UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCIANO DE SOUZA FRAGA

Orquestração de recursos para a oferta de serviços, em infraestruturas híbridas de computação de borda e nuvem, com foco em aplicações de realidade mista

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[X] Dissertação [ ] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Luciano de Souza Fraga

3. Título do trabalho

Orquestração de recursos para a oferta de serviços, em infraestruturas híbridas de computação de borda e nuvem, com foco em aplicações de realidade mista

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Kleber Vieira Cardoso**, **Professor do Magistério Superior**, em 27/06/2025, às 07:38, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Luciano De Souza Fraga**, **Discente**, em 27/06/2025, às 11:03, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5465038** e o código CRC **732C94D6**.

Referência: Processo nº 23070.021241/2025-21 SEI nº 5465038

LUCIANO DE SOUZA FRAGA

Orquestração de recursos para a oferta de serviços, em infraestruturas híbridas de computação de borda e nuvem, com foco em aplicações de realidade mista

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Informática, da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação.

Linha de Pesquisa: Sistemas de Computação.

Orientador: Prof. Kleber Vieira Cardoso

Co-Orientador: Prof. Leizer de Lima Pinto

Goiânia
2025

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Fraga, Luciano de Souza
Orquestração de recursos para a oferta de serviços, em infraestruturas híbridas de computação de borda e nuvem, com foco em aplicações de realidade mista [manuscrito] / Luciano de Souza Fraga. - 2025.
lxxxvi, 86 f.: il.

Orientador: Prof. Kleber Vieira Cardoso; co-orientador Leizer de Lima Pinto.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2025.
Bibliografia. Apêndice.
Inclui siglas, abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. computação de borda. 2. computação em nuvem. 3. realidade mista. 4. orquestração de tarefas. 5. alocação de recursos. I. Vieira Cardoso, Kleber, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº **11** da sessão de Defesa de Dissertação de **Luciano de Souza Fraga**, que confere o título de Mestre em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos trinta dias do mês de maio de dois mil e vinte e cinco, a partir das dez horas, via sistema de webconferência, realizou-se a sessão pública de Defesa de Dissertação intitulada **"Orquestração de recursos para a oferta de serviços, em infraestruturas híbridas de computação de borda e nuvem, com foco em aplicações de realidade mista"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor Kleber Vieira Cardoso (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professor Doutor Leizer de Lima Pinto (INF/UFG), coorientador; Professor Doutor José Ferreira de Rezende (COPPE/UFRJ), membro titular externo; e Professor Doutor Elivelton Ferreira Bueno (Minerva Foods), membro titular externo. A realização da banca ocorreu por meio de videoconferência. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Kleber Vieira Cardoso, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos trinta dias do mês de maio de dois mil e vinte e cinco.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **José Ferreira de Rezende**, **Usuário Externo**, em 30/05/2025, às 11:42, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luciano De Souza Fraga**, **Discente**, em 30/05/2025, às 11:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Kleber Vieira Cardoso**, **Professor do Magistério Superior**, em 30/05/2025, às 11:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Elivelton Ferreira Bueno**, **Usuário Externo**, em 30/05/2025, às 11:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Leizer De Lima Pinto**, **Professor do Magistério Superior**, em 30/05/2025, às 11:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5377297** e o código CRC **7A6DA81D**.

Referência: Processo nº 23070.021241/2025-21 SEI nº 5377297

Luciano de Souza Fraga

Graduou-se em Sistemas de Informação pela Universidade Federal de Goiás (UFG). Durante a graduação, atuou como monitor na disciplina de Pesquisa Operacional e participou de projetos voltados à alocação de recursos em redes 5G, com ênfase em computação de borda multiacesso. Atualmente, desenvolve atividades em projetos nas áreas de Otimização em Redes, Pesquisa Operacional, Computação de Borda, 5G e Redes Móveis.

Dedico este trabalho à minha família e amigos.

Agradecimentos

A conclusão desta dissertação, fruto de anos de trabalho contínuo e colaborativo, só foi possível graças à valiosa contribuição de pessoas e entidades. Por isso, sou grato:

A Deus, pela minha saúde e pela saúde daqueles que me cercam.

À minha família, em especial aos meus pais, Regina e Odair, por me ensinarem desde cedo o valor da educação e por me apoiarem incondicionalmente, independentemente das minhas escolhas.

Aos meus orientadores, professores Kleber Cardoso e Leizer Pinto, por abrirem as portas da pesquisa científica para mim em 2021, muito antes de ingressar no programa de pós-graduação, em um momento em que o país se fechava em uma das maiores crises globais em gerações. Sou grato pelo apoio, incentivo, conselhos, conhecimentos compartilhados e, sobretudo, pela paciência ao longo dessa jornada.

Aos colegas do laboratório de redes de computadores e sistemas distribuídos (LABORA), pela parceria e troca constante de experiências.

À Universidade Federal de Goiás (UFG), pelo suporte e pelos recursos oferecidos aos seus alunos, que tanto contribuíram para minha trajetória acadêmica.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela concessão da bolsa de estudos que me permitiu dedicar-me integralmente à pesquisa e que ajuda no fortalecimento da ciência no Brasil.

Resumo

Fraga, Luciano de Souza. **Orquestração de recursos para a oferta de serviços, em infraestruturas híbridas de computação de borda e nuvem, com foco em aplicações de realidade mista**. Goiânia, 2025. 86p. Dissertação de Mestrado. Programa de Pós-Graduação em Ciência da Computação, Instituto de Informática, Universidade Federal de Goiás.

A alocação eficiente de recursos em ambientes híbridos de computação em borda e nuvem torna-se cada vez mais importante diante da crescente adoção de aplicações de realidade mista e da popularização de dispositivos com restrições de energia, processamento e memória. Uma estratégia de alocação bem projetada não apenas garante o cumprimento dos requisitos de qualidade de serviço dessas aplicações, como também promove o uso otimizado da infraestrutura computacional e de rede, resultando em menor custo operacional. Neste trabalho, propomos um modelo baseado em Programação Linear Inteira (Integer Linear Programming (ILP)) com o objetivo de maximizar o atendimento da demanda gerada pelos dispositivos dos usuários, ao mesmo tempo em que minimizamos o custo associado à utilização das máquinas virtuais responsáveis pelo processamento. Avaliamos a complexidade do modelo e propomos simplificações estruturais, além de desenvolver uma heurística voltada à redução do tempo de geração das soluções. Por fim, introduzimos uma abordagem proativa baseada em um modelo preditivo que antecipa padrões de utilização dos recursos, contribuindo para decisões mais precisas em relação a abordagens reativas. Os resultados experimentais demonstram ganhos expressivos no volume de demanda atendida em comparação com outras abordagens na literatura, além de evidenciar os benefícios da adoção de estratégias proativas na alocação de recursos.

Palavras–chave

Computação de borda, computação em nuvem, realidade mista, orquestração de tarefas, alocação de recursos, otimização, abordagem proativa.

Abstract

Fraga, Luciano de Souza. **Resource Orchestration for Service Provisioning in Hybrid Cloud-Edge Computing Infrastructures with a Focus on Mixed Reality Applications**. Goiânia, 2025. 86p. MSc. Dissertation. Programa de Pós-Graduação em Ciência da Computação, Instituto de Informática, Universidade Federal de Goiás.

Efficient resource allocation in hybrid edge-cloud computing environments is becoming increasingly important due to the growing adoption of mixed reality applications and the widespread use of devices with limited energy, processing, and memory resources. A well-designed allocation strategy not only ensures compliance with the quality of service (QoS) requirements of these applications but also promotes optimized use of computational and network infrastructure, resulting in lower operational costs. In this work, we propose a model based on Integer Linear Programming (ILP) aimed at maximizing the fulfillment of demand generated by user devices, while minimizing the cost associated with the use of virtual machines responsible for processing. We evaluate the complexity of the model and propose structural simplifications, in addition to developing a heuristic designed to reduce solution generation time. Finally, we introduce a proactive approach based on a predictive model that anticipates resource usage patterns, contributing to more accurate decisions compared to reactive strategies. Experimental results demonstrate significant improvements in the volume of demand served when compared to other approaches in the literature, as well as highlight the benefits of adopting proactive strategies for resource allocation.

Keywords

Edge computing, cloud computing, mixed reality, task orchestration, resource allocation, optimization, proactive approach.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Siglas e Acrônimos

5G quinta geração de rede móvel celular
AP *Access Point*
APSC *Assignment Problems with Side Constraints*
APSCAQ *Assignment Problems with Side Constraints and Agent Qualification*
AR *Augmented Reality*
ARIMA *Autoregressive Integrated Moving Average*
BRB *Belief Rule-Based*
CC *Cloud Computing*
DAG *Directed Acyclic Graph*
DDQN *Double Deep Q-Network*
DQN *Deep Q-Learning*
DRL *Deep Reinforcement Learning*
EC *Edge Computing*
FL *Fuzzy Logic*
FC *Fog Computing*
GA *Genetic Algorithm*
GAP *Generalized Assignment Problem*
GNN *Graph Neural Network*
GRASP *Greedy Randomized Adaptive Search Procedure*
HMD *Head-Mounted Display*
ILP *Integer Linear Programming*
IoT *Internet of Things*
LSTM *Long Short-Term Memory*
MAN *Metropolitan Area Network*
MDP *Markov Decision Process*
MRL *Meta Reinforcement Learning*
MI *Million of Instructions*
MIPS *Million Instructions Per Second*
ML *Machine Learning*
MR *Mixed Reality*

MR-Leo *Mixed Reality Linköping edge offloading*
QoS *Quality of Service*
RL *Reinforcement Learning*
RMSE *Root Mean Square Error*
RSU *Road Side Units*
SA *Simulated Annealing*
SVM *Support Vector Machine*
VEC *Vehicular Edge Computing*
VM *Virtual Machine*
VR *Virtual Reality*
WAN *Wide-Area Network*
WLAN *Wireless Local Area Network*

CAPÍTULO 1

Introdução

O futuro das redes de acesso sem fio se caracteriza pela projeção de avanços expressivos que viabilizam a oferta de serviços com exigências cada vez mais rigorosas em termos de consumo de recursos e qualidade de serviço (*Quality of Service* (QoS)). Dentre esses serviços, destacam-se as aplicações de realidade mista (*Mixed Reality* (MR)), que exemplificam de forma clara os desafios impostos por cenários com requisitos críticos, especialmente devido à necessidade de baixa latência e elevada demanda computacional [26, 41, 39].

Além dos desafios intrínsecos a esse tipo de aplicação, a crescente densidade de dispositivos conectados às redes sem fio, impulsionada pela expansão da internet das coisas (*Internet of Things* (IoT)), aumenta a complexidade na manutenção dos parâmetros de qualidade exigidos [30]. Cada dispositivo móvel apresenta características particulares e restritivas em termos de capacidade de processamento, memória, energia e armazenamento, o que inviabiliza, na maioria dos casos, a execução local de cargas de trabalho mais intensivas [11, 15, 37].

A partir dessas limitações, a computação de borda (*Edge Computing* (EC)) surge como um elemento essencial para viabilizar o processamento de aplicações com carga de processamento intensiva, especialmente quando combinada com os avanços das tecnologias sem fio, como Wi-Fi 6E, 7 e 8 [19].

A adoção de computação de borda permite que cargas de trabalho sejam executadas em servidores localizados na borda da rede, oferecendo uma alternativa à computação em nuvem tradicional (*Cloud Computing* (CC)), na qual os serviços são ofertados em servidores concentrados em centros de processamento de dados distantes do local de geração da demanda. Ao aproximar os recursos computacionais dos usuários finais, essa abordagem reduz significativamente a latência, além de aliviar a sobrecarga tanto da rede quanto a sobrecarga no uso de recursos de dispositivos com capacidades limitadas [19, 11].

Apesar dos benefícios da implantação de serviços de maneira descentralizada na borda, a combinação de altas cargas de trabalho e a natureza dinâmica dessas demandas pode levar à escassez de recursos. Nesse contexto, torna-se crucial priorizar a alocação de serviços com requisitos mais rigorosos de QoS na borda. Além disso, quando os recursos

na borda são insuficientes, é essencial identificar tarefas que apresentam requisitos mais flexíveis e que permitam que sejam ofertadas de maneira centralizada na nuvem.

A escassez de recursos e a congestão da rede não são eventos incomuns e podem ocorrer de maneira sazonal durante a operação regular dos serviços. Esses problemas são ainda mais acentuados em eventos sociais que reúnem grandes grupos de usuários em um curto intervalo de tempo, como eventos esportivos ou festivais de música. Esses cenários destacam a necessidade de estratégias avançadas de alocação de recursos que garantam o cumprimento dos requisitos de QoS, otimizando ao mesmo tempo o uso dos recursos limitados disponíveis nos servidores distribuídos na borda.

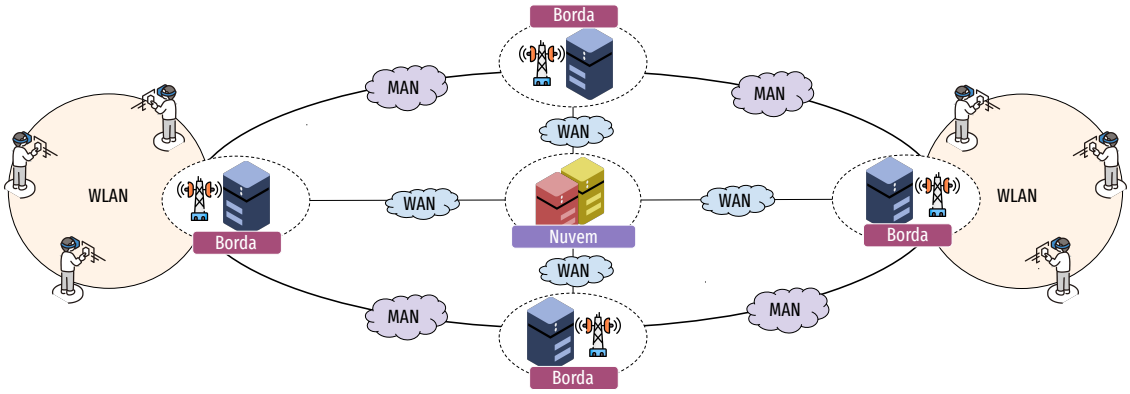


Figura 1.1: *Infraestrutura híbrida de computação de borda e nuvem em três camadas. A primeira camada contém os dispositivos de usuários, a segunda camada contém os servidores na borda e na terceira camada está disposto o servidor centralizado na nuvem.*

Neste contexto, o objetivo principal deste trabalho é investigar estratégias para alocação (i.e., orquestração) de recursos em uma infraestrutura híbrida de computação de borda e nuvem, com o intuito de atender adequadamente aos requisitos de latência de aplicações imersivas, em especial, realidade mista.

A Figura 1.1 ilustra a infraestrutura de computação híbrida considerada neste trabalho. Nessa infraestrutura, podemos visualizar um conjunto de usuários conectados a pontos de acesso (*Access Point* (AP)) na borda da rede, onde servidores são alocados próximos da geração da demanda. De maneira centralizada, um servidor na nuvem oferece recursos menos escassos de processamento intensivo para um conjunto maior de usuários, embora a sua disposição seja geograficamente mais distante da geração da demanda, acarretando maior latência no envio dos dados a serem processados. Maiores detalhes sobre os recursos de computação de borda e na nuvem e dos recursos de comunicação que conectam os elementos da arquitetura serão apresentados no Capítulo 2.

Consideramos que a formalização do problema mediante um arcabouço consolidado é fundamental, o que nos motiva a adotar programação matemática como ferramenta. Também investigamos a utilização de estratégias baseadas em heurísticas com o intuito

de diminuir o tempo de geração de soluções. Buscamos ainda aplicar técnicas de predição como modelos autoregressivos integrados de médias móveis (*Autoregressive Integrated Moving Average* (ARIMA)) para inferir elementos importantes durante o processo de alocação de recursos, no sentido de fornecer uma maior gama de informações aos modelos de orquestração propostos.

Nesse contexto, elaboramos as seguintes perguntas de pesquisa que buscamos responder ao longo do trabalho:

1. **Qual é o impacto ao se utilizar uma solução localmente exata que considera a alocação de recursos de maneira concorrente para um conjunto de usuários?**

A hipótese levantada com a primeira pergunta de pesquisa tem como base a noção de que, embora a formalização de problemas de alocação de recursos através de programação matemática seja uma prática comum em algumas áreas, observamos que há trabalhos importantes na literatura que não utilizam essa abordagem.

Da mesma forma, entre todos os trabalhos relacionados considerados, nenhum realiza a alocação de recursos de maneira concorrente, considerando um conjunto de usuários (mais detalhes no Capítulo 2). Portanto, consideramos que é importante avaliar o efetivo impacto da formulação e resolução de maneira localmente exata do problema de interesse.

Conforme será mostrado nos próximos capítulos, a utilização de uma solução localmente exata, considerando a resolução do problema de alocação para vários usuários de maneira conjunta, apresenta benefícios em comparação a outras soluções que alocam recursos de maneira individualizada. Os objetivos específicos relacionados a essa pergunta de pesquisa são:

- Investigar o impacto de uma solução formalizada e resolvida através de programação matemática em comparação a outras soluções não exatas.
- Avaliar o impacto da solução proposta levando em consideração um conjunto de tarefas, e verificar as diferenças em comparação a soluções que realizam a orquestração de maneira instantânea para cada tarefa gerada.

2. **Quais os benefícios ao alocar recursos no contexto de aplicações de realidade mista?**

No contexto de aplicações de realidade mista, a hipótese levantada com a segunda pergunta se relaciona à importância de se adotar uma estratégia que realize o balanceamento e a alocação eficiente de recursos entre a borda e a nuvem, priorizando os serviços com requisitos mais restritivos por meio de sua alocação na borda. Além disso, serviços imersivos têm como característica a exigência de baixa latência para

operarem de maneira adequada, e grande parte dos trabalhos considerados não utiliza tal parâmetro como elemento de decisão ao alocar recursos. Com isso, podemos supor que, ao utilizarmos uma aplicação de realidade mista como caso de uso, iremos visualizar de maneira clara a priorização de alocação dos recursos em um cenário com escassez.

Embora nossa solução tenha sido modelada para permitir a alocação de recursos a serviços de forma genérica, a caracterização de uma aplicação de realidade mista serviu como caso de uso para a avaliação dos pontos citados.

Deste modo, com relação à segunda pergunta, foram delineados os seguintes objetivos específicos:

- Investigar os benefícios de se utilizar uma solução que leva em consideração requisitos de latência requerida por uma tarefa, especialmente durante a priorização da alocação na borda ou na nuvem.
- Avaliar o impacto dessa solução em tarefas com características diversas e em especial para uma aplicação de realidade mista.

3. **Para o problema de interesse, há a necessidade de utilizar abordagens não exatas, como heurísticas?**

Com relação à terceira pergunta de pesquisa, observamos que ao usarmos uma abordagem exata (mesmo que localmente) baseada em ILP, as soluções geradas podem apresentar um custo computacional elevado. Diante disso, este trabalho busca investigar duas formas de melhoria do tempo de geração de solução do modelo: 1) a reformulação do modelo proposto; e 2) o desenvolvimento de uma heurística como alternativa eficiente para o problema de orquestração.

Neste sentido, foram mapeados os seguintes objetivos em relação à terceira pergunta de pesquisa:

- Verificar se é possível diminuir a complexidade do modelo original proposto a partir de reformulações e avaliar a qualidade das soluções geradas pelas simplificações em comparação ao modelo original.
- Propor e avaliar o uso de heurística em comparação as soluções geradas a partir das reformulações propostas e do modelo original.

