebyan. Multi-agent based dynamic resource provisioning and monitoring for cloud computing systems infrastructure. *Cluster Computing*, 18(2):919–932, 2015. 2

Mahmoud Al-Ayyoub, Mustafa Daraghmeh, Yaser Jararweh, and Qutaibah Althebyan. Towards improving resource management in cloud systems using a multi-agent framework. *International Journal of Cloud Computing*, 5(1-2):112–133, 2016. 2

May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013. 12

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, Technical Report No. UCB/ EECS-2009-28, University of California, Berkley, 2009. URL `https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`. 9

Wosley Arruda, Daniel S. Souza, Célia Ghedini Ralha, Maria Emília M. T. Walter, Tainá Raiol, Marcelo M. Brigido, and Peter F. Stadler. Knowledge-based reasoning to annotate noncoding RNA using multi-agent system. *J. Bioinform. Comput. Biol.*, 13(6): 1550021:1–1550021:19, 2015. doi: 10.1142/S0219720015500213. 3, 24

Ali Asghari, Mohammad Karim Sohrabi, and Farzin Yaghmaee. A cloud resource management framework for multiple online scientific workflows using cooperative reinforcement learning agents. *Computer Networks*, 179:107340, 2020. 36, 38

Costin Bădică, Lars Braubach, and Adrian Paschke. Rule-based distributed and agent systems. In Nick Bassiliades, Guido Governatori, and Adrian Paschke, editors, *Rule-Based Reasoning, Programming, and Applications*, pages 3–28, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-22546-8. 3, 24

Fabio Luigi Bellifemine, Giovanni Caire, and Dominic Greenwood. *Developing multi-agent systems with JADE*, volume 7. John Wiley & Sons, 2007. 23, 58

Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, volume 99, pages 478–485, 1999. 22

Michael E. Bratman. *Intention, Plans, and Practical Reason*. Center for the Study of Language and Information, 1987. Available at `https://press.uchicago.edu/ucp/books/book/distributed/I/bo362 9095.html`. 3, 24

Paul Browne and Peter Johnson. *JBoss Drools business rules*, volume 1847196063. Packt Publishing Birmingham, 2009. 51, 58

Rajkumar Buyya, Chee Shin Yeo, and Srikumar Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *2008 10th IEEE international conference on high performance computing and communications*, pages 5–13. Ieee, 2008. 17, 42

Rajkumar Buyya, James Broberg, and Andrzej M Goscinski. *Cloud computing: Principles and paradigms*. John Wiley & Sons, 2010. 9

Rajkumar Buyya, Satish Narayana Srirama, Giuliano Casale, Rodrigo Calheiros, Yogesh Simmhan, Blesson Varghese, Erol Gelenbe, Bahman Javadi, Luis Miguel Vaquero, Marco AS Netto, et al. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM computing surveys (CSUR)*, 51(5):1–38, 2018. 9

Rafael C. Cardoso and Angelo Ferrando. A review of agent-based programming for multi-agent systems. *Computers*, 10(2), 2021. ISSN 2073-431X. doi: 10.3390/computers10020016. URL `https://www.mdpi.com/2073-431X/10/2/16`. 23

José-Antonio Cervantes, Luis-Felipe Rodríguez, Sonia López, Félix Ramos, and Francisco Robles. Autonomous agents and ethical decision-making. *Cognitive Computation*, 8(2): 278–296, 2016. 24

Célia da Costa Pereira, Beishui Liao, Alessandra Malerba, Antonino Rotolo, Andrea G. B. Tettamanzi, Leendert W. N. van der Torre, and Serena Villata. Handling norms in multi-agent systems by means of formal argumentation. *IfCoLog Journal of Logics and Their Applications (FLAP)*, 4(9):3039–3073, 2017. 3

Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *arXiv preprint arXiv:1704.01665*, 2017. 29

Jonathan Dale and Margaret Lyell. Foundation for intelligent physical agents. *Online verfügbar unter http://www. fipa. org/, zuletzt aktualisiert am*, 4:2014, 2014. 22

George Bernard Dantzig and Mukund N Thapa. *Linear programming: Theory and extensions*, volume 2. Springer, 2003. 29

Fernando De la Prieta, Javier Bajo, Sara Rodríguez, and Juan M Corchado. Mas-based self-adaptive architecture for controlling and monitoring cloud platforms. *Journal of Ambient Intelligence and Humanized Computing*, 8(2):213–221, 2017. 2