4. **Quais os benefícios de se utilizar uma abordagem proativa em comparação a abordagens reativas na orquestração de recursos para tarefas de realidade mista? Da mesma forma, abordagens baseadas em predição de series temporais como ARIMA são promissoras no contexto de abordagens proativas?**

A hipótese considerada é a de que abordagens proativas auxiliam na tomada de decisão ao servir como fonte extra de informação utilizada pelo modelo. Ao considerarmos a quarta pergunta de pesquisa, observamos um predomínio na literatura de soluções

reativas para a orquestração de recursos, ou seja, abordagens que alteram sua estratégia de alocação de recursos após a degradação no desempenho.

Neste trabalho, ao investigarmos a qualidade de soluções baseadas em abordagens proativas, utilizamos ARIMA como modelo preditivo que realiza a inferência de elementos utilizados no modelo de orquestração de tarefas.

Com relação a essa pergunta de pesquisa, foram definidos os seguintes objetivos específicos:

- Investigar os benefícios de se utilizar uma abordagem proativa, onde parte dos dados disponíveis ao orquestrador são fornecidos por um modelo preditor.
- Avaliar o uso de ARIMA como ferramenta auxiliar no desenvolvimento da abordagem proativa.

A partir do capítulo introdutório, o restante do trabalho está organizado conforme apresentado a seguir:

No Capítulo 2, é apresentada a fundamentação teórica onde são descritos os principais elementos e conceitos considerados nesta pesquisa. Além disso, apresentamos o mapeamento dos trabalhos relacionados e suas principais características em contraste com este trabalho.

No Capítulo 3, o problema de alocação de recursos é formalizado e avaliado. A solução proposta, denominada *LOTOS*, é formulada em três estágios e apresenta dois objetivos. O primeiro estágio realiza o cálculo da demanda consumida por um conjunto de tarefas alocadas em uma mesma máquina virtual (*Virtual Machine* (VM)). O segundo estágio corresponde ao primeiro objetivo, onde buscamos maximizar o atendimento da demanda. O terceiro estágio corresponde ao segundo objetivo, onde buscamos minimizar o custo de utilização da infraestrutura de computação.

O Capítulo 4 apresenta duas remodelagens do problema de alocação, visando a uma maior eficiência. Na primeira remodelagem, unificamos os dois objetivos (estágios 2 e 3) em uma única função objetivo. Na segunda remodelagem, além das modificações já realizadas na primeira reformulação, também eliminamos o estágio 1, de forma que os cálculos das demandas passam a ser realizados de maneira mais eficiente, resultando em menor custo computacional.

No Capítulo 5 apresentamos e avaliamos uma heurística para o problema de alocação de recursos. A solução proposta, denominada *LATOS*, se caracteriza por ser uma heurística construtiva gananciosa que apresenta menor tempo de geração de soluções, mesmo se comparado à reformulação mais eficiente apresentada no Capítulo 4 apesar de conseguir gerar boas aproximações em relação ao modelo original em três estágios.

O Capítulo 6 apresenta uma abordagem proativa de orquestração de carga de trabalho baseada em ARIMA. A solução realiza a predição de demandas futuras de modo

que o orquestrador disponha de mais informações, prevenindo eventos futuros de escassez de recursos.

Por fim, no Capítulo 7, apresentamos as considerações finais do trabalho, o que inclui as contribuições produzidas, as publicações, relacionadas ou não, com os estudos realizados nesta pesquisa, além de delinearmos sugestões de trabalhos futuros.

CAPÍTULO 2

Fundamentação teórica e trabalhos relacionados

Neste capítulo, apresentamos os principais conceitos que fazem parte do escopo deste trabalho, assim como também serão discutidos e comparados os trabalhos relacionados e as diferenças em relação a este trabalho.

2.1 Fundamentação teórica

Nesta seção, apresentamos alguns conceitos fundamentais utilizados ao longo do desenvolvimento deste trabalho. Inicialmente, abordamos conceitos relacionados à orquestração de tarefas, bem como aspectos da infraestrutura híbrida de computação em nuvem e na borda, composta por recursos de computação e comunicação. Em seguida, descrevemos a caracterização de uma aplicação de realidade mista utilizada como principal caso de uso desta pesquisa. Também apresentamos aspectos da fase de experimentação, em especial a estrutura geral da ferramenta de simulação empregada nos experimentos. Por fim, apresentamos os princípios da abordagem proativa de alocação de recursos, com destaque para o uso do modelo ARIMA.

2.1.1 Orquestração de tarefas

A orquestração de tarefas envolve a alocação eficiente de recursos computacionais para o processamento da demanda gerada pelos dispositivos de usuário, com o objetivo de atender a requisitos característicos de cada aplicação. O processo de orquestração compreende o envio de um módulo de uma aplicação para um servidor remoto, onde é executado, e a posterior devolução dos resultados processados ao dispositivo que originou a demanda [22, 15].

Uma tarefa pode ser compreendida como uma unidade digital de um programa, caracterizada pela carga de dados gerada por um usuário e destinada ao armazenamento e processamento de maneira local ou remota [22]. Cada usuário pode originar um conjunto

de tarefas com diferentes atributos, embora possam apresentar padrões em suas características, caso sejam geradas pela mesma aplicação [15]. Entre as principais características de uma tarefa, destacam-se: o volume de dados a ser transmitido pelos meios de comunicação, os dados que retornarão ao dispositivo do usuário após o processamento e a carga computacional a ser processada.

Os recursos a serem alocados podem ser classificados em três categorias principais: computação, comunicação e armazenamento.

Os recursos de computação e armazenamento são disponibilizados de maneira local pelos dispositivos de usuários e de maneira remota por servidores na borda e na nuvem. Neste trabalho, consideramos apenas a alocação de recursos de maneira remota, pois, conforme apresentado no Capítulo 1 e na Seção 2.1.2, nossa hipótese é avaliar a orquestração de tarefas para serviços com carga de processamento que excedem os recursos limitados de dispositivos móveis, como aplicações de realidade mista.

Conforme ilustrado na Figura 1.1, consideramos uma infraestrutura organizada em três camadas. A primeira camada é representada pelos equipamentos dos usuários, sendo o ponto de origem das tarefas. A segunda camada abriga os servidores de borda e os pontos de acesso, por meio dos quais os dispositivos móveis se conectam para consumir os serviços desejados. Nessa mesma camada está localizado o orquestrador, responsável por definir o destino de processamento de cada tarefa, seja na borda ou na nuvem. Por fim, a terceira camada corresponde à nuvem centralizada, composta por servidores com alta capacidade computacional.

Os recursos de comunicação são representados pelos enlaces que habilitam o envio da carga gerada pelas tarefas ao servidor remoto e de volta ao dispositivo de usuário. Cada usuário está conectado a apenas um ponto de acesso por vez, geralmente o mais próximo de sua localização atual.

A comunicação entre o dispositivo móvel do usuário e o ponto de acesso ocorre por meio de um enlace de comunicação sem fio local (*Wireless Local Area Network* (WLAN)). A partir do ponto de acesso, todas as tarefas geradas pelos dispositivos dos usuários têm seu destino definido pelo orquestrador. Quando uma tarefa é alocada em um servidor de borda distinto daquele associado ao ponto de acesso, a comunicação entre os servidores de borda é realizada por meio de um enlace de acesso metropolitano (*Metropolitan Area Network* (MAN)). Caso a tarefa seja direcionada a um servidor na nuvem, a comunicação se dá através de um enlace de longa distância (*Wide-Area Network* (WAN)).

A Figura 2.1 ilustra o processo de orquestração de tarefas realizado pelo orquestrador, evidenciando como as características de cada tarefa influenciam diretamente na tomada de decisão. Tarefas associadas a serviços com alta sensibilidade à latência, por exemplo, tendem a ser alocadas em servidores próximos ao ponto de acesso. Já tarefas

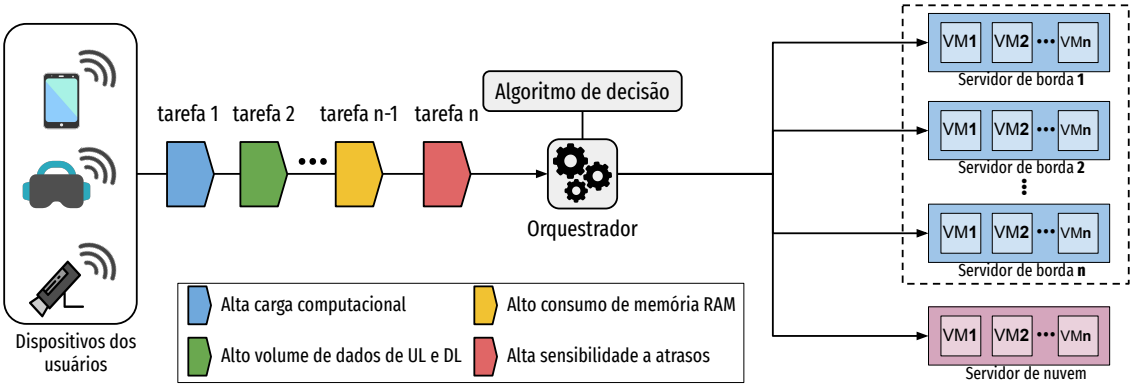


Figura 2.1: *Fluxo de orquestração de carga de trabalho. Tarefas com requisitos restritivos são alocadas na borda enquanto tarefas com requisitos flexíveis são alocadas na nuvem.*

que demandam maior capacidade de processamento, mas toleram maiores atrasos, podem ser encaminhadas a servidores de borda mais distantes, porém com maior quantidade de recursos disponíveis. Da mesma forma, tarefas com elevado consumo de memória e requisitos mais flexíveis em relação à latência podem ser direcionadas à nuvem, que dispõe de mais recursos.

2.1.2 Aplicações de realidade mista

A realidade mista é um tipo de aplicação imersiva onde elementos de um ambiente real apresentam aspectos interativos gerados a partir de um ambiente sintético. Embora faça parte da mesma categoria de outros conceitos como realidade virtual (*Virtual Reality* (VR)) e realidade aumentada (*Augmented Reality* (AR)) os três conceitos se diferenciam no nível de imersão proporcionado. Podemos entender realidade virtual como a total imersão em um ambiente sintético, sem interação do usuário com o ambiente real. Da mesma forma, a realidade aumentada se caracteriza pela sobreposição de elementos digitais em elementos do mundo real [25, 35].

Assim a realidade mista representa a combinação de elementos reais e virtuais onde existe um equilíbrio entre o nível de interação tanto com o ambiente real quanto com o ambiente virtual. A Figura 2.2 apresenta o espectro de virtualidade contínua proposto em [25], que ajuda a visualizar a realidade mista dentro de um espectro de imersão. Podemos notar que a realidade mista se encontra no meio do espectro que representa o nível de imersão proporcionado por uma tecnologia imersiva.

Aplicações de realidade mista, em geral, apresentam três componentes distintos: 1) Um sensor que permite a captura de elementos do mundo real; 2) Algoritmos com alta carga computacional que realizam a integração dos ambientes real e sintético; 3) Uma plataforma de apresentação dos resultados gerados para o usuário [39].

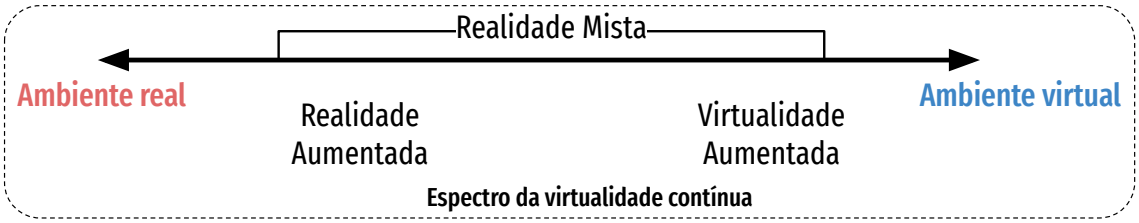


Figura 2.2: *Teoria da virtualidade continua. A realidade mista se encontra no centro do espectro pois apresenta uma integração equivalente entre elementos do mundo real e sintético.* [1]

Enquanto os componentes 1 e 3 podem ser integrados a óculos próprios para aplicações imersivas (*Head-Mounted Display* (HMD)) sem que haja perda em QoS, a integração do componente 2 enfrenta desafios devido à escassez de recursos em tais dispositivos que possam processar a carga computacional gerada.

Logo, a alocação desse tipo de serviço de maneira remota, isto é, em servidores na borda da rede ou centralizado na nuvem se torna uma alternativa ao problema de recursos limitados de dispositivos móveis. Levando em consideração as hipóteses levantadas nesta pesquisa, é possível notar como serviços de realidade mista são um bom caso de uso para problemas de alocação remota de serviços em uma infraestrutura híbrida. Portanto, para podermos testar estratégias de alocação em uma aplicação com tais requisitos, necessitamos de um perfil que represente de forma acurada um serviço de realidade mista.

Os autores em [39] apresentam uma caracterização de uma aplicação de realidade mista denominada MR-Leo [38]. Os autores consideram que o componente que contém algoritmos de integração dos ambientes virtual e real (componente 2), por conta de sua alta carga computacional, deve ser alocado em servidores de maneira remota. Enquanto isso, os outros dois componentes (sensores de captura do ambiente e exibição de resultados) continuam a ser executados nos dispositivos de usuário, por não exigirem quantidade elevada de recursos. O objetivo dessa caracterização é justamente traçar um perfil de consumo de recursos, de modo que seja possível simular a execução dessa aplicação em uma ferramenta específica.

São coletadas, através de experimentos, a demanda de processamento, a quantidade de núcleos requerida, o número de instruções por segundo, a quantidade de dados enviados e recebidos entre o servidor remoto e o dispositivo de usuário e a quantidade de memória demandada por essa aplicação. Assim, para que possamos modelar uma solução de alocação de recursos que aumente a qualidade de serviço de realidade mista, utilizaremos a caracterização do MR-Leo, para simular a alocação de tarefas em um cenário híbrido de computação de borda e nuvem.

[1] Figura adaptada de [25].

2.1.3 Simuladores

A avaliação de soluções para alocação de recursos na borda ou na nuvem apresenta desafios significativos, como a complexidade das operações envolvidas, os altos custos de implantação de infraestrutura de computação e rede, além do tempo excessivo exigido para a execução de experimentos em larga escala.

Por esse motivo, a experimentação da solução de orquestração de tarefas proposta neste trabalho em um ambiente real de grande porte torna-se inviável. Para viabilizar a realização de experimentos comparativos entre a nossa abordagem e outras propostas da literatura, adotamos uma ferramenta de simulação que torna o processo de avaliação mais acessível e controlado.

Uma ferramenta de simulação, especialmente no contexto da experimentação em computação de borda e em nuvem, permite modelar a criação de máquinas virtuais, servidores e tarefas, bem como simular a alocação de uma tarefa em determinada máquina virtual e a migração entre diferentes paradigmas de computação ou servidores.

A literatura apresenta uma ampla gama de ferramentas voltadas à simulação da orquestração de tarefas, cada uma com diferentes características. O *CloudSim* destaca-se por oferecer uma plataforma voltada à alocação de recursos em ambientes de computação em nuvem. Ao modelar uma máquina virtual associada a um servidor, são especificadas propriedades como a capacidade de processamento (medida em *Million Instructions Per Second* (MIPS)), memória, armazenamento e número de núcleos utilizados no processamento das cargas de trabalho (ou *Cloudlets*) [6]. A partir do *CloudSim*, diversas soluções foram propostas com o objetivo de estender suas funcionalidades. Uma dessas evoluções é o *CloudSim Plus*, que aprimora aspectos de usabilidade e manutenção em relação à versão original [31].

O *PureEdgeSim*, por sua vez, é uma extensão do *CloudSim Plus* voltada para a simulação de ambientes de computação de borda, nuvem e névoa (*Fog Computing* (FC)). Ele permite avaliar estratégias de alocação de recursos com base em métricas como latência, consumo de energia, utilização de recursos e taxa de eficácia no processamento de tarefas [24].

Da mesma forma, o *EdgeCloudSim* é uma ferramenta que estende as capacidades do *CloudSim*, adicionando módulos que possibilitam a modelagem de infraestruturas de computação de borda e nuvem, redes, padrões de mobilidade, orquestração e geração de carga [32]. Neste trabalho, utilizamos essa ferramenta devido à sua conveniência na implantação de estratégias de orquestração híbridas, que constituem o foco deste estudo.

A Figura 2.3 apresenta a relação entre os módulos disponíveis no *EdgeCloudSim*. A funcionalidade de cada módulo é descrita a seguir:

1. **Módulo de mobilidade:** Realiza a atualização da localização dos dispositivos de

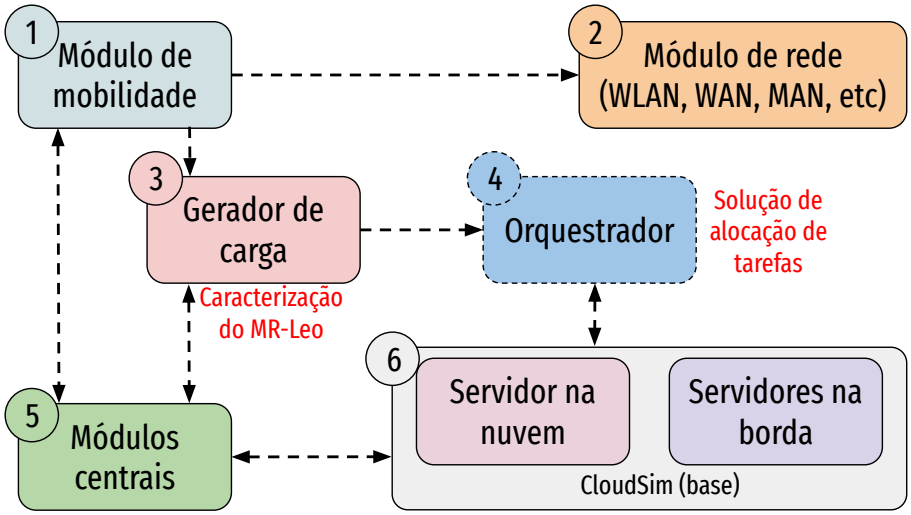


Figura 2.3: *Componentes do simulador EdgeCloudSim. Os módulos mais relevantes para este trabalho são o gerador de carga, onde integramos a caracterização do MR-Leo, e o módulo de orquestração onde inserimos a solução de orquestração proposta.* [2]

usuários de acordo com a estratégia considerada. Neste trabalho consideramos um modelo de mobilidade nômade (*nomadic*) onde um usuário permanece em uma localização por um período aleatório gerado com base no nível de atratividade de determinada localização. O nível de atratividade se refere a estimativas que definem a popularidade de determinado ponto em relação aos outros ou o tempo médio que um usuário permanece naquele local.

2. **Módulo de rede:** O módulo de rede define o modo de comunicação sem fio utilizado pelos dispositivos de usuários de modo a realizar o envio de dados a serem processados de maneira remota.
3. **Gerador de carga:** O gerador de carga possibilita a modelagem de carga de trabalho que serve como entrada pelo orquestrador. Neste módulo realizamos a integração do gerador de carga considerando a aplicação de realidade mista MR-Leo.
4. **Orquestrador:** Este é o principal módulo a ser utilizado neste trabalho pois a partir do orquestrador podemos inserir a solução de orquestração apresentada.
5. **Módulos centrais:** Módulo de inicialização da simulação e geração de resultados.
6. **Módulo base (*CloudSim*):** Neste módulo elementos do CloudSim são reutilizados pelo *EdgeCloudSim*.

[2] Figura adaptada de [32].