Fernando De la Prieta, Sara Rodríguez-González, Pablo Chamoso, Yves Demazeau, and Juan Manuel Corchado. An intelligent approach to allocating resources within an agent-based cloud computing platform. *Applied Sciences*, 10(12):4361, 2020. 2, 36, 39

Farah Bellaaj Elloumi, Hayet Brabra, Mohamed Sellami, Walid Gaaloul, and Sami Bhiri. A transactional approach for reliable elastic cloud resources. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 154–161. IEEE, 2019. 2

Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995. 27

Jacques Ferber. *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading, 1999. 2

Paola Festa and Mauricio GC Resende. Grasp: An annotated bibliography. In *Essays and surveys in metaheuristics*, pages 325–367. Springer, 2002. 27

Roy T. Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1 internet official protocol standards. Technical report, The Internet Society, 1999. 45

Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, School of Information and Computer Science, University of California, 6210 Donald Bren Hall, Irvine, CA 92697-3425, USA, 2000. 45

Ian Foster, Yong Zhao, Ioan Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10, 2008. doi: 10.1109/GCE.2008.4738445. 1

W. Funika, P. Koperek, and J. Kitowski. Automated cloud resources provisioning with the use of the proximal policy optimization. *The Journal of Supercomputing*, 79:6674–6704, 2023. doi: https://doi.org/10.1007/s11227-022-04924-3. 36

M. Georgeff, B. Pell, M. Pollack, Milind Tambe, and M. Wooldrige. The belief-desire-intention model of agency. In *Agents, Theories, Architectures and Languages (ATAL)*, 1999. 24

Mohammad Hossein Ghahramani, MengChu Zhou, and Chi Tin Hon. Toward cloud computing qos architecture: Analysis of cloud systems and cloud services. *IEEE/CAA Journal of Automatica Sinica*, 4(1):6–18, 2017. 11, 13

Sukhpal Singh Gill and Rajkumar Buyya. A taxonomy and future directions for sustainable cloud computing: 360 degree view. *ACM Computing Surveys (CSUR)*, 51(5):1–33, 2018. viii, 10, 11, 12

Tian Guo and Prashant Shenoy. Providing geo-elasticity in geographically distributed clouds. *ACM Transactions on Internet Technology (TOIT)*, 18(3):1–27, 2018. 36, 39

Saransh Gupta, Mohsen Imani, Behnam Khaleghi, Venkatesh Kumar, and Tajana Rosing. Rapid: A reram processing in-memory architecture for dna sequence alignment. In *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6. IEEE, 2019. 46

Frank Harrell. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015. 47

Mitchell Hashimoto. *Vagrant: up and running: create and manage virtualized development environments*. "O'Reilly Media, Inc.", 2013. 58

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. 17

Jomi F Hübner, Olivier Boissier, and Rafael H Bordini. A normative programming language for multi-agent organisations. *Annals of Mathematics and Artificial Intelligence*, 62:27–53, 2011. 25

Muhammed Tawfiqul Islam, Huaming Wu, Shanika Karunasekera, and Rajkumar Buyya. SLA-Based Scheduling of Spark Jobs in Hybrid Cloud Computing Environments. *IEEE Transactions on Computers*, 71(5):1117–1132, 2022. doi: 10.1109/TC.2021.3075625. ix, x, xi, 3, 36, 38, 60, 61, 62, 63, 80, 81, 82, 83, 84, 85, 86, 88, 89

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013. 19

Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998. 22

Joe. All about cpu and memory utilization in performance testing, Nov 2022. URL `https://testguild.com/performance-test-resource-utilization/`. 46

Mouna Jouini and Latifa Ben Arfa Rabai. A security framework for secure cloud computing environments. In *Cloud security: Concepts, methodologies, tools, and applications*, pages 249–263. IGI Global, 2019. 2

Amrita Jyoti and Manish Shrimali. Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing. *Cluster Computing*, 23(1): 377–395, 2020. 36, 39

KarlErickson. How to deploy large cpu and memory applications in azure spring apps in the enterprise plan, 2022. URL `https://learn.microsoft.com/en-us/azure/spring-apps/how-to-enterprise-large-cpu-memory-applications?tabs=azure-portal`. 46

Gurleen Kaur, Anju Bala, and Inderveer Chana. An intelligent regressive ensemble approach for predicting resource usage in cloud computing. *Journal of Parallel and Distributed Computing*, 123:1–12, 2019. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2018.08.008. 1

Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004. 31

M. Kolp, P. Giorgini, and J. Mylopoulos. Multi-agent architectures as organizational structures. *Auton Agent Multi-Agent Syst*, 13:3—-25, 2006. doi: 10.1007/s10458-006-5717-6. URL `https://doi.org/10.1007/s10458-006-5717-6`. 19

Serge Kruk. *Practical Python AI Projects Mathematical Models of Optimization Problems with Google OR-Tools*. Springer, 2018. 58

A. M. Senthil Kumar, K. Padmanaban, A. K. Velmurugan, X. S. Asha Shiny, and Dinesh Kumar Anguraj. A novel resource management framework in a cloud computing environment using Hybrid Cat Swarm BAT (HCSBAT) algorithm. *Distributed and Parallel Databases*, 41(1-2):53–63, 2023a. 36, 37

Akkrabani Bharani Pradeep Kumar and P Venkata Nageswara Rao. Energy efficient, resource-aware, prediction based vm provisioning approach for cloud environment. *International Journal of Ambient Computing and Intelligence (IJACI)*, 11(3):22–41, 2020. 47

Anil Kumar, Manoj Kumar Dash, Shrawan Kumar Trivedi, and Tapan Kumar Panda. *Handbook of Research on Intelligent Techniques and Modeling Applications in Marketing Analytics*. IGI Global, 2016. 18

K Dinesh Kumar and E Umamaheswari. Prediction methods for effective resource provisioning in cloud computing: A survey. *Multiagent and Grid Systems*, 14(3):283–305, 2018. viii, 3, 14, 15, 16

Rishi Kumar, Mohd Fadzil Hassan, Muhamad Hariz Muhamad Adnan, Saurabh Shukla, Sohail Safdar, Muhammad Aasim Qureshi, and Abdel-Haleem Abdel-Aty. A user-priorities-based strategy for three-phase intelligent recommendation and negotiating agents for cloud services. *IEEE Access*, 11:26932–26944, 2023b. doi: 10.1109/ACCESS.2023.3254552. 36, 37

Michael H Kutner, Christopher J Nachtsheim, John Neter, and William Wasserman. *Applied linear regression models*, volume 4. McGraw-Hill/Irwin New York, 2004. 18

Fatma Lahmar and Haithem Mezni. Multicloud service composition: a survey of current approaches and issues. *Journal of Software: Evolution and Process*, 30(10):e1947, 2018. 3

Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016. 25

Shujun Li, Miecyslaw Kokar, Shujun Li, and Mieczyslaw M Kokar. Agent communication language. *Flexible Adaptation in Cognitive Radios*, pages 37–44, 2013. 22

Xingjia Li, Li Pan, and Shijun Liu. A survey of resource provisioning problem in cloud brokers. *Journal of Network and Computer Applications*, page 103384, 2022. 17

Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003. 27

Maria Malik, Dean M Tullsen, and Houman Homayoun. Co-locating and concurrent fine-tuning mapreduce applications on microservers for energy efficiency. In *2017 IEEE International Symposium on Workload Characterization (IISWC)*, pages 22–31. IEEE, 2017. 58

Gopa Mandal, Santanu Dam, Kousik Dasgupta, and Paramartha Dutta. *A Linear Regression-Based Resource Utilization Prediction Policy for Live Migration in Cloud Computing*, pages 109–128. Springer Singapore, Singapore, 2020. ISBN 978-981-15-1041-0. doi: 10.1007/978-981-15-1041-0_7. 47

Toufik Marir, Farid Mokhati, Abdelouahed Gherbi, Ahmed Bali, et al. Norjade: An open source jade-based framework for programming normative multi-agent systems. *International Journal of Open Source Software and Processes (IJOSSP)*, 10(2):1–20, 2019. 2, 3

Mohammad Masdari, Sasan Gharehpasha, Mostafa Ghobaei-Arani, and Vafa Ghasemi. Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. *Cluster Computing*, 23:2533–2563, 2020. doi: 10.1007/s10586-019-03026-9. 47

Peter Mell and Tim Grance. The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009. 10, 13, 14

Peter Mell and Timothy Grance. The nist definition of cloud computing, 2011. 9

Aldo H.D. Mendes, Michel J.F. Rosa, Marcelo A. Marotta, Aleteia Araujo, Alba C.M.A. Melo, and Célia Ghedini Ralha. MAS-Cloud+: A novel multi-agent architecture with reasoning models for resource management in multiple providers. *Future Generation Computer Systems*, 154:16–34, 05 2024. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2023.12.022. URL `https://www.sciencedirect.com/science/article/pii/S0167739X23004776`. 6