2.1.4 Abordagem proativa de alocação de recursos

A alocação de recursos em cenários de computação em nuvem e borda distribuída deve levar em consideração o comportamento dinâmico da demanda, que varia continuamente ao longo do tempo. O processo decisório pode ser realizado através de uma abordagem reativa ou proativa. Em uma abordagem reativa, as decisões são tomadas com base em regras predefinidas, considerando apenas o estado atual do sistema. Já em estratégias proativas, busca-se antecipar variações futuras na utilização da infraestrutura, possibilitando ações preparatórias que busquem melhorar o aproveitamento dos recursos computacionais e de rede [8, 36].

Tradicionalmente, modelos de orquestração e alocação de recursos adotaram técnicas reativas que, embora simples, tendem a perder eficiência à medida que o sistema cresce em escala e complexidade. Por outro lado, abordagens proativas têm ganhado destaque com o avanço de métodos de aprendizagem e técnicas estatísticas aplicadas à análise de séries temporais. Esses métodos permitem identificar padrões históricos de uso e prever tendências futuras, o que contribui para uma gestão mais eficiente da infraestrutura, especialmente em aplicações com requisitos rígidos de desempenho [14, 21].

Dentre os modelos utilizados para previsão, podemos citar o ARIMA, amplamente reconhecido pela sua eficácia em modelar séries temporais. Esse modelo é capaz de capturar padrões históricos de variação de consumo de recursos e projetar seu comportamento futuro, contribuindo para decisões mais informadas e com menor risco de subutilização ou sobrecarga [9, 17].

O ARIMA é um modelo estatístico amplamente utilizado na análise e previsão de séries temporais, onde apresenta três características principais [42]:

- **Auto regressivo**: utiliza a dependência linear entre valores passados da série;
- **Integrado**: aplica diferenciação nos dados para torná-los estacionários (ou seja, com média e variância constantes ao longo do tempo);
- **Média móvel**: modela o erro da previsão como uma combinação linear de erros passados.

A forma geral do modelo, antes da aplicação da diferenciação, pode ser representada pela seguinte equação:

$$X_t = \alpha + \sum_{i=1}^{p} \beta_i X_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} \tag{2-1}$$

Nessa equação, X_{t-i} representa o valor da série temporal no instante $t-i$, enquanto α é um termo constante. O coeficiente β_i corresponde ao componente autorregressivo, que captura a influência dos valores passados da própria série sobre o valor atual. Já

o coeficiente θ_j pertence ao componente de média móvel, representando o impacto dos erros passados (ε_{t-j}) na previsão do instante t.

Para que um modelo seja classificado como ARIMA, é necessário aplicar um processo de diferenciação à série temporal original. Esse procedimento tem como objetivo torná-la estacionária, eliminando tendências e variações sazonais.

O parâmetro d representa justamente o número de vezes que essa diferenciação deve ser aplicada. Após a série ser diferenciada d vezes, ela é então modelada com base nos componentes autorregressivos e de média móvel, definidos pelos parâmetros p e q, respectivamente. O parâmetro p corresponde à ordem do modelo autorregressivo, ou seja, à quantidade de observações no passado da série que serão consideradas na predição. Já q representa a ordem do componente da média móvel, isto é, a quantidade de erros (resíduos) em instantes de tempo anteriores incorporados no modelo. Assim, um modelo ARIMA(p, d, q) descreve uma série temporal que foi transformada d vezes por diferenciação para se tornar estacionária, sendo posteriormente ajustada por um processo autorregressivo de ordem p e um processo de média móvel de ordem q.

Em suma, o uso de abordagens proativas representa um avanço significativo na gestão de recursos em ambientes computacionais complexos e dinâmicos. Ao incorporar mecanismos preditivos baseados em modelos como o ARIMA, essas estratégias permitem a adoção de decisões mais informadas, com maior capacidade de adaptação às flutuações da demanda. Isso resulta não apenas em maior eficiência no uso da infraestrutura, mas também em melhor qualidade de serviço, menor taxa de falhas e maior robustez frente a cenários de alta variabilidade. O caráter proativo dessas abordagens torna-se, portanto, essencial em arquiteturas modernas, como as de alocação de recursos em computação de borda e nuvem, onde a previsibilidade e o tempo de resposta são fatores críticos.

2.2 Trabalhos relacionados

Nesta seção, comparamos nossa solução com os trabalhos existentes na literatura que abordam a alocação de recursos e a orquestração de tarefas em infraestruturas de nuvem e borda.

Para a seleção de artigos foram realizadas pesquisas nas bases *Google* Acadêmico, e *IEEE Xplorer* para assuntos relacionados a orquestração de tarefas. Foram utilizados como critérios: 1) o ano de publicação dos artigos com intervalo de 7 anos de 2019 a 2025; 2) a utilização de uma arquitetura em três camadas onde exista a divisão entre camada de geração de tarefas ou camada de dispositivos de usuários, camada de computação de borda (foram aceitos alguns trabalhos que utilizaram o termo névoa no lugar de borda) e a camada de computação na nuvem.

A Tabela 2.1 resume os principais aspectos comparativos desses trabalhos. A primeira coluna lista os artigos de referência, a segunda coluna descreve o tipo de solução empregada, onde PM representa o uso de programação matemática. As quatro últimas colunas indicam os objetivos perseguidos pelos respectivos estudos. As siglas **MF**, **OQ**, **OR** e **MC** correspondem a minimizar falhas, otimizar qualidade de serviço (*QoS*), otimizar utilização de recursos e minimizar custos, respectivamente.

Tabela 2.1: *Trabalhos relacionados. A coluna 1 identifica o artigo. A coluna 2 indica a solução empregada. As colunas 3-6 indicam os objetivos tratados nos trabalhos. **MF**, **OQ**, **OR** e **MC** significam minimizar falhas, otimizar qualidade de serviço, otimizar utilização de recursos e minimizar custos, respectivamente.*

Artigo	Solução proposta	MF	OQ	OR	MC
[33]	Lógica difusa	✓	✓	✓	-
[43]	DDQN	✓	✓	-	-
[20]	MRL	✓	✓	-	-
[3]	K-Medoids	✓	✓	-	-
[2]	DQN / Lógica difusa	✓	-	✓	-
[18]	DRL	✓	✓	-	-
[40]	Heurística	✓	-	-	-
[16]	BRB	✓	✓	✓	-
[27]	Aprendizado de máquina	✓	✓	-	-
[4]	Lógica difusa	✓	✓	✓	-
[34]	Aprendizado de máquina	✓	✓	-	-
[1]	Lógica difusa	✓	✓	-	-
[44]	DQN	✓	✓	-	-
Este trabalho	PM / Heurística / Proativo (ARIMA)	✓	✓	✓	✓

As soluções apresentadas nos trabalhos relacionados apresentam soluções de orquestração que tratam de uma tarefa por vez durante a fase de decisão. O que difere das abordagens propostas neste trabalho, onde um conjunto de tarefas é alocado em conjunto, de maneira a ter maior êxito quando se têm recursos escassos. A maioria dos trabalhos apresenta soluções baseadas em alguma variação de aprendizado por reforço (*Reinforcement Learning* (RL)) [43, 20, 2, 18, 44] ou lógica difusa (*Fuzzy Logic* (FL)) [33, 2, 4, 1].

Dentre os artigos que propõem soluções baseadas em aprendizado por reforço, os autores em [43] realizam a modelagem do problema como uma cadeia de Markov (*Markov Decision Process* (MDP)) e apresentam uma solução baseada em *Double Deep Q-Network* (DDQN) denominada *DeepEdge* onde o objetivo é maximizar a quantidade de tarefas a serem executadas. É apresentada uma formalização do problema, onde são consideradas restrições de latência a serem atendidas por cada tarefa, assim como o

atendimento da capacidade de processamento de cada máquina virtual, embora o modelo não tenha sido implementado na prática.

Meta-aprendizado por reforço (*Meta Reinforcement Learning* (MRL)) é empregado pelos autores em [20] como parte de uma solução em dois estágios, denominada *MRL-TSO*. No primeiro estágio, uma decisão inicial de orquestração é tomada com base em informações preliminares sobre o ambiente. Em seguida, no segundo estágio, essas informações são combinadas com dados atualizados sobre os servidores na borda, a fim de verificar e evitar alocações em servidores sem recursos disponíveis e que possam resultar em falhas de processamento.

Os autores em [18] também empregam *Deep Reinforcement Learning* (DRL) como parte de uma solução de orquestração onde o trabalho considera a interdependência entre múltiplas tarefas, modelando essas relações por meio de grafos acíclicos dirigidos (*Directed Acyclic Graph* (DAG)).

Em [44], aprendizado por reforço profundo (*Deep Q-Learning* (DQN)) também é aplicado como uma solução de orquestração, onde a função de recompensa considera a minimização da latência e da quantidade de tarefas com falhas de execução. Embora o artigo apresente um modelo formalizando o problema abordado, ele não inclui uma avaliação com uma solução exata.

Já dentro da categoria de trabalhos que utilizam lógica difusa, os autores em [33] propõem uma solução de orquestração em duas fases: na primeira fase, é tomada uma decisão para selecionar o nó de borda candidato a receber a carga de trabalho. Na segunda fase, é realizada uma comparação entre o nó de borda selecionado e o nó central da nuvem para determinar onde a carga de trabalho será processada. A solução proposta tem como objetivo minimizar falhas causadas por insuficiência de recursos de rede ou processamento, bem como problemas relacionados à mobilidade, como assincronia entre a associação do usuário e os pontos de acesso durante o *upload* e *download*. Além disso, o estudo analisa o tempo de serviço das tarefas e a utilização dos recursos de processamento.

Os autores em [4] propõem uma solução de orquestração de tarefas geradas por dispositivos IoT com o objetivo principal de reduzir o tempo de serviço e otimizar a utilização de recursos na borda e na nuvem.

Em [1], os autores propõem uma solução composta por três módulos. O módulo de identificação determina o servidor de borda mais adequado para a alocação da tarefa. No módulo de classificação, a tarefa é categorizada como de baixa ou alta demanda. Por fim, no módulo de decisão, os parâmetros obtidos nos módulos anteriores são inseridos no modelo baseado em lógica difusa, que decide se a tarefa deve ser alocada no servidor de nuvem ou no servidor de borda identificado no primeiro módulo.

Os autores em [2] propõem uma solução baseada em ambas as tecnologias, i.e. aprendizado por reforço e lógica difusa. A princípio, a lógica difusa é utilizada para

decidir qual a camada uma tarefa deve ser alocada (névoa local, névoa colaborativa ou nuvem). Em seguida, um algoritmo baseado em DQN é aplicado para selecionar o servidor de névoa mais adequado para o processamento da tarefa. A solução proposta é avaliada com base na utilização da rede, no consumo de energia, na latência e na taxa de execução das tarefas.

Além disso, outros trabalhos apresentam soluções baseadas em aprendizado de máquina (*Machine Learning* (ML)) [34, 27].

Em [34], os autores propõem um modelo de orquestração de tarefas estruturado em duas fases. Na primeira fase, um modelo de classificação é utilizado para prever se uma tarefa deve ser alocada na borda ou na nuvem. Na segunda fase, um modelo de regressão é empregado para prever o tempo de serviço das tarefas alocadas à arquitetura selecionada na primeira fase. O trabalho considera a computação em borda veicular (*Vehicular Edge Computing* (VEC)), onde as tarefas podem ser alocadas tanto na borda quanto na nuvem por meio de unidades de comunicação rodoviária (*Road Side Units* (RSU)) ou através de redes celulares, como a quinta geração de rede móvel celular (5G).

Semelhante à abordagem em [34], os autores em [27] também exploram um cenário de computação em borda veicular e abordam o problema de orquestração utilizando algoritmos de aprendizado de máquina. Os autores analisam e comparam diversos modelos de regressão, incluindo floresta randômica, regressão linear e *Support Vector Machine* (SVM). No entanto, diferentemente do trabalho em [34], o modelo proposto consiste em uma única fase. Nesse caso, a solução proposta, que utiliza apenas um modelo de regressão para prever o tempo de serviço das tarefas alocadas na borda ou na nuvem, demonstra resultados mais eficientes em comparação com sua contraparte.

Por fim, apenas três trabalhos não utilizam aprendizado por reforço, lógica difusa ou aprendizado de máquina como parte da solução de orquestração [3, 40, 16].

Em [3], é proposta uma estratégia que utiliza o algoritmo *K-Medoids* para formar agrupamentos (*clusters*) de servidores na borda, com o objetivo de reunir recursos computacionais suficientes e, assim, minimizar o tempo de processamento das tarefas associadas a cada *cluster*.

Os autores em [40] propõem uma heurística chamada *ORCH* para abordar o problema de orquestração de tarefas em cenários envolvendo servidores de borda não estacionários. Essa abordagem não apenas determina a alocação das tarefas, mas também otimiza o posicionamento dos próprios servidores, garantindo que todas as requisições sejam atendidas com o mínimo de falhas.

Em [16], a técnica *Belief Rule-Based* (BRB) é aplicada para lidar com a alta complexidade e as incertezas inerentes ao problema de orquestração. Essa abordagem se baseia em regras de decisão preestabelecidas para determinar a orquestração das tarefas. Semelhante à metodologia apresentada em [33], o problema é tratado em duas fases,

mantendo o mesmo escopo decisório. O principal objetivo é otimizar a utilização dos recursos computacionais, minimizar falhas e garantir a conformidade com os requisitos de latência.

Com relação ao processo de experimentação, vale destacar que quase todos os artigos (a exceção de [4]) utilizaram alguma ferramenta de simulação como método de avaliação da solução, sendo que a maioria dos trabalhos integrou suas soluções ao *EdgeCloudSim* [33, 43, 3, 40, 16, 27, 4, 34, 1, 44]. Um artigo utilizou iFogSim [2] e outro fez uso de um simulador desenvolvido no escopo do próprio trabalho [20].

Nenhum dos trabalhos utilizou uma abordagem proativa de orquestração de tarefas de modo a prevenir falhas causadas pela escassez de recursos. Apenas três dos trabalhos mapeados formalizaram o problema tratado como um problema de programação matemática com função objetivo e restrições bem definidas [43, 20, 44]. Além disso, nenhum desses trabalhos considera o custo da infraestrutura no processo de tomada de decisão.

2.3 Considerações finais do capítulo

Neste capítulo, foram apresentados os principais conceitos relacionados ao escopo deste trabalho. Discutiu-se a orquestração de tarefas, com ênfase em infraestruturas híbridas. Também foram abordados os fundamentos de realidade mista e as características da aplicação utilizada como estudo de caso. Além disso, apresentou-se uma visão geral sobre simuladores de computação em borda e em nuvem além dos conceitos de alocação proativa de recursos. Por fim, foram descritos os trabalhos relacionados, cujas diferenças em relação a este trabalho foram devidamente mapeadas.

CAPÍTULO 3

Solução localmente exata para o problema de orquestração de tarefas no contexto de aplicações de realidade mista

A Seção 3.1 apresenta o modelo do sistema para o problema de orquestração de tarefas em uma infraestrutura de computação em borda e nuvem. A Seção 3.2 apresenta a formulação do problema e a proposta de uma solução denominada *Local Optimal Task Orchestration Solution (LOTOS)*, onde o problema é resolvido em três estágios, levando em consideração dois objetivos: 1) maximizar o atendimento da demanda; 2) minimizar o custo de utilização da infraestrutura.

3.1 Modelo de sistema

Consideramos que um usuário $u \in \mathcal{U}$ está conectado a um ponto de acesso $a \in \mathcal{A}$ em uma infraestrutura semelhante à mostrada na Figura 1.1. Cada usuário gera um conjunto de tarefas $t \in \mathcal{T}$ caracterizadas por parâmetros como quantidade de instruções a serem processadas, demanda de memória, utilização da CPU e a quantidade de dados enviados ao nó de processamento e recebidos na forma de resultados, ao dispositivo do usuário.

Da mesma forma, a infraestrutura é composta por máquinas virtuais $v \in \mathcal{E} \cup \mathcal{C}$, onde $\mathcal{E}$ é uma máquina localizada na borda e $\mathcal{C}$ é uma máquina localizada na nuvem.

O objetivo geral do modelo é alocar tarefas de usuários às máquinas virtuais para maximizar o número de tarefas executadas com sucesso de acordo com as restrições de rede, atraso, processamento e memória, minimizando o custo de uso da infraestrutura.

3.2 Formulação do problema

Nesta seção, apresentamos uma solução para orquestração de tarefas de maneira localmente exata. Isto significa que as tarefas são agrupadas em lotes que são enviados ao

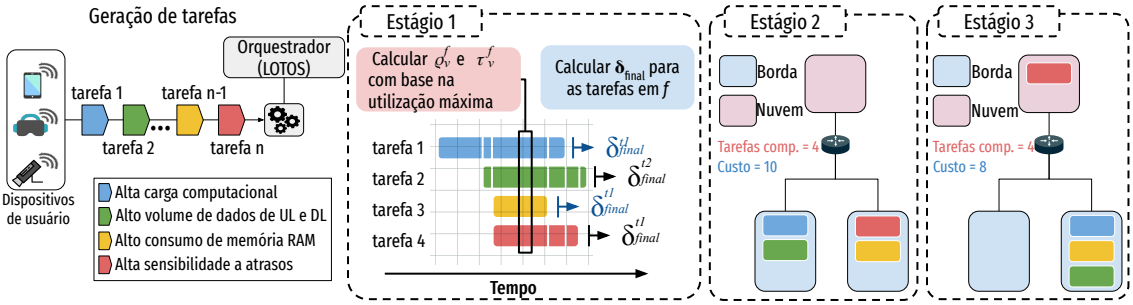


Figura 3.1: *Local Optimal Task Orchestration Solution (LOTOS). O primeira estágio calcula para todas as combinações de tarefas e máquinas virtuais a demanda gerada ao alocarmos essas tarefas em determinada máquina. No segundo estágio buscamos maximizar o atendimento de tarefas. No terceiro estágio queremos minimizar o custo de utilização da infraestrutura.*

orquestrador para alocação. A resolução do problema para um conjunto de tarefas permite a viabilidade de geração de soluções, já que uma tarefa deve esperar até que um lote seja formado antes de ser enviada ao orquestrador. Ou seja, para uma solução global, o tempo de espera até o envio ao orquestrador tenderia ao infinito, considerando um cenário de demanda contínua.

A solução proposta consiste em três estágios. No primeiro estágio, calculamos a quantidade de recursos consumidos por um conjunto de tarefas alocadas a uma máquina virtual. Simultaneamente, dados os recursos disponíveis, determinamos o tempo de processamento das tarefas que compartilham recursos. No segundo estágio, tendo como entrada a saída do primeiro estágio, o objetivo é maximizar o número de tarefas executadas com sucesso, decidindo onde cada tarefa será alocada. Finalmente, no terceiro estágio, o objetivo é minimizar o custo monetário do processamento do número máximo de tarefas identificadas no segundo estágio. Ou seja, a solução alcançada no estágio 2 não é degradada ao final do estágio 3. A Figura 3.1 fornece uma visão geral da solução em três estágios proposta neste trabalho. As subseções a seguir fornecem uma explicação detalhada de cada um dos três estágios.

3.2.1 Estágio 1 - Calculo de configurações e suas demandas

No primeiro estágio do *LOTOS*, precisamos calcular o consumo de recursos e o tempo necessário para processar cada tarefa em uma determinada máquina virtual. Isso ocorre, pois o tempo de processamento de determinada tarefa leva em consideração a quantidade de tarefas alocadas na mesma máquina. Neste sentido, definimos uma configuração f como um elemento do conjunto de potência $\mathcal{F} = \mathbb{P}(\mathcal{T})$, representando uma combinação específica de tarefas, onde os recursos disponíveis na máquina virtual v são compartilhados entre elas.

Para realizar esses cálculos, utilizamos os Algoritmos 3.1 e 3.2, sobre todas as combinações de f e v. As saídas desses algoritmos para todo $f \in \mathcal{F}$ e $v \in \mathcal{V}$ serão usadas na execução dos estágios 2 e 3.

Algoritmo 3.1: Calculo de demanda da configuração

Entrada: $v, f, \Phi^v, \Omega^v, \omega_u^t, \phi_u^t, \delta_{inicial}^{t,u} \quad \forall t \in f$
Saída: $\delta_{final}^{t,u}, \rho_v^f, \tau_v^f$

1 $\mathcal{T}_{ordenado}^v \leftarrow \text{Ordenar}(\delta_{inicial}^{t,u} \quad \forall t \in f)$;
2 $\delta_{atual} \leftarrow \text{Min}(\delta_{inicial}^{t,u} \quad \forall t \in f)$;
3 $k \leftarrow 1 \quad i \leftarrow \mathcal{T}_{ordenado}^v[k] \quad j \leftarrow \mathcal{T}_{ordenado}^v[k+1] \quad \mathcal{T}_{concorrente}^v \leftarrow i$;
4 **enquanto** $(k \neq |\mathcal{T}_{ordenado}^v|)$ **faça**
5 $\quad \delta_{dif} \leftarrow \delta_{inicial}^j - \delta_{inicial}^i$;
6 $\quad$ Atualização da carga de trabalho();
7 $\quad \mathcal{T}_{concorrente}^v \leftarrow \mathcal{T}_{concorrente}^v \cup j \qquad k \leftarrow k+1 \qquad \delta_{atual} \leftarrow \delta_{atual} + \delta_{dif}$;
8 $\quad \rho_v^f \leftarrow \text{Max}(\rho_v^f, \text{CPU}(\mathcal{T}_{concorrente}^v)) \quad , \quad \tau_v^f \leftarrow \text{Max}(\tau_v^f, \text{RAM}(\mathcal{T}_{concorrente}^v))$;
9 $\quad i \leftarrow \mathcal{T}_{ordenado}^v[k] \qquad j \leftarrow \mathcal{T}_{ordenado}^v[k+1]$;
10 **fim**
11 **enquanto** $(\mathcal{T}_{concorrente}^v \neq \emptyset)$ **faça**
12 $\quad i \leftarrow \text{Min}(\omega_u^t \times \phi_u^t \quad \forall t \in \mathcal{T}_{concorrente}^v)$;
13 $\quad cpt \leftarrow \frac{\Phi^v}{|\mathcal{T}_{concorrente}^v| + |\mathcal{T}_{legado}^v|} \qquad cap_u^i \leftarrow \frac{cpt}{\text{Max}(\phi_u^i, cpt)} \times \Omega^v \qquad \delta_{dif} \leftarrow \frac{\omega_u^i}{cap_u^i}$;
14 $\quad$ Atualização da carga de trabalho();
15 $\quad \delta_{atual} \leftarrow \delta_{atual} + \delta_{dif}$;
16 **fim**

O Algoritmo 3.1 descreve o cálculo do tempo de processamento de todas as tarefas $t \in f$ quando atribuídas a uma máquina virtual v. Também é calculado o uso máximo de memória RAM e CPU. Além de f e v, o algoritmo recebe como entrada a quantidade de instruções processadas por segundo (MIPS) e a quantidade de núcleos da máquina virtual v (Ω^v, Φ^v). Também são dados de entrada as demandas de cada tarefa t (ω_u^t, ϕ_u^t) em milhões de instruções (*Million of Instructions* (MI)) e a quantidade de núcleos demandados, respectivamente, além da estimativa do tempo em que cada tarefa irá concluir o *upload* dos dados para v ($\delta_{inicial}^{t,u}$). A saída do algoritmo fornece uma estimativa do tempo em que as tarefas em f terminam de executar toda a sua carga de trabalho.

Inicialmente, as tarefas em f são ordenadas pelos valores $\delta_{inicial}^{t,u}$ (linha 1). A variável δ_{atual} recebe o menor tempo entre todas as tarefas em f (linha 2). O conjunto $\mathcal{T}_{concorrente}^v$ armazena todas as tarefas com cargas de trabalho residuais que competem por

Algoritmo 3.2: Atualização da carga de trabalho

Entrada: $\mathcal{T}^v_{concorrente}, \mathcal{T}^v_{legado}, \delta_{dif}, \delta_{atual}, \Phi^v, \Omega^v, \phi^t_u, \omega^t_u \ \ \forall t \in \mathcal{T}^v_{concorrente}$

Saída: Atualização de $\mathcal{T}^v_{concorrente}$, $\delta^{t,u}_{final}$ e ω^t_u

1 **para** $t \in \mathcal{T}^v_{concorrente}$ **faça**
2 $\quad cpt \leftarrow \frac{\Phi^v}{|\mathcal{T}^v_{concorrente}|+|\mathcal{T}^v_{legado}|} \qquad cap^t_u \leftarrow \frac{cpt}{\text{Max}(\phi^t_u, cpt)} \times \Omega^v$;
3 $\quad$ **se** $\delta_{dif} \times cap^t_u \geq \omega^t_u$ **então**
4 $\quad\quad \mathcal{T}^v_{concorrente} \leftarrow \mathcal{T}^v_{concorrente} - t$;
5 $\quad\quad \delta^{t,u}_{final} \leftarrow \delta_{atual} + \frac{\omega^t_u}{cap^t_u}$;
6 $\quad$ **senão**
7 $\quad\quad \omega^t_u \leftarrow \omega^t_u - (\delta_{dif} \times cap^t_u)$
8 $\quad$ **fim**
9 **fim**

recursos de processamento. Inicialmente, $\mathcal{T}^v_{concorrente}$ é preenchido com a primeira tarefa no conjunto $\mathcal{T}^v_{ordenado}$ (linha 3).

No primeiro laço (linhas 4-8), iteramos pela lista $\mathcal{T}^v_{ordenado}$ enquanto atualizamos $\mathcal{T}^v_{concorrente}$ com as tarefas que começam a demandar recursos computacionais. A variável δ_{dif} representa o período de tempo durante o qual as tarefas em $\mathcal{T}^v_{concorrente}$ competem por recursos. Este valor é usado para diminuir o número restante de instruções para cada tarefa, bem como para atualizar o tempo e determinar quando uma tarefa é concluída (Algoritmo 3.2). Além disso, na linha 8, é realizado o registro do uso máximo de CPU e RAM, comparando o registro de uso mais recente com a quantidade consumida pelas tarefas em $\mathcal{T}^v_{concorrente}$.

O Algoritmo 3.2 atualiza a carga de trabalho restante para cada tarefa t. cap^t_u representa a capacidade de processamento em MIPS disponível para a tarefa t gerada pelo usuário u quando as tarefas no conjunto $\mathcal{T}^v_{concorrente}$ estão sendo processadas simultaneamente. No final do primeiro laço, $\mathcal{T}^v_{concorrente}$ contém as tarefas que ainda não terminaram de processar sua carga de trabalho.

No segundo laço (linhas 9-14), calculamos o tempo de finalização do processamento das tarefas com a menor carga de trabalho restante, enquanto atualizamos simultaneamente a carga de trabalho residual das outras tarefas. $\mathcal{T}^v_{legado}$ representa as tarefas legadas que ainda estão sendo processadas na máquina virtual. Quando $\mathcal{T}^v_{concorrente} = \emptyset$, teremos concluído o cálculo do tempo de processamento final para todas as tarefas em f na máquina virtual v.

3.2.2 Estágio 2 - Maximização de execução de tarefas

Uma vez calculadas as demandas relacionadas às combinações de tarefas f e de máquinas virtuais v, bem como o tempo de processamento estimado para essas

combinações, no estágio 2 essas informações são utilizadas no modelo de orquestração de tarefas representado pelas equações 3-1-3-9.

O objetivo, definido na Equação 3-1, visa maximizar o número de tarefas executadas com sucesso, onde a variável de decisão binária x_v^f indica a alocação do conjunto de tarefas em f na máquina v, onde $x_v^f = 1$ se o conjunto f for alocado na máquina virtual v, e $x_v^f = 0$ caso contrário. A função $A(x_v^f)$ retorna a quantidade total de tarefas em f.

Cada máquina virtual, seja instanciada na borda ou na nuvem, tem capacidades de processamento definidas por P_v. Além disso, o consumo de recursos computacionais quando um conjunto de tarefas f é executado em uma máquina virtual v é representado por ρ_v^f. A equação 3-2 estabelece que a soma dos recursos computacionais consumidos ao alocarmos as tarefas em f na máquina virtual v não deve exceder sua capacidade P_v.

Da mesma forma, a restrição 3-3 garante que a soma da quantidade de memória RAM demandada por cada tarefa, denotada como τ_v^f, não exceda a capacidade da máquina virtual T_v. O consumo de CPU e RAM, representados por ρ_v^f e τ_v^f, são calculados no Algoritmo 3.1, que registra o consumo máximo de recursos durante a execução da carga de trabalho das tarefas.

As equações 3-4-3-6 estabelecem as restrições de capacidade para os enlaces WLAN, WAN e MAN, respectivamente. Consideramos a utilização da infraestrutura de comunicação semelhante ao que foi definido pelos autores em [33]. O enlace WLAN é utilizado quando uma tarefa é alocada em qualquer máquina virtual dentro da infraestrutura de borda. O enlace WAN é empregado quando a tarefa é atribuída a uma máquina virtual instanciada na infraestrutura da nuvem, enquanto o enlace MAN é utilizado por tarefas alocadas em máquinas na borda, mas que não estão co-localizadas com o ponto de acesso em que o usuário está associado.

Cada ponto de acesso apresenta enlaces de comunicação WLAN e WAN dedicados: o enlace WLAN conecta usuários dentro de sua área de cobertura ao ponto de acesso, e o enlace WAN conecta usuários à infraestrutura de nuvem.

Os parâmetros ζ_a^{wlan} e ζ_a^{wan} definem o número máximo de usuários que podem ser suportados pelos enlaces WLAN e WAN, respectivamente, e são fornecidos como valores de entrada para o modelo de orquestração. As funções $W_{wlan}(x_v^f, a)$ e $W_{wan}(x_v^f, a)$ calculam o número de usuários transmitindo dados com destino à máquina virtual v, pelos enlaces WLAN e WAN, no ponto de acesso a, de acordo com a variável x_v^f.

A Equação 3-6 estabelece o limite de uso para o enlace MAN, que é compartilhado entre todos os pontos de acesso. Os parâmetros $\zeta_{ul}^{man} = \frac{B_{man}}{\eta_{ul}}$ e $\zeta_{dl}^{man} = \frac{B_{man}}{\eta_{dl}}$ especificam o número máximo de tarefas que podem ser transmitidas no enlace MAN em *upload* e *download*, respectivamente. B_{man} representa a largura de banda do enlace MAN, enquanto η_{ul} e η_{dl} representam o tamanho médio dos dados enviados no enlace em *upload*

e *download*, respectivamente. A função $W_{man}(x_v^f)$ representa o número de tarefas transmitindo dados através do enlace MAN, de acordo com a variável de decisão x_v^f.

A restrição 3-7 garante que o tempo de serviço total de cada tarefa permaneça abaixo do limite de entrada predefinido. O tempo de serviço total desde a geração da tarefa pela aplicação até o envio dos resultados de volta para o dispositivo do usuário compreende três componentes: 1) o atraso de comunicação, que considera o tempo necessário para transmitir uma tarefa para uma máquina virtual somado ao tempo de envio dos resultados para o dispositivo do usuário; 2) o atraso de processamento, que reflete o tempo necessário para concluir a carga de trabalho da tarefa; e 3) o tempo de espera antes que uma tarefa seja enviada ao orquestrador.

A função $M(x_v^f,t,u) = 1$ se a tarefa t, gerada pelo usuário u, estiver incluída no conjunto f, e $M(x_v^f,t,u) = 0$ caso contrário. Os termos $\Gamma_{com}^{t,u,v}$, $\Gamma_{proc}^{t,u,v}$ e $\Gamma_{esp}^{t,u}$ representam o tempo estimado de *upload* e *download* dos dados, o tempo de processamento da carga de trabalho e o tempo de espera, respectivamente.

$$\text{Maximizar} \quad \sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}} x_v^f \times A(x_v^f) \tag{3-1}$$

$$\text{Sujeito a:} \quad \sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}} (x_v^f \times \rho_v^f) \leq P_v, \tag{3-2}$$

$$\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}} (x_v^f \times \tau_v^f) \leq T_v, \tag{3-3}$$

$$\sum_{a\in\mathcal{A}}\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{E}} (x_v^f \times W_{wlan}(x_v^f,a)) \leq \zeta_a^{wlan}, \tag{3-4}$$

$$\sum_{a\in\mathcal{A}}\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{C}} (x_v^f \times W_{wan}(x_v^f,a)) \leq \zeta_a^{wan}, \tag{3-5}$$

$$\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{E}} (x_v^f \times W_{man}(x_v^f)) \leq Min(\zeta_{ul}^{man} - L_{ul},\ \zeta_{dl}^{man} - L_{dl}), \tag{3-6}$$

$$(\Gamma_{com}^{t,u,v} + \Gamma_{proc}^{t,u,v} + \Gamma_{esp}^{t,u}) \times x_v^f \times M(x_v^f,t,u) \leq \Delta_u^t, \quad \forall f \in \mathcal{F}, t \in \mathcal{T}, u \in \mathcal{U}, v \in \mathcal{V}, \tag{3-7}$$

$$\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}} x_v^f \times M(x_v^f,t,u) \leq 1, \forall t \in \mathcal{T}, u \in \mathcal{U}, \tag{3-8}$$

$$\sum_{f\in\mathcal{F}} x_v^f \leq 1, \forall v \in \mathcal{V}. \tag{3-9}$$

O tempo de comunicação é calculado usando as seguintes equações:

$$\Gamma_{com}^{t,u,v} = \frac{\alpha_u^t + \beta_u^t}{B_{wlan}^{u,a(t)}} \quad , \quad \frac{\alpha_u^t + \beta_u^t}{B_{wan}^{u,a(t)}} + \Gamma_{prop}^{wan} \quad , \quad \frac{1}{\zeta_{ul}^{man} - \lambda} + \frac{1}{\zeta_{dl}^{man} - \lambda} + \Gamma_{prop}^{man}, \tag{3-10}$$

onde cada equação representa o cálculo do tempo estimado de transferência

dos dados, para os enlaces WLAN, WAN e MAN, respectivamente. α_u^t e β_u^t denotam o tamanho da tarefa durante *upload* e *download*, respectivamente, enquanto $B_{wlan}^{u,a(t)}$ e $B_{wan}^{u,a(t)}$ representam a largura de banda por usuário dos enlaces WLAN e WAN. O termo Γ_{prop}^{wan} corresponde ao atraso de propagação do enlace WLAN.

A terceira equação calcula o atraso de comunicação para o enlace MAN, modelado como um sistema de enfileiramento MMPP/M/1. Neste contexto, λ representa uma estimativa do número de tarefas utilizando o enlace MAN por segundo. O termo Γ_{prop}^{man} representa o atraso de propagação do enlace MAN.

O tempo de processamento é determinado usando a seguinte equação:

$$\Gamma_{proc}^{t,u,v} = \delta_{final}^{t,u} - \delta_{inicial}^{t,u}, \tag{3-11}$$

onde $\delta_{inicial}^{t,u}$ é fornecido como entrada, e $\delta_{final}^{t,u}$ é calculado no estágio 1 usando os Algoritmos 3.1 e 3.2.

Por fim, a Equação 3-8 garante que o modelo produza uma única solução para cada tarefa t gerada pelo usuário u, enquanto a Equação 3-9 garante que, no máximo, um conjunto de tarefas f seja alocado a v.

3.2.3 Estágio 3 - Minimização do custo de infraestrutura

A solução apresentada no segundo estágio determina o destino de cada tarefa, visando minimizar falhas e otimizar a utilização dos recursos de armazenamento, processamento e comunicação, ao mesmo tempo em que garante o atendimento aos requisitos de QoS específicos da aplicação. No estágio 3, buscamos minimizar o custo de implementação da infraestrutura, sem degradar os resultados alcançados no estágio 2.

A Equação 3-12 representa a função objetivo de minimização do custo total do uso de máquinas virtuais para processamento de tarefas. O custo c^v representa o valor monetário por unidade de tempo que uma máquina virtual incorre para processar tarefas. A função $T(x_v^f)$ denota o tempo total para que a máquina virtual v processe a carga de trabalho das tarefas em f. A equação 3-13 garante que o número de tarefas executadas sem falhas permaneça semelhante ao resultado obtido no estágio 2. Neste contexto, x_v^{f*} representa as variáveis de decisão correspondentes à solução ótima obtida no estágio 2.

$$\text{Minimizar} \quad \sum_{f\in\mathcal{F}} \sum_{v\in\mathcal{V}} x_v^f \times T(x_v^f) \times c^v \tag{3-12}$$

$$\text{Sujeito a:} \quad \sum_{f\in\mathcal{F}} \sum_{v\in\mathcal{V}} x_v^f \times A(x_v^f) = \sum_{f\in\mathcal{F}} \sum_{v\in\mathcal{V}} x_v^{f*} \times A(x_v^{f*}), \tag{3-13}$$

$$\text{Equações} \quad 3\text{-}2\text{-}3\text{-}9 \tag{3-14}$$

3.3 Análise de complexidade

Assumindo o pior caso, a complexidade do estágio 1 do *LOTOS* é dada por $O(|\mathcal{V}||\mathcal{F}|O(A1))$, em que $\mathcal{V}$ representa a quantidade de máquinas virtuais disponíveis, $\mathcal{F}$ representa o conjunto de combinações possíveis por lote, com cardinalidade $|\mathcal{F}| = 2^{|\text{tamanho do lote}|}$, e $O(A1)$ denota a complexidade do Algoritmo 3.1.

A complexidade de A1 é definida como $O(|f|O(A2) + |f|O(A2))$, onde $f \in \mathcal{F}$ representa uma combinação específica e, no pior caso, seu tamanho é igual ao tamanho do lote. A complexidade do Algoritmo 3.2, representada por $O(A2)$, é definida como $O(|f|)$, uma vez que, no máximo, o Algoritmo 3.2 irá iterar sobre todas as tarefas no conjunto f.

Para analisar a complexidade dos estágios 2 e 3, mostraremos que é possível realizar uma redução em tempo polinomial de ambos para uma variação do problema de atribuição com restrições laterais (*Assignment Problems with Side Constraints* (APSC)), denominado *Assignment Problems with Side Constraints and Agent Qualification* (APSCAQ) [28]. Embora a redução possa ser conduzida de forma semelhante para ambos os estágios, optamos por realizá-la com base apenas na formulação do estágio 2, por uma questão de simplicidade.

APSCAQ: Seja $\mathcal{M}$ o conjunto de tarefas a serem atribuídas a $\mathcal{N}$ agentes, e seja c_{ij} o custo associado à designação da tarefa i ao agente j, o objetivo do APSC é atribuir cada tarefa a um agente, buscando maximizar um determinado custo agregado. Na variação do problema, considera-se adicionalmente a qualificação do agente j para executar a tarefa i. Essa qualificação é indicada por q_{ij}, onde $q_{ij} = 1$ se o agente possui a habilidade necessária para receber a tarefa, e $q_{ij} = 0$ caso contrário. A formulação geral do APSCAQ é dada por:

$$\text{Maximizar} \quad \sum_{i \in \mathcal{M}} \sum_{j \in \mathcal{N}} c_{ij} x_{ij} \tag{3-15}$$

$$\text{Sujeito a:} \quad \sum_{i \in \mathcal{M}} q_{ij} x_{ij} \leq 1 \quad , \quad j \in \mathcal{N}, \tag{3-16}$$

$$\sum_{j \in \mathcal{N}} q_{ij} x_{ij} \leq 1 \quad , \quad i \in \mathcal{M}, \tag{3-17}$$

$$x_{ij} \in \{0, 1\}. \tag{3-18}$$

Resolver o problema APSCAQ é uma tarefa de complexidade NP-completa [28, 23]. Assim, buscamos demonstrar que o estágio 2 do LOTOS (bem como o estágio 3) também é NP-completo, por meio de uma redução em tempo polinomial do APSCAQ para o problema formulado no estágio 2.