Abigail M Methley, Stephen Campbell, Carolyn Chew-Graham, Rosalind McNally, and Sudeh Cheraghi-Sohi. Pico, picos and spider: a comparison study of specificity and sensitivity in three search tools for qualitative systematic reviews. *BMC health services research*, 14(1):1–10, 2014. 33

Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021. 17

Sehrish Nadeem, Noor ul Amin, Sardar Khaliq uz Zaman, Muhammad Amir Khan, Zulfiqar Ahmad, Jawaid Iqbal, Ajab Khan, Abeer D. Algarni, and Hela Elmannai. Runtime management of service level agreements through proactive resource provisioning for a cloud environment. *Electronics*, 12(2), 2023. ISSN 2079-9292. doi: 10.3390/electronics12020296. 3, 45

Jose Pergentino Araujo Neto, Donald M. Pianto, and Célia Ghedini Ralha. Towards increasing reliability of amazon EC2 spot instances with a fault-tolerant multi-agent architecture. *Multiagent Grid Syst.*, 15(3):259–287, 2019. doi: 10.3233/MGS-190312. URL `https://doi.org/10.3233/MGS-190312`. 2

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50(9):3826–3839, 2020. doi: 10.1109/TCYB.2020.2977374. 24

Patryk Osypanka and Piotr Nawrocki. Resource usage cost optimization in cloud computing using machine learning. *IEEE Transactions on Cloud Computing*, 10(3):2079–2089, 2022. doi: 10.1109/TCC.2020.3015769. 36, 38

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998. 29

Chenhao Qu, Rodrigo N. Calheiros, and Rajkumar Buyya. Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Comput. Surv.*, 51(4):1–33, 2018. doi: 10.1145/3148149. 47, 48

Ima Rahmanian, Yves Lucet, and Solomon Tesfamariam. Optimal design of reinforced concrete beams: A review. *Computers and Concrete*, 13(4):457–482, 2014. 30

Célia Ghedini Ralha, Aldo H. D. Mendes, Luiz A. Laranjeira, Aletéia P. F. Araújo, and Alba C. M. A. Melo. Multiagent system for dynamic resource provisioning in cloud computing platforms. *Future Gener. Comput. Syst.*, 94:80–96, 2019. doi: 10.1016/j.future.2018.09.050. 2, 48, 51, 68

Sebastian Raschka and Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019. 19

Mikko Rautiainen and Tobias Marschall. Graphaligner: rapid and versatile sequence-to-graph alignment. *Genome biology*, 21(1):253, 2020. 46

Muhammad Raheel Raza and Asaf Varol. Qos parameters for viable sla in cloud. In *2020 8th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–5. IEEE, 2020. 10

KR Remesh Babu and P Samuel. Review of the quality of service scheduling mechanisms in cloud. *Int J Eng Technol*, 7(3):1677–1695, 2018. 14, 16

Mauricio GC Resende and Celso C Ribeiro. *Optimization by GRASP*. Springer, 2016. 3

S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010. ISBN 9780136042594. viii, 20, 21, 22, 25

Stuart Russell. Rationality and intelligence: A brief update. In *Fundamental issues of artificial intelligence*, pages 7–28. Springer, 2016. 24

Adel Saadi, Ramdane Maamri, and Zaidi Sahnoun. Behavioral flexibility in belief-desire-intention (bdi) architectures. *Multiagent and grid systems*, 16(4):343–377, 2020. 3

Coşkun Şahin and Reda Alhajj. Crowd behavior modeling in emergency evacuation scenarios using belief-desire-intention model. In *Putting Social Media and Networking Data in Practice for Education, Planning, Prediction and Recommendation*, pages 1–14. Springer, 2020. 3, 24

Mufeed Ahmed Naji Saif, SK Niranjan, Belal Abdullah Hezam Murshed, Hasib Daowd Esmail Al-ariki, and Hudhaifa Mohammed Abdulwahab. Multi-agent qos-aware autonomic resource provisioning framework for elastic bpm in containerized multi-cloud environment. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–26, 2022. 2, 36, 37

Edans F. de O. Sandes, Guillermo Miranda, Xavier Martorell, Eduard Ayguade, George Teodoro, and Alba C. M. A. De Melo. MASA: A multiplatform architecture for sequence aligners with block pruning. *ACM Trans. Parallel Comput.*, 2(4), feb 2016. ISSN 2329-4949. doi: 10.1145/2858656. 61, 65