Redução: Ao analisarmos as equações 3-16 e 3-17 do APSCAQ, observamos que cada agente pode receber, no máximo, uma tarefa, e cada tarefa pode ser atribuída a

no máximo um agente. Essa característica também está presente no problema formulado no estágio 2, por meio das restrições 3-8 e 3-9. A restrição 3-8 assegura que cada configuração seja alocada em uma única máquina virtual, enquanto a restrição 3-9 garante que cada máquina virtual receba apenas uma configuração.

A qualificação do agente, representada por q_{ij}, pode ser mapeada para as restrições 3-2 e 3-7, de modo que a alocação de uma configuração em uma máquina virtual deve seguir a qualificação da máquina virtual de acordo com essas restrições. Em outras palavras, para que uma configuração f possa ser atribuída a uma máquina virtual v ($q_{fv} = 1$), devem ser atendidas as restrições de processamento (3-2), memória RAM (3-3), capacidade de enlace (3-4 a 3-6) e latência (3-7). Caso contrário, a máquina virtual v não está qualificada para hospedar as tarefas que compõem a configuração f ($q_{fv} = 0$).

O custo c_{ij} pode ser mapeado para a função $A(x_v^f)$, que compõe a função objetivo na Equação 3-1, de forma que maximizar a quantidade de tarefas atendidas torna-se equivalente a maximizar o custo definido no APSCAQ.

Concluímos, portanto, que é possível transformar uma instância do APSCAQ em uma instância do problema abordado no estágio 2, em tempo polinomial. Dessa forma, demonstramos que o problema tratado no estágio 2 é NP-completo.

3.4 Simulação

Neste trabalho, realizamos uma avaliação extensiva do desempenho do *LOTOS*, comparando-o com quatro abordagens: 1) solução baseada em lógica difusa (*FUZZY*), proposta em [33]; 2) solução baseada em DDQN (*DeepEdge*), proposta em [43]; 3) solução que prioriza a orquestração de tarefas em máquinas menos carregadas (*UTIL.*); e 4) solução que seleciona a máquina virtual com o menor custo (*MIN. COST*).

Como já mencionado no Capítulo 2, para esta avaliação, usamos o *EdgeCloudSim*, como ferramenta de simulação e avaliação das soluções de orquestração. Também, como foi detalhado no Capítulo 2 empregamos o gerador de carga apresentado pela caracterização do MR-Leo [38].

A implementação dos modelos foi realizada usando *Java* 21.0.5 e *Python* 3.10. Também foi utilizada a ferramenta de otimização *CPLEX* (versão 22.1.0) para resolver os problemas de programação matemática, além do *docplex* (versão 2.23.221) para modelagem das soluções.

A Tabela 3.1 apresenta os parâmetros que caracterizam quatro aplicações conforme definido em [39, 33]. Os símbolos $\mathcal{E}$ e $\mathcal{C}$ representam borda e nuvem, respectivamente. *RM1* e *RM2* representam a caracterização de dois módulos disponíveis na aplicação MR-Leo. Em *RM1*, está sendo representada a captura de uma cena dinâmica e a geração de uma nuvem de pontos que representa a superfície da cena. A caracterização

em *RM2* representa a sobreposição de um elemento virtual sobre a cena real capturada, constituindo a principal característica das aplicações de realidade mista, conforme detalhado no Capítulo 2.

Além da caracterização do MR-Leo os autores em [39] consideram a existência de serviços de diferentes tipos concorrendo por recursos com a aplicação de realidade mista. As aplicações *APP2* (serviço relacionado à saúde), *APP3* (serviço de computação intensiva) e *APP4* (serviço relacionado ao entretenimento) representam outros serviços que compartilham recursos com o MR-Leo.

Tabela 3.1: *Caracterização de quatro aplicações que geram a demanda utilizada na avaliação dos modelos de orquestração.* ***RM1*** *e* ***RM2*** *representam módulos do MR-Leo.* ***APP2***, ***APP3*** *e* ***APP4*** *representam outros tipos de serviços concorrendo por recursos.*

	RM1	***RM2***	***APP2***	***APP3***	***APP4***
Intevalo de chegada	33ms	$\lambda = 5(min)$	$\lambda = 3s$	$\lambda = 20s$	$\lambda = 7s$
Upload α^t (kB)	41	0,06	20	2500	20
Download β^t (kB)	41	0,06	1250	25	1000
Limite de tempo Δ_u^t (ms)	33	66	100	500	1000
Taxa de geração (%)	30		20	20	30
Tamanho da tarefa ω_u^t (MI)	$\mathcal{N}(47\%)$ e $\mathcal{N}(53\%)$		3000	45000	15000
CPU $\mathcal{E}$ / $\mathcal{C}$ (%)	6/0,6		2/0,2	30/3	10/1
Núcleos requisitados ϕ_u^t (#)	3		1		
Memória RAM (GB)	2,4 + 1,3		0 + 1,3		

A linha 1 na Tabela 3.1 representa o intervalo de tempo entre cada tarefa gerada por um usuário. Para *RM1* é considerado um intervalo de chegada constante de 33 ms entre as tarefas, enquanto em *RM2*, os intervalos seguem uma distribuição de *Poisson*, com valor médio de 5 tarefas geradas por minuto. Para *APP2*, *APP3* e *APP4* os intervalos de chegada seguem uma distribuição exponencial com valores médios de 3, 20 e 7 segundos entre tarefas geradas pelo mesmo usuário.

Nas linhas 2 e 3, é definida a quantidade de dados enviados à máquina virtual e retornados na forma de resultados para o dispositivo de usuário em kB. Na linha 4, definimos o limite de tempo de serviço para cada tipo de tarefa. A linha 5 representa a porcentagem de tarefas geradas, onde a soma dos valores representa 100% da demanda. Na linha 6, o número de instruções necessárias para processar cada tarefa é especificado.

Para *RM1* e *RM2*, o tamanho da tarefa é definido seguindo uma distribuição normal. Para 47% das tarefas de realidade mista, o tamanho da tarefa é gerado de maneira aleatória seguindo uma distribuição normal com média de 155,62 MI e com desvio padrão de 14,10. Da mesma forma, para os 53% restantes das tarefas, o tamanho segue uma distribuição normal com média de 322,38 MI e um desvio padrão de 71,18.

A linha 7 representa a porcentagem de uso de CPU pela aplicação na borda e na nuvem. A linha 8 define a quantidade de núcleos requisitada. A linha 9 define o uso de memória RAM, que é definido em 1,3 GB para executar as operações gerais do MR-Leo, além de 2,4 GB adicionais para cada dispositivo de usuário conectado e usando os serviços *RM1* ou *RM2*. Para as aplicações *APP2*, *APP3* e *APP4* consideramos que apenas é utilizada a memória relacionada à inicialização da aplicação, sem valores adicionais para cada dispositivo utilizando a aplicação.

A Tabela 3.2 fornece parâmetros adicionais que definem a configuração dos experimentos. As duas primeiras colunas descrevem as características das máquinas virtuais, incluindo o número de núcleos de processamento (linhas 1 e 2), memória RAM (linhas 3 e 4), velocidade de processamento (linha 6) e custo de utilização por hora (linhas 7 e 8), além do tipo do modelo de mobilidade usado (mais detalhes no Capítulo 2).

Para os parâmetros de quantidade de núcleos, memória RAM e custo de utilização por hora foram utilizadas instâncias reais extraídas do serviço da *AWS* [5]. As instâncias utilizadas foram *Region Cloud (t4g.xlarge, r5ad.xlarge, r6g.2xlarge, r5ad.4xlarge*) localizadas em *US East (Ohio)* e *Wavelenght Zone (Far Edge) (t3.xlarge, r5d.xlarge, r5.2xlarge, r5d.4xlarge)* localizadas em *US East (Verizon) - Washington DC*.

As duas últimas colunas apresentam características das instâncias consideradas nos experimentos, incluindo o tempo de simulação e aquecimento (*warmup*) (linha 1), a quantidade de servidores em cada infraestrutura (linha 2), a quantidade de máquinas virtuais em cada infraestrutura (linha 3), a largura de banda nos enlaces (linhas 4 e 5), o limite de tempo para geração do lote (linha 6), o tamanho máximo do lote (linha 7) e a quantidade de dispositivos (linha 8).

Cada experimento tem uma duração virtual de 11 segundos, sendo 1 segundo como período de aquecimento antes da geração dos resultados, com o número de dispositivos de usuário por instância variando de 200 a 2400. O intervalo do lote representa a maior diferença de tempo entre tarefas de um mesmo lote. O tamanho do lote define o número máximo de tarefas agrupadas em um único lote.

Tabela 3.2: *Parâmetros do experimento.*

Parâmetros	**Valores**	**Parâmetros**	**Valores**
Φ^v $\mathcal{E}$ por VM (#)	4/4/8/16	simulação/*warmup* (s)	11/1
Φ^v $\mathcal{C}$ por VM (#)	4/4/8/16	serv. $\mathcal{E}/\mathcal{C}$ (#)	4/1
T_v $\mathcal{E}$ por VM (GB)	16/32/64/128	VMs $\mathcal{E}/\mathcal{C}$ (#)	2/4
T_v $\mathcal{C}$ por VM (GB)	16/32/64/128	Banda WLAN/WAN	empírico
Ω^v $\mathcal{E}/\mathcal{C}$ (MIPS)	10000/100000	Banda MAN	MMPP/M/1
c^v $\mathcal{E}$ ($/hora)	0,134/0,262/0,403/1,048	Intervalo do lote (ms)	5
c^v $\mathcal{C}$ ($/hora)	0,224/0,389/0,68/1,555	Tamanho do lote (#)	6
Mobilidade	Nômade	Dispositivos (#)	200-2400

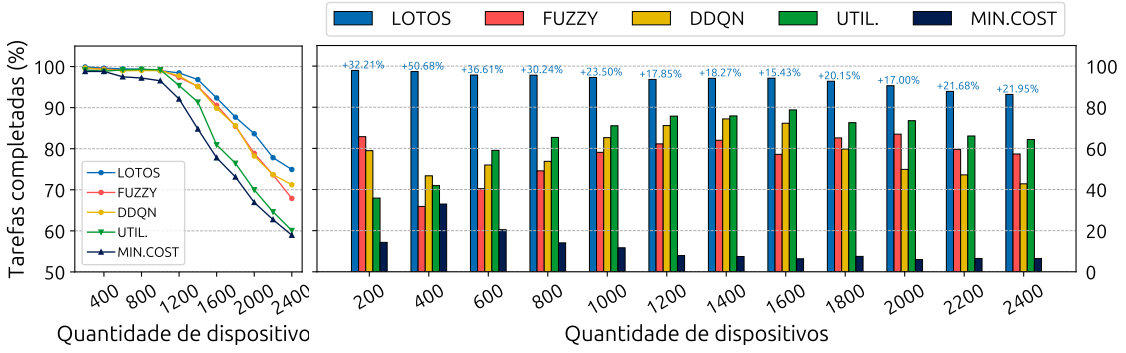
Todos os valores especificados nas Tabelas 3.1 e 3.2 representam o *cenário principal* considerado neste trabalho. Porém, como será visto na Seção 3.5, nós avaliamos os modelos de orquestração também para um segundo cenário que representa uma aproximação ao que foi considerado pelos autores em [33, 43]. Esse segundo cenário será chamado *cenário original*. No *cenário original*, o uso de memória RAM e os limites de atraso para cada tarefa não são considerados (i.e. $\tau_u^t = 0$ e $\Delta_u^t = \infty$), e todas as máquinas virtuais na borda são configuradas com o mesmo número de núcleos (i.e. $\Phi^v\ \mathcal{E} = 16$ e $\Phi^v\ \mathcal{C} = 4$). Essa configuração foi escolhida para garantir um ambiente de avaliação consistente, correspondendo ao apresentado em [33, 43].

3.5 Avaliação

Nesta seção, iremos avaliar nossa solução em relação à taxa de sucesso na execução das tarefas, custo de utilização das máquinas virtuais, utilização dos recursos na borda e na nuvem e o tempo de serviço experimentado pelos usuários.

A Figura 3.2 ilustra a porcentagem de tarefas concluídas e que não encontraram falhas, em dois cenários distintos. Tanto no *cenário original* quanto no *cenário principal* o *LOTOS* consegue sobressair a todas as outras soluções. Conforme mostrado na Figura 3.2(b), nossa solução obtém uma melhoria significativa na execução bem-sucedida de tarefas, com aumentos variando de 15,43% a 50,68%.

A partir das diferenças observadas nos resultados das Figuras 3.2(a) e 3.2(b) podemos concluir que os elementos não tratados nas soluções *FUZZY* e *DDQN* tiveram grande impacto na piora de seus resultados no *cenário principal*.



(a) *Cenário original.* (b) *Cenário principal.*

Figura 3.2: *Porcentagem de tarefas completadas. No cenário original estamos representando as características apresentadas pelos autores em [33, 43]. No segundo cenário, consideramos a variação nas características das máquinas virtuais além de considerar falhas relacionadas a escassez de memória RAM e ao tempo de serviço da tarefa.*

Tanto a solução *FUZZY* quanto *DDQN* não incorporam a utilização de memória RAM como critério de falhas no processamento de tarefas. E, embora os trabalhos considerem latência no processo de orquestração, a nossa solução consegue obter melhores resultados ao tratar esses critérios como restrições a serem seguidas. Além disso, a capacidade de nossa abordagem de orquestrar várias tarefas simultaneamente durante o processo de tomada de decisão permite uma estratégia de alocação mais precisa e planejada em comparação com a abordagem reativa dos modelos *FUZZY* e *DDQN*.

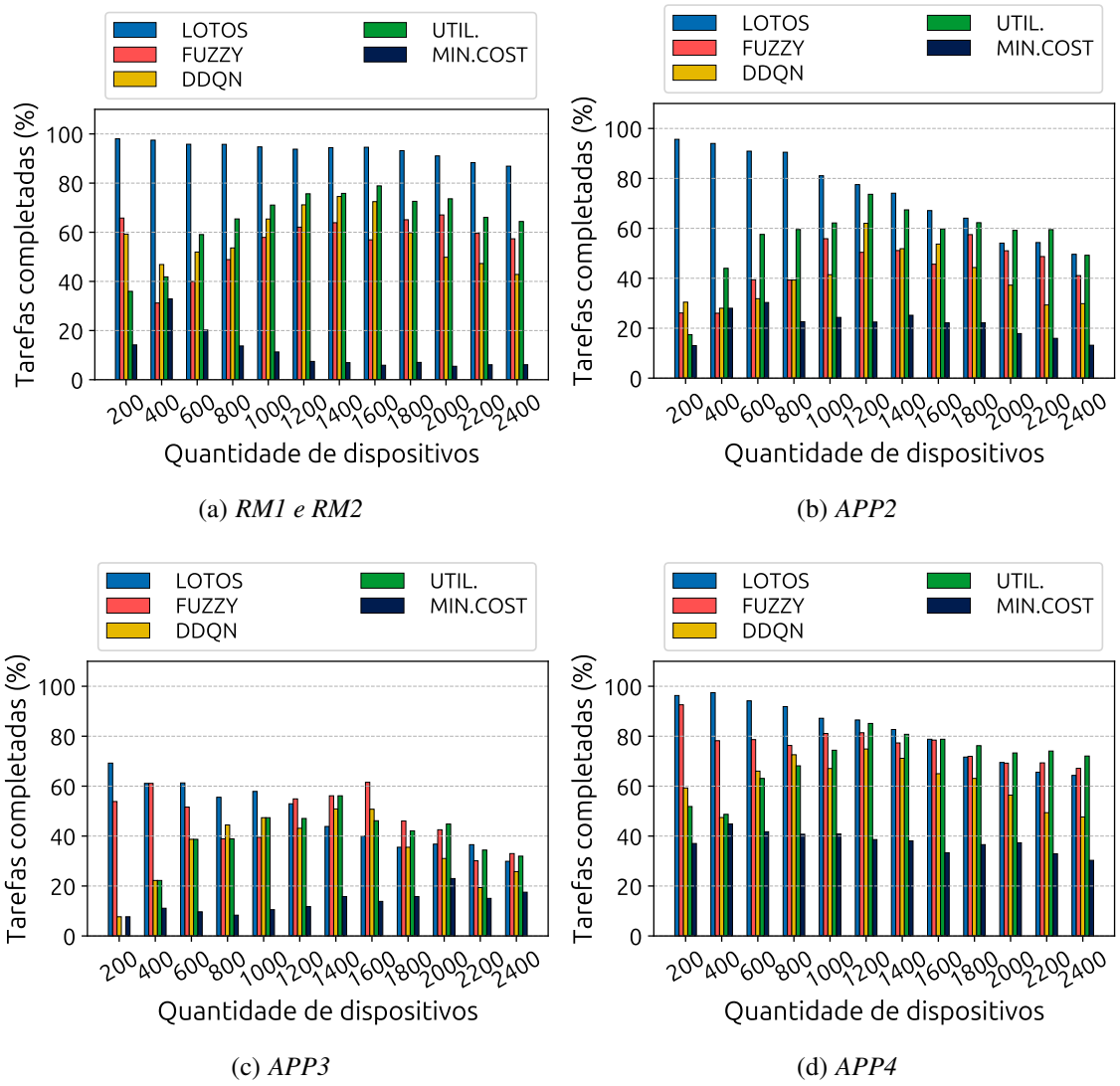
Os resultados apresentados na Figura 3.3 mostram a porcentagem de tarefas executadas sem falhas, porém considerando as aplicações de maneira separada. Podemos notar que para determinadas aplicações (*APP2*, *APP3* e *APP4*) o *LOTOS* não apresenta os melhores resultados em todas as instâncias, embora sempre se mostre competitivo. Isso corrobora com a hipótese levantada no Capítulo 1 sobre a priorização de serviços em cenários de escassez de recursos com o intuito de atender à maior demanda. Como a maioria das tarefas é gerada para aplicações de realidade mista (o intervalo de chegada entre tarefas é menor), esse tipo de serviço foi priorizado com o intuito de atender à maior demanda. Por isso, para os serviços *RM1* e *RM2*, que se caracterizam por serem os mais restritos, o *LOTOS* apresentou melhores resultados em todas as instâncias.

A Figura 3.4 compara os tipos de falhas observadas durante a simulação para todas as instâncias. Na abordagem *FUZZY*, 24,4% dos erros foram causados por violações de limite de latência, enquanto 16% resultaram de sobrecarga de memória RAM. Da mesma forma, a solução *DDQN* apresentou 37,5% de falhas relacionadas a violações do limite de latência. Conforme mencionado anteriormente, essas abordagens não consideraram esses fatores como restrições durante a tomada de decisão, levando a uma maior taxa geral de erros.

O modelo *UTIL.* teve melhor desempenho do que *FUZZY*, *DDQN* e *MIN. COST* no *cenário principal*, alcançando 69,4% das tarefas concluídas. Em contraste, o modelo *MIN. COST* apresentou o pior desempenho, com apenas 8,8% das tarefas concluídas. Este resultado inferior confirma que selecionar a máquina mais barata não é a melhor estratégia, como evidenciado pelos 73,6% de erros causados por sobrecarga de memória RAM nesta abordagem. O *LOTOS* superou todas as outras abordagens, alcançando 91,5% das tarefas concluídas. Os erros causados por conta dos limites de atraso foram mínimos, em apenas 5,4% das tarefas. Isso é compreensível, pois o *LOTOS* emprega uma solução localmente ótima.

Observamos que aproximadamente 3% de todas as tarefas em nossa abordagem não foram concluídas devido à rejeição durante o processo de otimização. Isso indica que, durante os estágios 2 e 3, o modelo identificou a impossibilidade de executar essas tarefas dentro das restrições fornecidas e optou por não alocar recursos.

Enquanto as outras soluções identificam os erros durante a execução da simula-



(a) *RM1 e RM2*

(b) *APP2*

(c) *APP3*

(d) *APP4*

Figura 3.3: *Porcentagem de tarefas completadas por aplicação. Cenário principal.*

ção, nossa solução identifica de maneira preventiva as falhas de modo que os recursos não são desperdiçados. A tomada de decisão preventiva garante uma utilização mais eficiente dos recursos, pois eles não são alocados para tarefas que apresentariam falhas na execução e, como consequência, os resultados não seriam recebidos pelos usuários. Além disso, ao considerarmos os custos de utilização dos recursos da infraestrutura (um aspecto mais detalhado nos próximos resultados), essa alocação eficiente de recursos desempenha um papel crucial na redução das despesas gerais da infraestrutura.

A Figura 3.5(a) apresenta o custo por tarefa concluída, demonstrando que nossa solução atinge maior eficiência na alocação de recursos ao concluir um número maior de tarefas sem falhas. Nossa abordagem reduz os custos em até aproximadamente 8 vezes em comparação ao modelo *UTIL.*, mesmo com eficiência superior na conclusão de tarefas.

A Figura 3.5(b) ilustra a eficiência de utilização de recursos, medida como a porcentagem de tarefas que consomem recursos e são executadas com sucesso. Essa

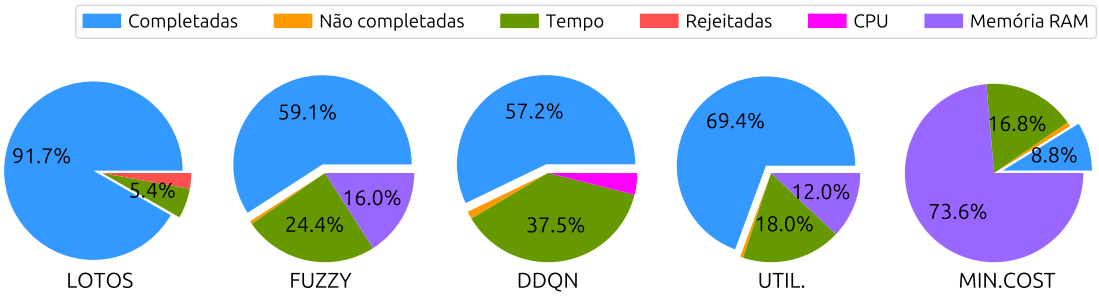
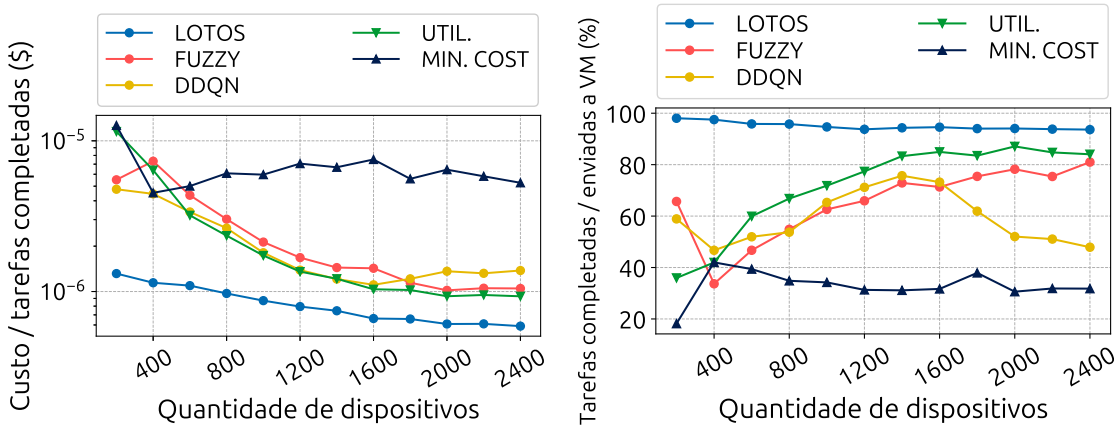


Figura 3.4: *Taxa de erros por categoria. Cenário principal.*

eficiência é calculada dividindo o número de tarefas aceitas pelo número total de tarefas enviadas a uma máquina virtual. Nos resultados do *LOTOS*, pelo menos 90% das tarefas enviadas a uma máquina virtual são executadas com sucesso, enquanto todas as outras soluções exibem níveis de eficiência abaixo de 90% em todas as instâncias.

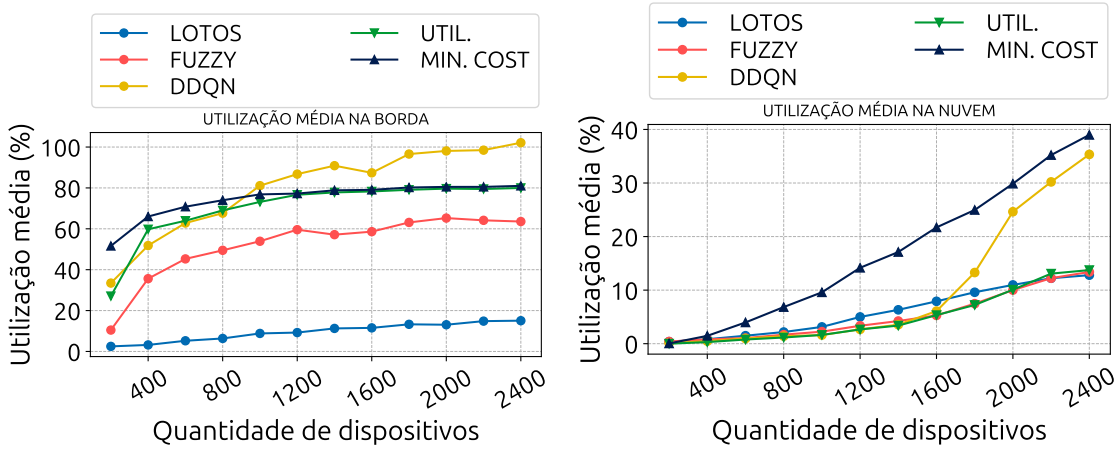


(a) *Custo por tarefa completada.* (b) *Eficiência de tarefas executadas.*

Figura 3.5: *Custo da infraestrutura e eficiência do modelo. Cenário principal.*

Os resultados apresentados na Figura 3.6 mostram a utilização média de recursos na borda e na nuvem. Conforme observado, nossa solução exibe uma utilização significativamente menor na borda, ao mesmo tempo em que continua sendo um forte concorrente no uso dos recursos da nuvem. Isso contribui para minimizar os custos de infraestrutura, apesar de nossa solução atingir um maior número de tarefas concluídas.

Por fim, a Figura 3.7 ilustra o tempo de serviço médio considerando as tarefas que foram executadas. Os resultados mostram que, além dos benefícios alcançados em relação ao atendimento de demanda, consumo de recursos e custos, nossa abordagem também se sobressai com melhores valores de qualidade de serviço em comparação a outras estratégias de orquestração.



(a) *Borda* (b) *Nuvem*

Figura 3.6: *Utilização média na borda e na nuvem. Cenário principal.*

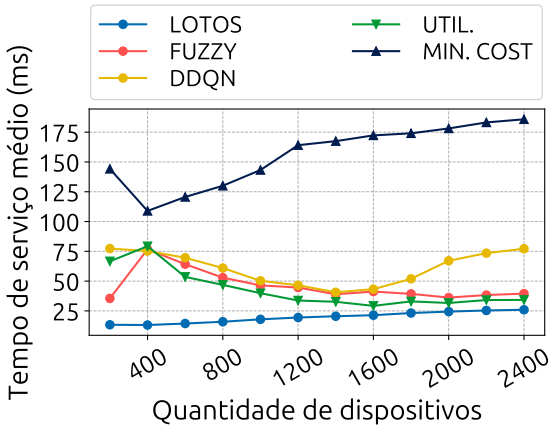


Figura 3.7: *Tempo de serviço médio em relação ao número de tarefas executadas. Cenário principal.*

3.6 Considerações finais do capítulo

Neste capítulo, foi apresentada uma solução de orquestração de tarefas e alocação de recursos, estruturada em três estágios e composta por dois modelos de otimização. No primeiro modelo, buscou-se maximizar o atendimento à demanda dos usuários, enquanto no segundo, o objetivo foi minimizar os custos de utilização da infraestrutura. Por fim, a solução proposta foi avaliada em comparação com outras abordagens, incluindo dois trabalhos relevantes da literatura.

CAPÍTULO 4

Reformulação do modelo de otimização

Neste capítulo, propomos modificações na formulação original do problema de orquestração, com o objetivo de otimizar o modelo em relação ao tempo de geração de soluções. Para isso, simplificamos o modelo de alocação, reduzindo-o para versões com dois e um estágio.

4.1 Solução em dois estágios

O primeiro passo para a simplificação do modelo de alocação de recursos consiste na reformulação do problema em dois estágios. No Capítulo 3, os estágios 2 e 3 possuem objetivos distintos: o estágio 2 busca maximizar a quantidade de tarefas aceitas, enquanto o estágio 3 visa minimizar o custo da infraestrutura. No entanto, é possível unificar esses dois objetivos em uma única função objetivo, por meio da normalização da função do estágio 3. O resultado dessa unificação é apresentado na Equação 4-1, onde a primeira parte da formulação corresponde à maximização das tarefas processadas com sucesso, enquanto a segunda parte representa a minimização dos custos da infraestrutura. A normalização é realizada dividindo-se o custo obtido pela solução pelo maior custo possível, o que impede que o objetivo de minimizar o custo interfira no objetivo de maximizar a quantidade de tarefas processadas com sucesso.

A Equação 4-2 apresenta o custo máximo possível $C(x_v^f, T(x_v^f), c^v)$, entre todas as soluções possíveis.

$$\text{Maximizar} \quad \sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}}(x_v^f \times A(x_v^f)) \quad - \quad \frac{\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}} x_v^f \times T(x_v^f)\times c^v}{C(x_v^f, T(x_v^f), c^v)} \tag{4-1}$$

$$C(x_v^f, T(x_v^f), c^v) = Max\Big(\sum_{f\in\mathcal{F}}\sum_{v\in\mathcal{V}} x_v^f \times T(x_v^f)\times c^v\Big) \tag{4-2}$$

$$\text{Sujeito a:} \quad \text{3-2-3-9} \tag{4-3}$$

As demais restrições do modelo permanecem inalteradas nessa reformulação.

4.2 Solução em um estágio

Na reformulação para um único estágio, também eliminamos o uso dos Algoritmos 3.1 e 3.2 no cálculo do consumo de recursos da configuração f ao ser enviada à máquina virtual v. Nessa reformulação, passamos a considerar a variável de decisão x_v^t, que indica a alocação da tarefa t à máquina virtual v.

As Equações 4-4 e 4-5 representam a função objetivo do modelo, sendo que a primeira parte visa maximizar a quantidade de tarefas processadas com sucesso, enquanto a segunda parte tem como objetivo minimizar o custo de utilização da infraestrutura, semelhante à função objetivo proposta na Seção 4.1.

Semelhante às restrições do modelo em três estágios e à reformulação em dois estágios, as Equações 4-6-4-8 têm como objetivo impor limites ao uso de recursos. Porém, nesse caso, consideramos as restrições apenas de memória RAM, tempo de serviço da tarefa, bem como a restrição que estabelece que a tarefa deve ser alocada em, no máximo, uma máquina virtual v (Equação 4-8).

No modelo original, o consumo de CPU era considerado um parâmetro que limitava a sobrecarga na máquina virtual. Essa sobrecarga poderia gerar problemas relacionados ao próprio tempo de processamento de uma tarefa, aumentando o número de falhas. Da mesma forma, o modelo original e a reformulação apresentada na Seção 4.1 consideravam um limite de utilização dos enlaces WLAN, WAN e MAN. Esses limites também buscavam prevenir uma latência alta, influenciando nos requisitos de QoS da aplicação. Porém, a solução original e todas as reformulações já consideram como restrição o tempo de serviço (Equação 4-6). Logo, consideramos que apenas a restrição de tempo de serviço é suficiente para garantir os requisitos de QoS da aplicação.

$$\text{Maximizar} \quad \sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}} x_v^t \quad - \quad \frac{\sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}} x_v^t \times T(x_v^t)\times c^v}{C(x_v^t, T(x_v^t), c^v)} \tag{4-4}$$

$$C(x_v^t, T(x_v^t), c^v) = Max\Big(\sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}} x_v^t \times T(x_v^t)\times c^v\Big) \tag{4-5}$$

$$\text{Sujeito a:} \quad \sum_{t\in\mathcal{T}}\sum_{v\in\mathcal{V}} (x_v^t \times \tau_v^t) \leq T_v, \tag{4-6}$$

$$(\Gamma_{com}^{t,u,v} + \Gamma_{proc}^{t,u,v} + \Gamma_{esp}^{t,u}) \times x_v^t \leq \Delta_u^t, \;\; \forall t \in \mathcal{T}, u \in \mathcal{U}, v \in \mathcal{V}, \tag{4-7}$$

$$\sum_{v\in\mathcal{V}} x_v^t \leq 1, \forall t \in \mathcal{T}. \tag{4-8}$$

O tempo de envio dos dados de uma tarefa t para a máquina virtual v e o retorno dos resultados ao dispositivo do usuário é representado por:

$$\Gamma_{com}^{t,u,v} = \frac{\alpha_u^t + \beta_u^t}{B_{wlan}^{u,a(t)}} \times x_v^t \times W(t,u,a), \tag{4-9}$$

$$\frac{\alpha_u^t + \beta_u^t}{B_{wan}^{u,a(t)}} \times x_v^t \times W(t,u,a) + \Gamma_{wan}^{prop}, \tag{4-10}$$

$$\frac{\alpha_u^t + \beta_u^t}{B_{man}^t} \times x_v^t \times W(t,u,a) + \Gamma_{man}^{prop}, \tag{4-11}$$

onde a função $W(t,u,a) = 1$ caso a tarefa t, gerada pelo usuário u, tenha sido originada no ponto de acesso a e $W(t,u,a) = 0$ caso contrário.

A largura de banda disponível para cada usuário ($B_{wlan}^{u,a(t)}$, $B_{wan}^{u,a(t)}$) e cada tarefa (B_{man}^t) é calculada por:

$$B_{wlan}^{u,a(t)} = \frac{B_{wlan}^{a(t)}}{\sum_{t \in \mathcal{T}} \sum_{v \in \mathcal{E}} x_v^t \times W(t,u,a)}, \tag{4-12}$$

$$B_{wan}^{u,a(t)} = \frac{B_{wan}^{a(t)}}{\sum_{t \in \mathcal{T}} \sum_{v \in \mathcal{C}} x_v^t \times W(t,u,a)}, \tag{4-13}$$

$$B_{man}^t = \frac{B_{man}}{\sum_{t \in \mathcal{T}} \sum_{v \in \{\mathcal{E} \neq \theta(t)\}} x_v^t}, \tag{4-14}$$

onde os parâmetros $B_{wlan}^{a(t)}$, $B_{wan}^{u,a(t)}$ e B_{man} representam a largura de banda total disponível no ponto de acesso a para os respectivos enlaces.

A latência de processamento pode ser definida por:

$$\Gamma_{proc}^{t,u,v} = \frac{\omega_u^t}{P(x_v^t, \mathcal{T}_{legado}, \phi_u^t, \Phi^v)} \times x_v^t, \tag{4-15}$$

onde ω_u^t representa a quantidade de instruções a serem processadas pela tarefa t, gerada pelo usuário u, e $P(x_v^t, \mathcal{T}_{legado}, \phi_u^t, \Phi^v)$ denota a capacidade de processamento da máquina virtual v, expressa em quantidade de instruções por segundo.

O cálculo de $P(x_v^t, \mathcal{T}_{legado}, \phi_u^t, \Phi^v)$ pode ser definido como:

$$P(x_v^t, \mathcal{T}_{legado}, \phi_u^t, \Phi^v) = \frac{\Phi^v / Max(\sum_{t \in \mathcal{T}} x_v^t + |\mathcal{T}_{legado}|, 1)}{Max(\phi_u^t,\ \Phi^v / Max(\sum_{t \in \mathcal{T}} x_v^t + |\mathcal{T}_{legado}|, 1))}, \tag{4-16}$$

onde no numerador, temos o tempo total de processamento da máquina virtual v, dividido igualmente entre todas as tarefas alocadas. No denominador, é utilizada uma função Max que retorna o maior valor entre a quantidade de tempo de processamento

requisitada pela tarefa t (em núcleos), representada por ϕ_u^t, e a quantidade efetivamente recebida pela tarefa. Se ϕ_u^t for maior que o tempo disponível, isso indica que a tarefa não terá acesso ao tempo de processamento ideal requisitado, recebendo apenas uma fração. Caso contrário, ela receberá exatamente a quantidade solicitada.

4.3 Análise de complexidade

Para a reformulação proposta na Seção 4.1, embora a unificação dos estágios 2 e 3 tenha simplificado de maneira considerável o modelo ao tratar o problema de orquestração em apenas dois estágios, tal simplificação não tem impacto real na sua complexidade.

Ou seja, ainda permanece a complexidade do estágio 1 como $O(|\mathcal{V}||\mathcal{F}|O(A1))$. Da mesma forma, a unificação dos estágios 2 e 3 em um único modelo resulta em um problema de programação inteira (*Integer Linear Programming* (ILP)) com as mesmas características que permitem a sua redução para o problema APSCAQ, e que, portanto, possui complexidade NP-Completo [28, 23].

Por fim, na reformulação apresentada na Seção 4.2, modificamos a variável de decisão, o que altera a estrutura do problema em relação ao APSCAQ. Para analisar sua complexidade, demonstraremos que é possível realizar uma redução em tempo polinomial para o problema de atribuição generalizado (*Generalized Assignment Problem* (GAP)) [28].

GAP: Seja $\mathcal{M}$ o conjunto de tarefas a serem atribuídas a um conjunto de agentes $\mathcal{N}$, onde c_{ij} representa o custo associado a atribuição da tarefa i ao agente j, a_{ij} a quantidade de recursos consumidos pelo agente j ao receber a tarefa i, e b_j a capacidade total do agente j. O objetivo é realizar a atribuição de tarefas aos agentes de forma a maximizar o custo agregado. Diferentemente do APSCAQ, o GAP permite que um mesmo agente seja designado para múltiplas tarefas. A formulação geral do GAP é dada por:

$$\text{Maximizar} \quad \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{N}} c_{ij}x_{ij} \tag{4-17}$$

$$\text{Sujeito a:} \quad \sum_{i\in\mathcal{M}} a_{ij}x_{ij} \leq b_j \quad , \quad j \in \mathcal{N}, \tag{4-18}$$

$$\sum_{j\in\mathcal{N}} x_{ij} \leq 1 \quad , \quad i \in \mathcal{M}, \tag{4-19}$$

$$x_{ij} \in \{0,1\}. \tag{4-20}$$

Resolver o GAP é uma tarefa de complexidade NP-difícil [28, 7]. Assim, buscamos demonstrar que a reformulação apresentada também é NP-difícil, por meio de uma redução em tempo polinomial do GAP para o problema definido na Seção 4.2.