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347v2 [cs.LG]*, 2017. 36

Haiying Shen and Liuhua Chen. Resource demand misalignment: An important factor to consider for reducing resource over-provisioning in cloud datacenters. *IEEE/ACM Transactions on Networking*, 26(3):1207–1221, 2018. 1

Sukhpal Singh and Inderveer Chana. Cloud resource provisioning: survey, status and future research directions. *Knowledge and Information Systems*, 49(3):1005–1069, 2016. 3, 14, 16

Sukhpal Singh, Inderveer Chana, and Rajkumar Buyya. STAR: SLA-aware autonomic management of cloud resources. *IEEE Transactions on Cloud Computing*, 2017. 49

Sukhpal Singh, Inderveer Chana, and Rajkumar Buyya. STAR: SLA-aware autonomic management of cloud resources. *IEEE Transactions on Cloud Computing*, 8(4):1040–1053, 2020. doi: 10.1109/TCC.2017.2648788. 46, 62

R Leena Sri and N Balaji. An empirical model of adaptive cloud resource provisioning with speculation. *Soft Computing*, 23(21):10983–10999, 2019. 47

Katia P. Sycara. Multiagent systems. *AI magazine*, 19(2):79–92, 1998. 21

Hassan Jamil Syed, Abdullah Gani, Raja Wasim Ahmad, Muhammad Khurram Khan, and Abdelmuttlib Ibrahim Abdalla Ahmed. Cloud monitoring: A review, taxonomy, and open research issues. *Journal of Network and Computer Applications*, 98:11–26, 2017. ISSN 1084-8045. doi: 10.1016/j.jnca.2017.08.021. 14, 16

El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009. 26

Ming Tao, Chao Qu, Wenhong Wei, Bin Zhou, and Shuqiang Huang. Hybrid cloud architecture for cross-platform interoperability in smart homes. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 608–617. Springer, 2018. 11, 13

Luan Teylo, Ubiratam de Paula, Yuri Frota, Daniel de Oliveira, and Lúcia MA Drummond. A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds. *Future Generation Computer Systems*, 76:1–17, 2017. 47

Garret N Vanderplaats. *Numerical optimization techniques for engineering design: with applications*, volume 1. McGraw-Hill New York, 1984. 28

Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008. 13

Lei Wang, Jianfeng Zhan, Chunjie Luo, Yuqing Zhu, Qiang Yang, Yongqiang He, Wanling Gao, Zhen Jia, Yingjie Shi, Shujie Zhang, et al. Bigdatabench: A big data benchmark suite from internet services. In *2014 IEEE 20th international symposium on high performance computer architecture (HPCA)*, pages 488–499. IEEE, 2014. 80

Wei-Jen Wang, Yue-Shan Chang, Win-Tsung Lo, and Yi-Kang Lee. Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments. *The Journal of Supercomputing*, 66:783–811, 2013. 81

Pallavi Wankhede, Minaiy Talati, and Rutuja Chinchamalatpure. Comparative study of cloud platforms-microsoft azure, google cloud platform and amazon ec2. *J. Res. Eng. Appl. Sci*, 5(02):60–64, 2020. 13

Gerhard Weiss. *Multiagent Systems*. The MIT Press, 2^{nd} edition, 2013. 2, 21

Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009. 2, 20, 21

Xiaolong Xu, Hanzhong Rong, Marcello Trovati, Mark Liptrott, and Nik Bessis. Cs-pso: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems. *Soft Computing*, 22(3):783–795, 2018. 29

Xiaolong Xu, Ruichao Mo, Fei Dai, Wenmin Lin, Shaohua Wan, and Wanchun Dou. Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud. *IEEE Transactions on Industrial Informatics*, 2019. 2

Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010. 27

Xin-She Yang. Chapter 10 - bat algorithms. In Xin-She Yang, editor, *Nature-Inspired Optimization Algorithms*, pages 141–154. Elsevier, Oxford, 2014. ISBN 978-0-12-416743-8. doi: https://doi.org/10.1016/B978-0-12-416743-8.00010-5. URL `https://www.sciencedirect.com/science/article/pii/B9780124167438000105`. 37

Lamia Youseff, Maria Butrico, and Dilma Da Silva. Toward a unified ontology of cloud computing. In *2008 Grid Computing Environments Workshop*, pages 1–10. IEEE, 2008. 17, 42

Xin Yuan, Michael John Liebelt, Peng Shi, and Braden J. Phillips. Creating rule-based agents for artificial general intelligence using association rules mining. *Int. J. Mach. Learn. & Cyber.*, 12:223—-230, 2021. URL `https://doi.org/10.1007/s13042-020-01166-8`. 3, 24

Saman Zahoor, Sakeena Javaid, Nadeem Javaid, Mahmood Ashraf, Farruh Ishmanov, and Muhammad Khalil Afzal. Cloud–fog–based smart grid model for efficient resource management. *Sustainability*, 10(6):2079, 2018. 47

Weiting Zhang, Dong Yang, Haixia Peng, Wen Wu, Wei Quan, Hongke Zhang, and Xuemin Shen. Deep reinforcement learning based resource management for dnn inference in industrial iot. *IEEE Transactions on Vehicular Technology*, 70(8):7605–7618, 2021. 24

Eduard Zharikov, Sergii Telenyk, and Petro Bidyuk. Adaptive workload forecasting in cloud data centers. *Journal of Grid Computing*, 18:149—168, 2020. doi: 10.1007/s10723-019-09501-2. 47

Abdallah Ziraba and Mbata David. *Forecasting Cloud Storage Consumption Using Regression Model*. Grin Verlag, 2018. ISBN 3668660409. 47

Apêndice A

Publicações Resultantes da Revisão Sistemática

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura.

Ano	Autor	Título	Veículo
2023	Kumar et al.	A novel resource management framework in a cloud computing environment using hybrid cat swarm BAT (HCSBAT) algorithm	Distributed and Parallel Databases
2023	Kumar et al.	A User-Priorities-Based Strategy for Three-Phase Intelligent Recommendation and Negotiating Agents for Cloud Services	IEEE Access
2023	Funika et al.	Automated cloud resources provisioning with the use of the proximal policy optimization	The Journal of Supercomputing
2022	Saif et al.	Multi-agent QoS-aware autonomic resource provisioning framework for elastic BPM in containerized multi-cloud environment	Journal of Ambient Intelligence and Humanized Computing
2022	Osypanka et al.	Resource Usage Cost Optimization in Cloud Computing Using Machine Learning	IEEE Transactions on Cloud Computing
2021	Islam et al.	Sla-based scheduling of spark jobs in hybrid cloud computing environments	IEEE Transactions on Computers
2020	Asghari et al.	A cloud resource management framework for multiple online scientific workflows using cooperative reinforcement learning agents	Computer Networks

Continua na próxima página

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura. (Continuação)

Ano	Autor	Título	Veículo
2020	Zhong et al.	A Cost-Efficient Container Orchestration Strategy in Kubernetes-Based Cloud Computing Infrastructures with Heterogeneous Resources	ACM Trans. Internet Technol.
2020	Khan et al.	An energy, performance efficient resource consolidation scheme for heterogeneous cloud datacenters	Journal of Network and Computer Applications
2020	De la Prieta et al.	An Intelligent Approach to Allocating Resources within an Agent-Based Cloud Computing Platform	Applied Sciences
2020	Jyoti et al.	Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing	Cluster Computing
2020	Zhou et al.	Genetic Ant Colony Algorithm Improves Resource Scheduling in Cloud Computing	Proceedings of the 3rd Int. Conf. on Information Science and Systems
2020	Taghavi et al.	Two-stage game theoretical framework for IaaS market share dynamics	Future Generation Computer Systems
2019	López-Huguet et al.	A self-managed Mesos cluster for data analytics with QoS guarantees	Future Generation Computer Systems
2019	Canali et al.	AGATE: Adaptive Gray Area-Based TEchnique to Cluster Virtual Machines with Similar Behavior	IEEE Transactions on Cloud Computing
2019	Li et al.	Deep Reinforcement Scheduling for Mobile Crowdsensing in Fog Computing	ACM Trans. Internet Technol.
2019	Mazouzi et al.	DM2-ECOP: An Efficient Computation Offloading Policy for Multi-User Multi-Cloudlet Mobile Edge Computing Environment	ACM Trans. Internet Technol.
2019	Qiu et al.	Dynamic Demand Prediction and Allocation in Cloud Service Brokerage	IEEE Transactions on Cloud Computing
2019	Duc et al.	Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey	ACM Comput. Surv.