Redução: Assim como no GAP, a reformulação apresentada também permite a alocação de múltiplas tarefas a um mesmo agente (máquina virtual), ao mesmo tempo em que impõe a restrição de que cada tarefa seja atribuída a, no máximo, um agente (Equações 4-19 e 4-8).

A restrição de capacidade dos agentes, expressa na Equação 4-18, encontra correspondência nas restrições 4-6 e 4-7 da reformulação, as quais impõem limites à alocação de tarefas com base na capacidade de memória e nos limites de tempo de serviço. De maneira análoga, o custo agregado c_{ij} presente na Equação 4-17 do GAP está refletido na função objetivo da Equação 4-4, na qual se busca maximizar o atendimento das tarefas e, simultaneamente, minimizar os custos de uso da infraestrutura.

Concluímos, portanto, que uma instância do GAP pode ser transformada, em tempo polinomial, em uma instância do problema abordado na reformulação proposta na Seção 4.2. Assim, demonstra-se que o problema em questão é NP-difícil.

4.4 Avaliação

Nesta avaliação, foi considerado o *cenário principal*, definido no Capítulo 3, com algumas modificações: a execução do modelo foi limitada a 1,1 segundo (0,1 segundo de *warmup*), o tamanho máximo do lote foi ampliado para 10 tarefas, e os experimentos foram realizados com 200, 2400 e 4600 usuários. Embora o tempo do experimento tenha sido diminuído, a alteração da quantidade máxima de tarefas por lote e o aumento da quantidade de usuários permitem explorar aspectos do tempo de execução das soluções, que é o objetivo desta seção.

A Figura 4.1 apresenta uma comparação entre as três abordagens (três estágios, dois estágios e um estágio) em relação à quantidade de tarefas concluídas com sucesso. Embora o modelo com apenas um estágio não atinja a mesma eficiência das demais abordagens, ele se aproxima consideravelmente das soluções mais complexas, com uma diferença máxima de aproximadamente 3% se comparado à abordagem em três estágios. No entanto, os impactos dessa simplificação foram relativamente pequenos quando comparados aos benefícios em tempo de execução, conforme discutido a seguir.

A Figura 4.2 apresenta o tempo de execução das três abordagens. Observa-se que, na abordagem em dois estágios, a remoção de um dos estágios de otimização resultou em uma redução proporcional no tempo total de execução de um dos estágios. De forma similar, a eliminação dos Algoritmos 3.1 e 3.2 na abordagem em 1 estágio contribuiu

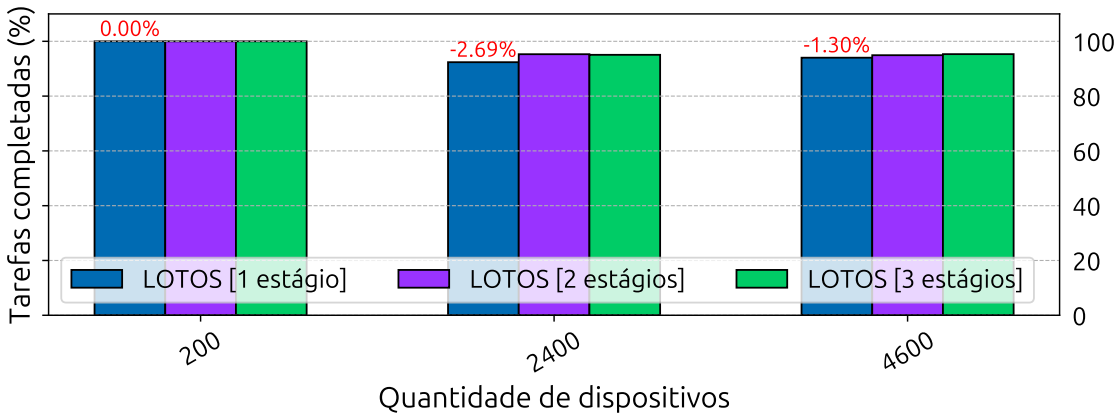


Figura 4.1: *Eficiência em termos de tarefa completadas comparando modelos em um, dois e três estágios. O valor em vermelho indica a diferença de taxas de execução de tarefas entre as abordagens em um e três estágios.*

significativamente para a eficiência computacional, permitindo uma melhoria no tempo de execução de aproximadamente 1400 segundos em relação à abordagem em 3 estágios.

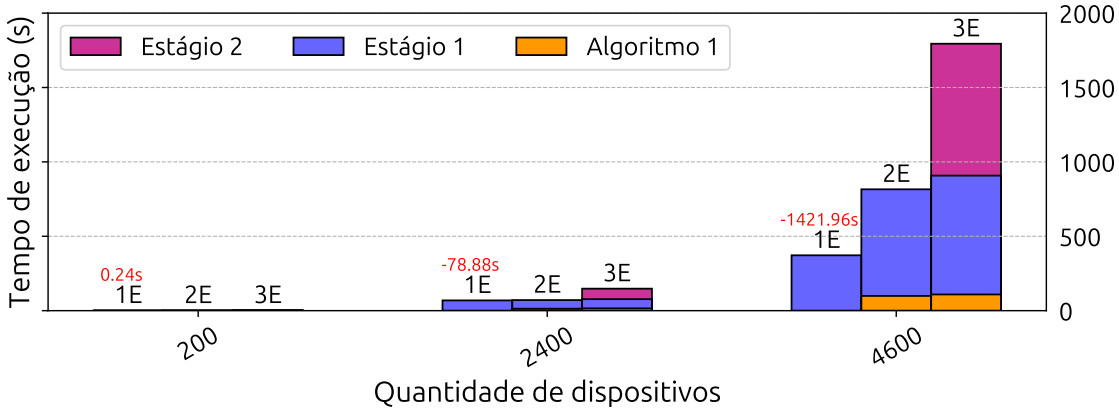


Figura 4.2: *Tempo de execução dos modelos em um, dois e três estágios. Os símbolos 1E, 2E, 3E significam respectivamente, 1 estágio, 2 estágios e 3 estágios. O valor em vermelho indica a diferença de tempos de execução entre as abordagens em um e três estágios.*

4.5 Considerações finais do capítulo

Neste capítulo, foram propostas reformulações no modelo original de orquestração em três estágios, com o objetivo de aumentar a eficiência. A primeira reformulação abordou o problema em dois estágios, unificando os modelos de otimização e priorizando o atendimento à demanda dos usuários, mas ainda preservando o objetivo de minimizar os custos. A segunda reformulação, por sua vez, tratou a orquestração por meio de um único estágio de otimização, simplificando o processo decisório sem comprometer significativamente os objetivos do modelo original.

CAPÍTULO 5

Abordagem utilizando heurística

Neste capítulo, discutimos as limitações apresentadas pelo *LOTOS* bem como as suas reformulações, em termos de tempo de execução. Além disso, será apresentada uma solução heurística desenvolvida com o objetivo de otimizar o processo de alocação de recursos, buscando torná-lo mais eficiente.

5.1 LATOS

Embora as remodelagens apresentadas no Capítulo 4 tenham possibilitado a geração de soluções mais otimizadas em comparação à solução original discutida no Capítulo 3, a complexidade de ambas as abordagens, com uso de programação matemática, ainda limita a eficiência dos modelos. Diante desse cenário, propomos uma solução denominada *Local Approximated Task Orchestration Solution (LATOS)*, com o objetivo de gerar aproximações eficientes para o problema de alocação de recursos e orquestração de tarefas abordado nas seções anteriores.

A solução apresentada caracteriza-se como uma heurística construtiva gananciosa, na qual a primeira máquina virtual com recursos disponíveis é selecionada para receber a carga de trabalho de uma tarefa a ser processada. O Algoritmo 5.1 descreve a execução completa da *LATOS*.

Inicialmente, na linha 1, as tarefas são ordenadas com base no tempo restante para que cada uma atinja o limite de latência definido pelo tipo de aplicação que a gerou. Esse tempo restante é calculado como $\Delta_u^t - \Gamma_{esp}^{t,u}$, ou seja, o limite de latência estabelecido pela aplicação subtraído do tempo que a tarefa aguardou até que um lote viável fosse formado e enviado ao orquestrador.

Em seguida, na linha 2, as máquinas virtuais são ordenadas em ordem crescente de custo de utilização. Dessa forma, o algoritmo prioriza, ao longo das iterações e da geração de soluções, tanto as tarefas com menos tempo restante, quanto as máquinas virtuais mais econômicas, promovendo uma alocação eficiente dos recursos.

Em cada iteração, conforme indicado na linha 4, uma tarefa é selecionada, e iteramos sobre, no máximo, todas as máquinas virtuais disponíveis, com o objetivo de escolher a primeira que esteja apta a processá-la. Para cada máquina virtual considerada, realiza-se a verificação de sua capacidade para receber a carga de trabalho da tarefa t.

Essa verificação ocorre nas linhas 5 a 6, por meio das funções Verificar_RAM(t, v) e Verificar_Tempo_de_Serviço(t, v). Cada uma dessas funções retorna TRUE caso a máquina virtual v atenda ao critério especificado, e FALSE caso contrário.

Os critérios avaliados por essas funções são os seguintes:

- (c1) Verificação de disponibilidade de memória RAM;
- (c2) Verificação se o tempo total para transferência e processamento da tarefa não ultrapassa o limite estabelecido pela aplicação.

Na linha 7, é feita uma checagem global para garantir que todos os critérios acima foram atendidos. Caso verdadeiro, a tarefa é associada à máquina virtual correspondente. Em seguida, a iteração interna (linha 5) é interrompida (linha 9), permitindo que o algoritmo prossiga para a orquestração da próxima tarefa na fila.

Algoritmo 5.1: LATOS

Entrada: $\mathcal{T}, \mathcal{V}$
Saída: Associação de tarefas e VMs

```
1  T_ordenado ← Ordenar(T, Γ^{t,u}_esp − Δ^t_u);
2  V_ordenado ← Ordenar(V, c^v);
3  para t ∈ T_ordenado faça
4      para v ∈ V_ordenado faça
5          c1 ←Verificar_RAM(t, v);
6          c2 ←Verificar_Tempo_de_Serviço(t, v);
7          se c1 ∧ c2 então
8              CriarAssociação(t, v);
9              pare;
10         fim
11     fim
12 fim
```

5.2 Análise de complexidade

Para a análise de complexidade no pior caso, podemos assumir que, para cada tarefa, será necessário percorrer todas as máquinas virtuais até encontrar uma que possua recursos disponíveis. Dessa forma, o tempo de execução no pior cenário pode ser expresso como: $O(\mathcal{T} \times \mathcal{V} \times O(c1) \times O(c2))$.

No entanto, como todas as verificações dos critérios ($O(c1)$ e $O(c2)$) são operações de tempo constante, a complexidade pode ser simplificada para: $O(\mathcal{T} \times \mathcal{V})$. Portanto, a complexidade geral do algoritmo cresce proporcionalmente em relação ao número de tarefas e ao número de máquinas virtuais disponíveis, permitindo escalabilidade para cenários com grande volume de tarefas.

5.3 Avaliação

Para a avaliação do desempenho do *LATOS*, utilizamos os mesmos cenários descritos no Capítulo 4. A eficiência do modelo foi analisada em comparação com a solução *LOTOS*, considerando suas versões com três, dois e um estágio. A Figura 5.1 apresenta uma comparação baseada na porcentagem de tarefas executadas com sucesso.

A partir do gráfico, observa-se que o *LATOS* alcança uma taxa de aceitação de tarefas bastante próxima à obtida pelas diferentes versões do *LOTOS*, com destaque especial para a versão de três estágios, onde a heurística apresentou uma diferença de aproximadamente 7%. Embora a abordagem adotada pelo *LATOS* seja baseada em uma heurística gananciosa, a priorização na alocação de tarefas mais críticas permite à solução manter uma taxa elevada de tarefas executadas sem erros, demonstrando sua viabilidade e competitividade frente às abordagens mais complexas do *LOTOS*.

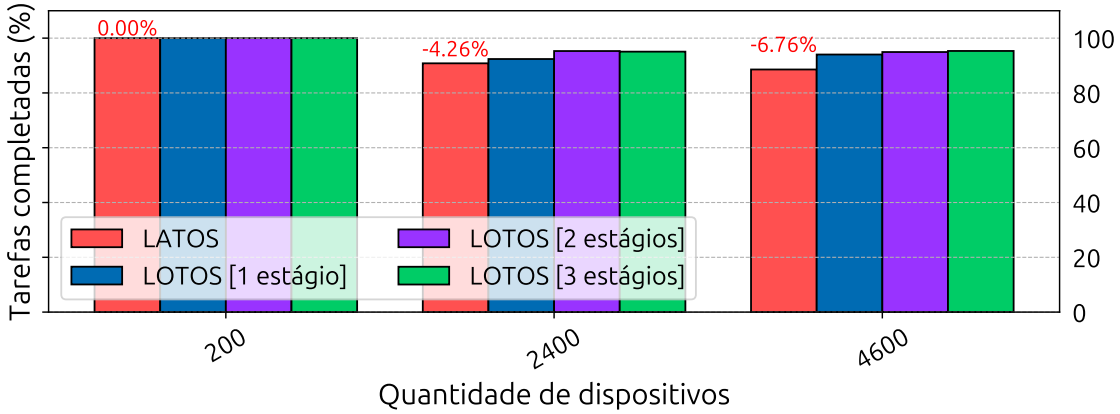


Figura 5.1: *Eficiência em termos de tarefas completadas comparando o LATOS, com modelos em um, dois e três estágios. O valor em vermelho indica a diferença de taxas de execução de tarefas entre as abordagens LATOS e LOTOS três estágios.*

O tempo de execução do *LATOS*, em comparação com as três versões do LOTOS, está ilustrado na Figura 5.2. Em todas as instâncias avaliadas, o LATOS foi capaz de gerar soluções em menos de 1 segundo.

Na instância com 2400 dispositivos, a diferença de desempenho chegou a aproximadamente 1500 vezes em relação à versão mais eficiente do *LOTOS*, e 3500 vezes em comparação à versão menos eficiente.

Na última instância analisada, o *LATOS* demonstrou um desempenho de aproximadamente 9000 vezes superior em relação à versão mais eficiente do LOTOS e aproximadamente 44000 vezes superior quando comparado à versão menos eficiente. Esses resultados evidenciam a vantagem significativa do *LATOS* em termos de tempo de execução, especialmente em cenários com alta carga computacional.

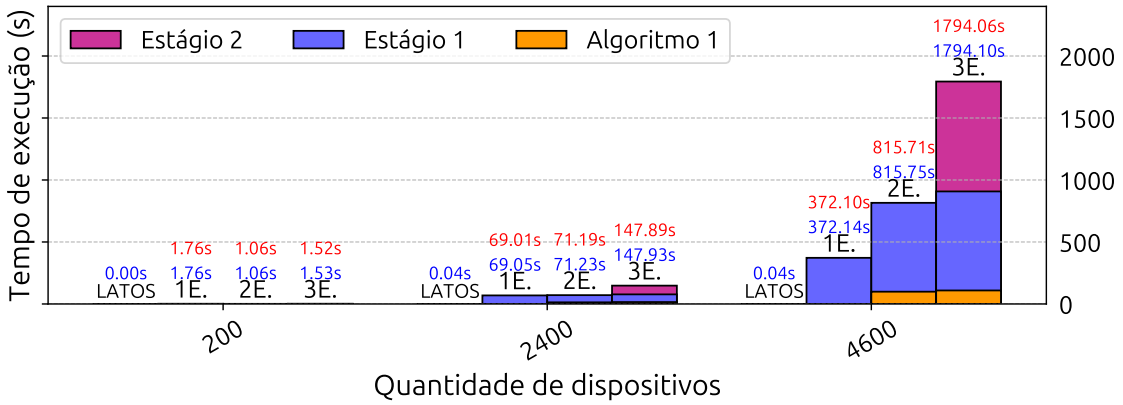


Figura 5.2: *Tempo de execução do LATOS. O valor em azul representa o tempo de execução da solução. Os valores em vermelho indicam a diferença dos tempos de execução em relação ao LATOS.*

A Figura 5.4 apresenta um *boxplot* com os tempos de execução de todas as execuções das soluções para cada instância. Embora as três versões do *LOTOS* apresentem tempos de execução bastante distintos entre si, todas apresentam *outliers* com tempos superiores a 10 segundos, especialmente na instância com o maior número de dispositivos.

Por outro lado, o *LATOS* demonstra um comportamento significativamente mais eficiente. Em todas as execuções, independentemente do tamanho do lote processado, o tempo de resposta permanece abaixo de 0,1 segundo. Além disso, para a grande maioria das chamadas ao orquestrador, o tempo permanece inferior a 0,001 segundo. Esses resultados reforçam a eficiência computacional da abordagem heurística proposta.

Por fim, na Figura 5.4, avaliamos o tempo médio de execução considerando todas as chamadas ao orquestrador com o mesmo tamanho de lote. Podemos observar que, nas três versões do *LOTOS*, o tempo de execução aumenta de forma significativa à medida que o tamanho do lote cresce, refletindo a maior complexidade computacional envolvida na resolução do problema por meio de programação matemática (considerando a complexidade do problema).

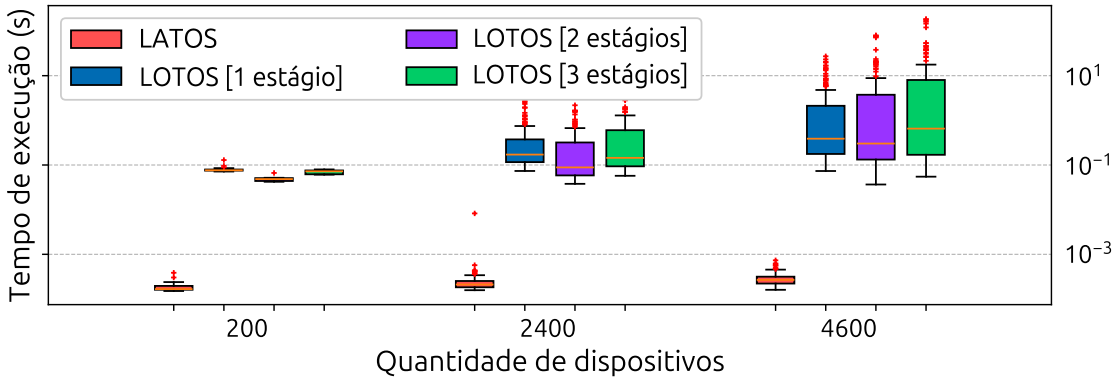


Figura 5.3: *Boxplot do tempo de execução.*

Em contraste, o *LATOS* mantém um tempo de execução consistente, independentemente do tamanho do lote, resultado diretamente associado à menor complexidade da abordagem heurística empregada.