Continua na próxima página

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura. (Continuação)

Ano	Autor	Título	Veículo
2019	Zakarya et al.	Managing energy, performance and cost in large scale heterogeneous datacenters using migrations	Future Generation Computer Systems
2019	Adhikari et al.	Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud	Journal of Network and Computer Applications
2019	Stephen et al.	Monitoring IaaS using various cloud monitors	Cluster Computing
2019	Rastegar et al.	Online Virtual Machine Assignment Using Multi-Armed Bandit in Cloud Computing	Int. Conf. on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)
2019	Singh et al.	Optimizing Energy Consumption for Cloud Computing: A Cluster and Migration Based Approach (CMBA)	Proceedings of the 3rd Int. Conf. on Computer Science and Artificial Intelligence
2018	Iranpour et al.	A distributed load balancing and admission control algorithm based on Fuzzy type-2 and Game theory for large-scale SaaS cloud architectures	Future Generation Computer Systems
2018	Righi et al.	A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications	Future Generation Computer Systems
2018	Labba et al.	A predictive approach for the efficient distribution of agent-based systems on a hybrid-cloud	Future Generation Computer Systems
2018	Ouyang et al.	Adaptive Speculation for Efficient Internetware Application Execution in Clouds	ACM Trans. Internet Technol.
2018	Callau-Zori et al.	An experiment-driven energy consumption model for virtual machine management systems	Sustainable Computing: Informatics and Systems
2018	Radhika et al.	An RNN-LSTM Based Flavor Recommender Framework in Hybrid Cloud	17th IEEE Int. Conf. on Machine Learning and Applications (ICMLA).

Continua na próxima página

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura. (Continuação)

Ano	Autor	Título	Veículo
2018	Alsadie et al.	Dynamic Resource Allocation for an Energy Efficient VM Architecture for Cloud Computing	Proceedings of the Australasian Computer Science Week Multiconference
2018	Caglar et al.	iTune: Engineering the Performance of Xen Hypervisor via Autonomous and Dynamic Scheduler Reconfiguration	IEEE Transactions on Services Computing
2018	Zhou et al.	Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms	Future Generation Computer Systems
2018	Chen et al.	Multiple Virtual Machines Live Migration Scheduling Method Study on VMware vMotion	3rd Int. Conf. on Computer and Communication Systems (ICCCS)
2018	Akingbesote et al.	Performance Modelling of a Cloud based Health Monitoring System Using Dynamic Control System	Int. Conf. on Advances in Big Data, Computing and Data Communication Systems (icABCD)
2018	Guo et al.	Providing Geo-Elasticity in Geographically Distributed Clouds	ACM Trans. Internet Technol.
2018	Thein et al.	Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers	Journal of King Saud University - Computer and Information Sciences
2018	Zarrin et al.	Resource discovery for distributed computing systems: A comprehensive survey	Journal of Parallel and Distributed Computing
2018	Chang et al.	Reward-based Markov chain analysis adaptive global resource management for inter-cloud computing	Future Generation Computer Systems
2018	Sáez et al.	Utility-Based Decision Making for Migrating Cloud-Based Applications	ACM Trans. Internet Technol.
2017	Colajanni et al.	A Cloud Computing Network and an Optimization Algorithm for IaaS Providers	Proceedings of the 2nd Int. Conf. on Internet of Things, Data and Cloud Computing

Continua na próxima página

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura. (Continuação)

Ano	Autor	Título	Veículo
2017	Ciavotta et al.	A mixed integer linear programming optimization approach for multi-cloud capacity allocation	Journal of Systems and Software
2017	Batista et al.	A QoS-driven approach for cloud computing addressing attributes of performance and security	Future Generation Computer Systems
2017	Samo et al.	Advocating isolation of resources among multi-tenants by containerization in IaaS cloud model	Int. Conf. on Innovations in Electrical Engineering and Computational Technologies (ICIEECT)
2017	Liu et al.	An adaptive prediction approach based on workload pattern discrimination in the cloud	Journal of Network and Computer Applications
2017	Abd et al.	An effective approach for managing power consumption in cloud computing infrastructure	Journal of Computational Science
2017	Park et al.	Dynamic Resource Management for Efficient Utilization of Multitasking GPUs	Proceedings of the 22nd Int. Conf. on Architectural Support for Programming Languages and Operating Systems
2017	Amarasinghe et al.	Fault-tolerant IaaS management for networked cloud infrastructure with SDN	IEEE Int. Conf. on Communications (ICC)
2017	Costache et al.	Market-based autonomous resource and application management in private clouds	Journal of Parallel and Distributed Computing
2017	Knodel et al.	Migration of Long-Running Tasks between Reconfigurable Resources Using Virtualization	SIGARCH Comput. Archit. News
2017	Ouyang et al.	ML-NA: A Machine Learning Based Node Performance Analyzer Utilizing Straggler Statistics	IEEE 23rd Int. Conf. on Parallel and Distributed Systems (ICPADS)
2017	Arianyan et al.	Novel fuzzy multi objective DVFS-aware consolidation heuristics for energy and SLA efficient resource management in cloud data centers	Journal of Network and Computer Applications

Continua na próxima página

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura. (Continuação)

Ano	Autor	Título	Veículo
2017	Xiao et al.	QoE and power efficiency tradeoff for fog computing networks with fog node cooperation	IEEE INFOCOM 2017-IEEE Conference on Computer Communications
2017	Sharma et al.	Reliable and Energy Efficient Resource Provisioning and Allocation in Cloud Computing	Proceedings of the 10th Int. Conf. on Utility and Cloud Computing
2017	Hong et al.	Research and Implementation of Load Balancing Technology for Cloud Computing	Int. Conf. on Computer Network, Electronic and Automation (ICCNEA)
2017	Lin et al.	Resource Allocation of Cloud Application Through Machine Learning: A Case Study	Int. Conf. on Green Informatics (ICGI)
2017	Singh et al.	STAR: SLA-aware Autonomic Management of Cloud Resources	IEEE Transactions on Cloud Computing
2016	Govindaraju et al.	A QoS and Energy Aware Load Balancing and Resource Allocation Framework for IaaS Cloud Providers	Proceedings of the 9th Int. Conf. on Utility and Cloud Computing.
2016	Yan et al.	An Adaptive Decision Making Approach Based on Reinforcement Learning for Self-Managed Cloud Applications	IEEE Int. Conf. on Web Services (ICWS)
2016	Zhou et al.	CASH: Supporting IaaS Customers with a Sub-core Configurable Architecture	ACM SIGARCH Computer Architecture News
2016	Talib et al.	Cloud Computing Based E-Commerce as a Service Model: Impacts and Recommendations	Proceedings of the Int. Conf. on Internet of Things and Cloud Computing
2016	Yu et al.	CloudSeer: Workflow Monitoring of Cloud Infrastructures via Interleaved Logs	ACM SIGARCH Computer Architecture News
2016	Albonico et al.	Controlling the Elasticity of Web Applications on Cloud Computing	Proceedings of the 31st Annual ACM Symposium on Applied Computing

Continua na próxima página

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura. (Continuação)

Ano	Autor	Título	Veículo
2016	Vakilinia et al.	Dynamic Resource Allocation of Smart Home Workloads in the Cloud	12th Int. Conf. on Network and Service Management (CNSM)
2016	Wang et al.	FD4C: Automatic Fault Diagnosis Framework for Web Applications in Cloud Computing	IEEE Transactions on Systems, Man, and Cybernetics: Systems
2016	Bui et al.	Fuzzy Fault Detection in IaaS Cloud Computing	Proceedings of the 10th Int. Conf. on Ubiquitous Information Management and Communication
2016	Qi et al.	Resource orchestration for multi-task application in home-to-home cloud	IEEE Transactions on Consumer Electronics
2016	Singh et al.	Resource provisioning and scheduling in clouds: QoS perspective	The Journal of Supercomputing
2016	Serrano et al.	SLA guarantees for cloud services	Future Generation Computer Systems
2015	Mann et al.	Allocation of Virtual Machines in Cloud Data Centers—A Survey of Problem Models and Optimization Algorithms	ACM Comput. Surv.
2015	Assunção et al.	Big Data computing and clouds: Trends and future directions	Journal of Parallel and Distributed Computing
2015	Zhan et al.	Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches	ACM Comput. Surv.
2015	Zhang et al.	CloudMonatt: An Architecture for Security Health Monitoring and Attestation of Virtual Machines in Cloud Computing	Proceedings of the 42nd Annual Int. Symposium on Computer Architecture
2015	Zabolotnyi et al.	JCloudScale: Closing the Gap Between IaaS and PaaS	ACM Trans. Internet Technol.
2015	Coutinho et al.	Optimizing virtual machine allocation for parallel scientific workflows in federated clouds	Future Generation Computer Systems
2015	Singh et al.	Q-aware: Quality of service based cloud resource provisioning	Computers & Electrical Engineering

Continua na próxima página

Tabela A.1: Trabalhos aceitos na revisão sistemática da literatura. (Continuação)

Ano	Autor	Título	Veículo
2015	Nadaph et al.	Regional Parting and Equipoise Distribution of Job on Cloud Division for Public Cloud	Procedia Computer Science
2015	Parida et al.	Truthful Resource Allocation Detection Mechanism for Cloud Computing	Proceedings of the 3rd Int. Symposium on Women in Computing and Informatics