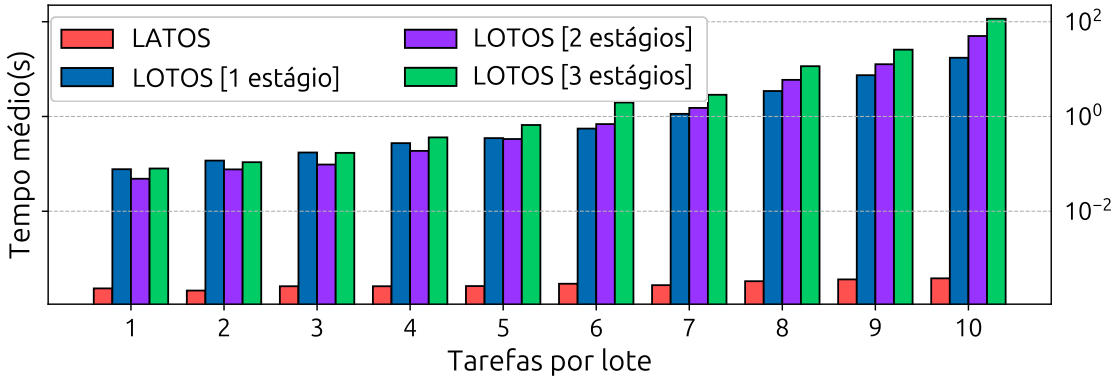


Figura 5.4: *Tempo de execução médio por tamanho do lote.*

5.4 Considerações finais do capítulo

Neste capítulo, foi proposta uma heurística construtiva gananciosa como alternativa às soluções de orquestração de tarefas baseadas em programação matemática. A heurística foi avaliada em comparação com todas as versões do *LOTOS*, demonstrando eficiência computacional sem causar degradações significativas na qualidade das soluções obtidas.

CAPÍTULO 6

Abordagem proativa

Neste capítulo, é apresentada uma abordagem proativa para a alocação de recursos, com o suporte do modelo ARIMA. Realizamos a integração do ARIMA tanto na abordagem baseada em programação matemática (*LOTOS*) quanto na heurística proposta (*LATOS*). Por fim, avaliamos as estratégias proativas em contraste com suas versões originais.

6.1 Motivação

No contexto dos modelos propostos para alocação de recursos (*LOTOS* e *LATOS*), o tamanho do lote representa um elemento fundamental na qualidade das soluções geradas. A definição desse parâmetro é, inclusive, o principal motivo pelo qual as soluções apresentadas são consideradas como localmente exatas, já que os modelos são capazes de produzir soluções exatas apenas para um subconjunto do problema completo.

Esse cenário se justifica pelo fato de que, ao considerarmos o fluxo contínuo de tarefas originadas por inúmeros dispositivos, torna-se inviável a geração de uma solução exata de forma global. Isso porque o número de tarefas acumuladas em um único lote tenderia ao infinito. Além disso, o tempo de espera de cada tarefa até a formação completa do lote também cresceria indefinidamente, resultando em latências elevadas ou até mesmo infinitas para todas as tarefas, o que comprometeria os requisitos de desempenho do sistema.

De todo modo, ao considerarmos em nossa modelagem que as soluções geradas são exatas, porém restritas a um escopo local, algumas informações relevantes para os modelos de decisão acabam sendo desconsideradas. Um exemplo dessa limitação é ilustrado na Figura 6.1.

Nesse cenário, consideramos que a tarefa 228 chega ao orquestrador no instante de tempo 918. O orquestrador então decide alocar essa tarefa na máquina virtual 4, de modo que, na estampa de tempo 923, a tarefa chega à máquina virtual e inicia seu processamento. No momento da decisão, o orquestrador assume que a tarefa 228 não irá

concorrer por recursos com nenhuma outra tarefa, o que leva à estimativa de tempo de processamento compatível com os requisitos definidos para a tarefa.

Na estampa de tempo 921, a tarefa 230 é enviada ao orquestrador por meio do lote seguinte. Novamente, o orquestrador decide alocá-la na máquina virtual 4, uma vez que, no momento da decisão, todas as restrições de utilização de recursos são atendidas. Ao realizar essa alocação, o orquestrador assume que a tarefa 230 não enfrentará concorrência por recursos, já que a tarefa 228, previamente alocada para a mesma máquina virtual, ainda não chegou ao destino (sua chegada está prevista apenas para t.923).

No entanto, quando a tarefa 228 de fato chega à máquina virtual 4 em t.923, ambas passam a competir pelos mesmos recursos. Esse conflito ocorre porque a informação utilizada tanto pelo *LOTOS* quanto pelo *LATOS* é estritamente local, sem considerar interações futuras entre alocações feitas em diferentes momentos.

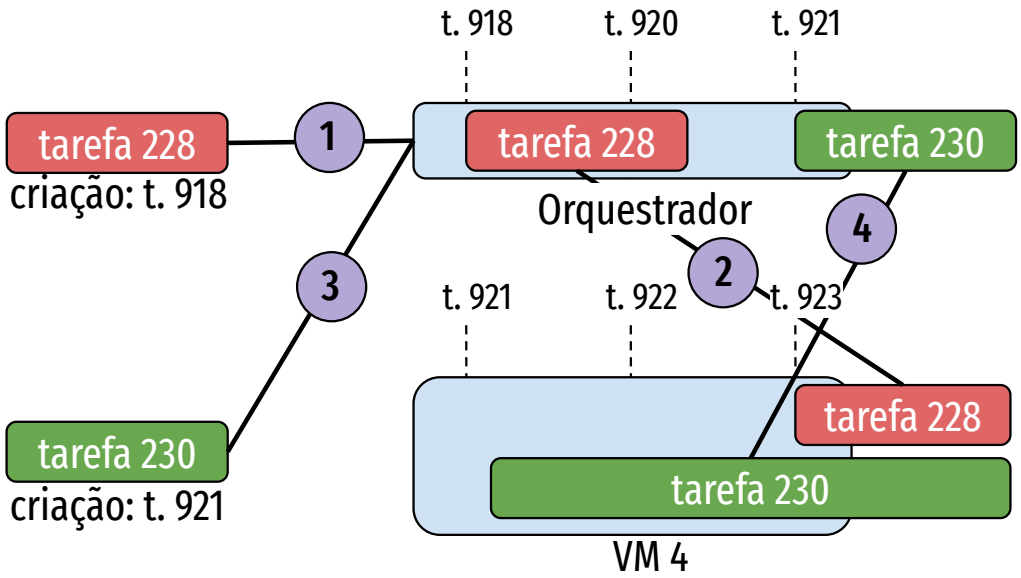


Figura 6.1: *Assincronia de informação ao orquestrar tarefas de lotes diferentes.*

Considerando o cenário descrito, é possível inferir que há benefícios significativos em tratar o problema de forma proativa. Nesse contexto, determinadas informações do ambiente podem ser antecipadas por meio de modelos de predição, o que permite que os modelos de decisão tenham uma visão mais precisa e abrangente do sistema. Essa antecipação contribui para uma alocação mais eficiente de recursos e para uma orquestração de tarefas mais eficaz, reduzindo conflitos e melhorando o desempenho geral da solução.

6.2 Solução proativa baseada em ARIMA

Neste trabalho, adotamos o modelo ARIMA como base para a implementação de uma abordagem proativa.

A Figura 6.2 ilustra o fluxo de execução da abordagem proativa proposta. Inicialmente, o orquestrador recebe os dados de entrada referentes a um conjunto de tarefas agrupadas em um lote. Com base nessas informações, a quantidade de tarefas alocadas em cada máquina virtual é registrada em um histórico temporal, que será utilizado como entrada para o modelo ARIMA. A escolha por monitorar a quantidade de tarefas em cada máquina virtual justifica-se pelo fato de que essa informação é essencial para estimar o tempo de processamento, o qual impacta diretamente na latência total percebida pela tarefa.

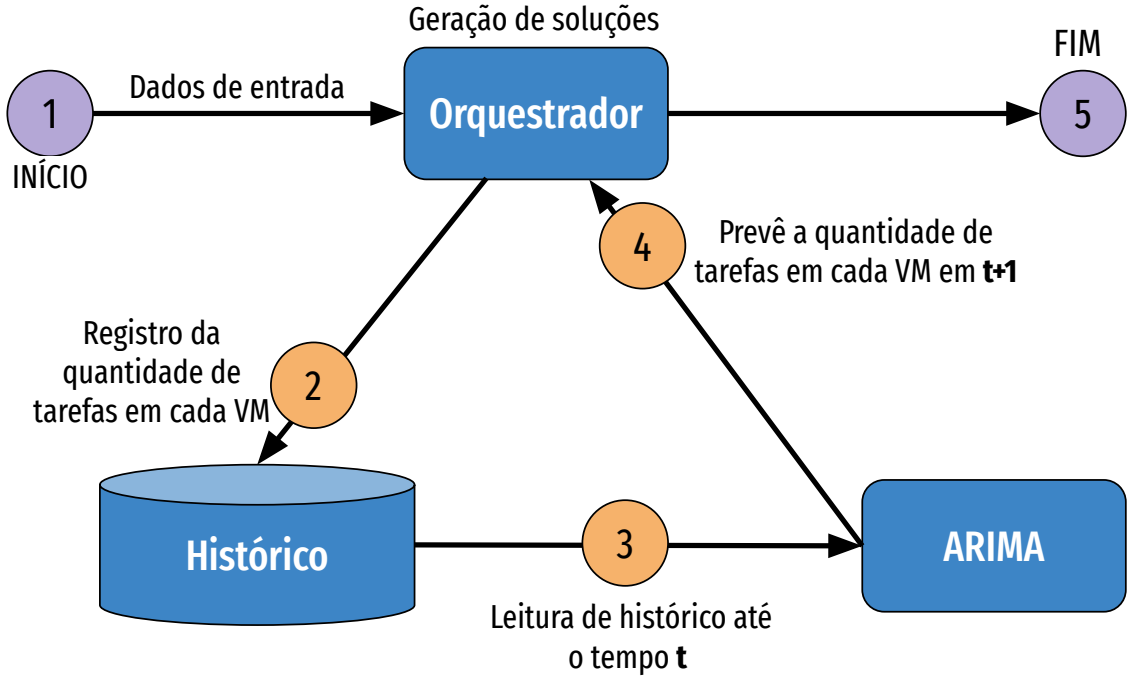


Figura 6.2: *Fluxo de execução da abordagem proativa.*

Após a leitura dos dados históricos, o modelo ARIMA realiza a predição da quantidade de tarefas que estarão alocadas em cada máquina virtual no instante de tempo $t+1$, ou seja, na próxima chamada ao orquestrador. Dessa forma, o orquestrador passa a dispor de informações tanto sobre o estado atual (t) quanto sobre uma estimativa do estado futuro ($t+1$) de ocupação de cada máquina virtual.

Para fins de alocação, consideramos o maior valor entre esses dois instantes, assumindo que ele representa o cenário de maior concorrência por recursos em um futuro próximo. Com base nessas informações, tanto as observadas quanto as previstas, o orquestrador procede à geração das soluções de alocação e orquestração de tarefas.

6.3 Avaliação

Para a avaliação do modelo proposto, foi considerado o mesmo cenário experimental descrito no Capítulo 4. A abordagem proativa baseada em predição foi, então, aplicada a ambas as soluções de alocação: *LOTOS* (na sua versão com apenas um estágio) e *LATOS*.

A Figura 6.3 apresenta a comparação entre os modelos com e sem a aplicação da abordagem proativa. Os resultados demonstram um ganho máximo de aproximadamente 3% na quantidade de tarefas executadas com sucesso quando se utiliza uma abordagem proativa baseada em ARIMA.

Esse resultado evidencia que, embora uma abordagem localmente exata seja adequada ao contexto do problema, ela ainda carece de informações suficientes para alcançar soluções mais eficientes. A incorporação de previsões permite ao orquestrador antecipar cenários de escassez, resultando em decisões mais assertivas e melhor aproveitamento dos recursos disponíveis.

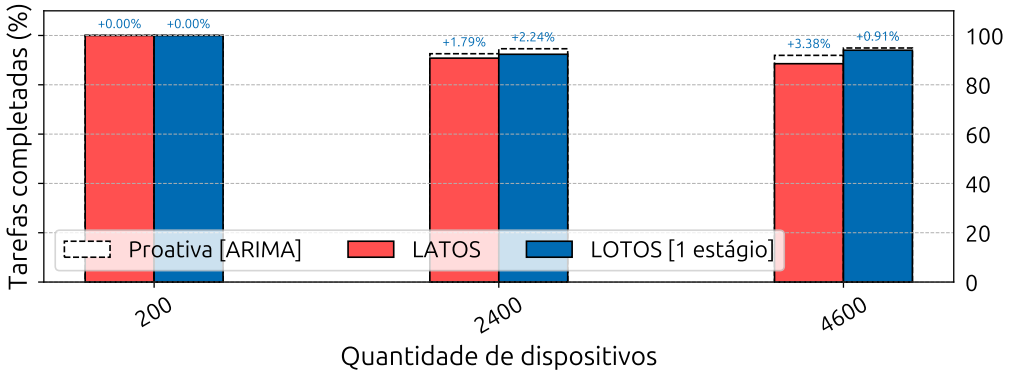


Figura 6.3: *Eficiência da abordagem proativa em comparação ao modelo original. Os valores em azul representam o ganho da abordagem proativa em comparação a solução original.*

As Figuras 6.4 e 6.5 apresentam as curvas correspondentes aos dados históricos temporais e às séries previstas pelo modelo ARIMA. A primeira linha dos gráficos representa a quantidade de tarefas alocadas em máquinas virtuais na nuvem, enquanto as duas últimas linhas correspondem às máquinas virtuais localizadas na borda. Nas Figuras 6.4(a) e 6.5(a), são exibidos os resultados da aplicação da abordagem proativa sobre a heurística *LATOS*, e nas Figuras 6.4(b) e 6.5(b), os resultados referentes à aplicação da mesma abordagem sobre o modelo *LOTOS*. Para avaliar a precisão das previsões realizadas pelo ARIMA, utilizamos o erro quadrático médio (*Root Mean Square Error* (RMSE)) como métrica de desempenho.

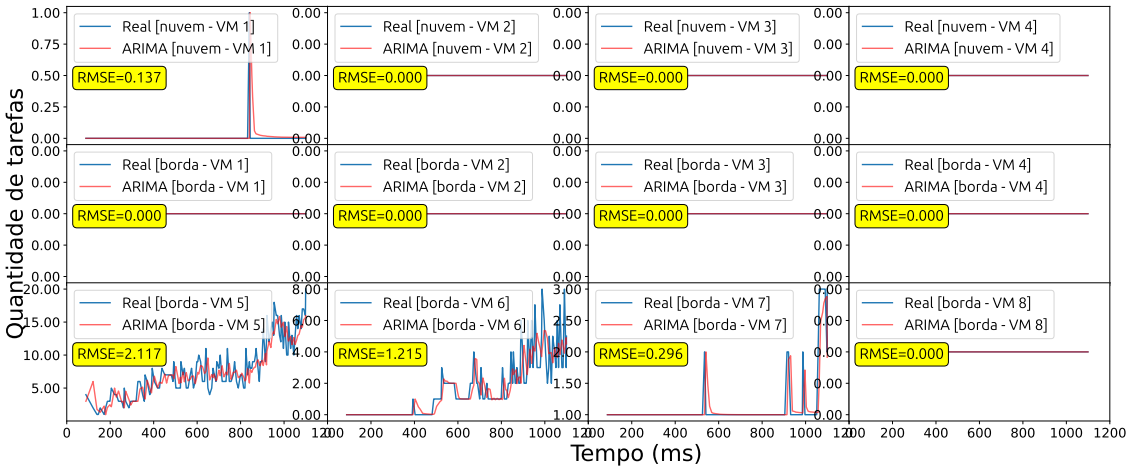
O RMSE é uma métrica amplamente utilizada para avaliar a precisão de modelos de predição, especialmente em contextos de regressão e séries temporais. Essa métrica quantifica a magnitude média dos erros entre os valores previstos pelo modelo e os valores

reais observados. O RMSE é calculado como a raiz quadrada da média dos quadrados das diferenças entre os valores previstos ($\hat{y}_i$) e os valores reais (y_i). Assim, valores mais baixos de RMSE indicam maior aderência entre o modelo preditivo e os dados observados. A expressão matemática do RMSE é definida por:

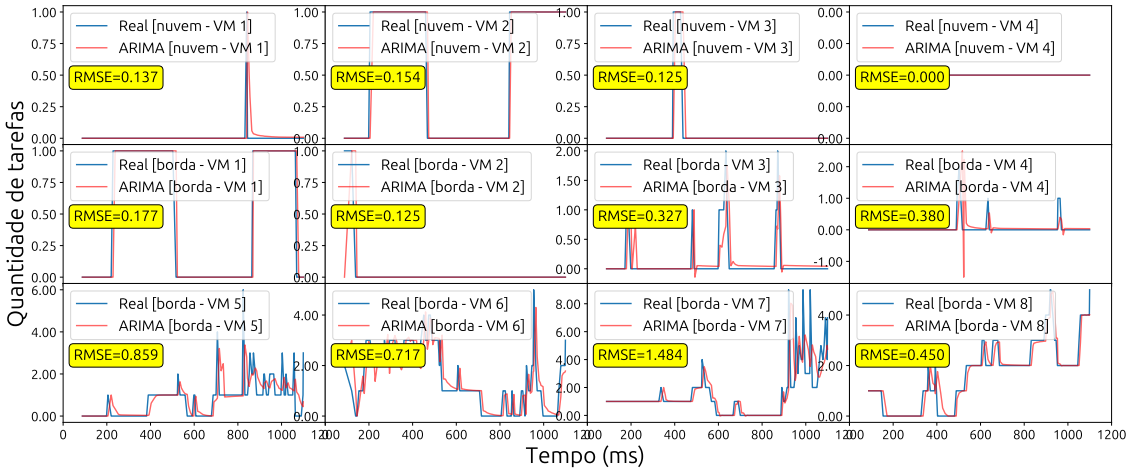
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6-1}$$

onde n representa o número total de observações. Como os erros são elevados ao quadrado, o RMSE penaliza fortemente grandes desvios entre a previsão e o valor real, o que o torna sensível a *outliers*. Em geral, quanto menor o valor do RMSE, melhor o desempenho do modelo.

Observa-se, a partir dos gráficos apresentados nas Figuras 6.4 e 6.5, que o maior valor de RMSE é 2,577, registrado na máquina virtual 6 nas soluções geradas pelo *LATOS* (Figura 6.5(a)). Ainda assim, esse valor representa uma boa acurácia, uma vez que as previsões estão bastante próximas dos dados históricos temporais observados.

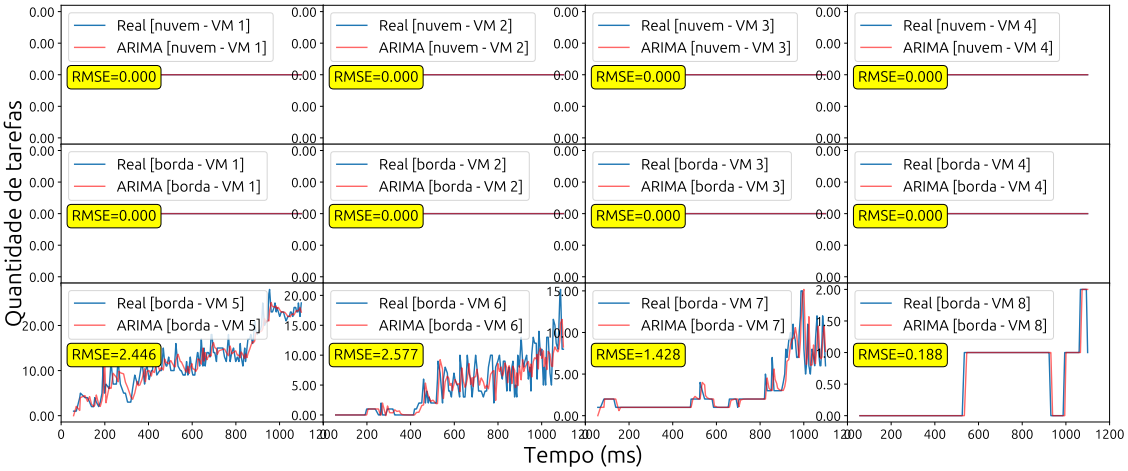


(a) *LATOS*

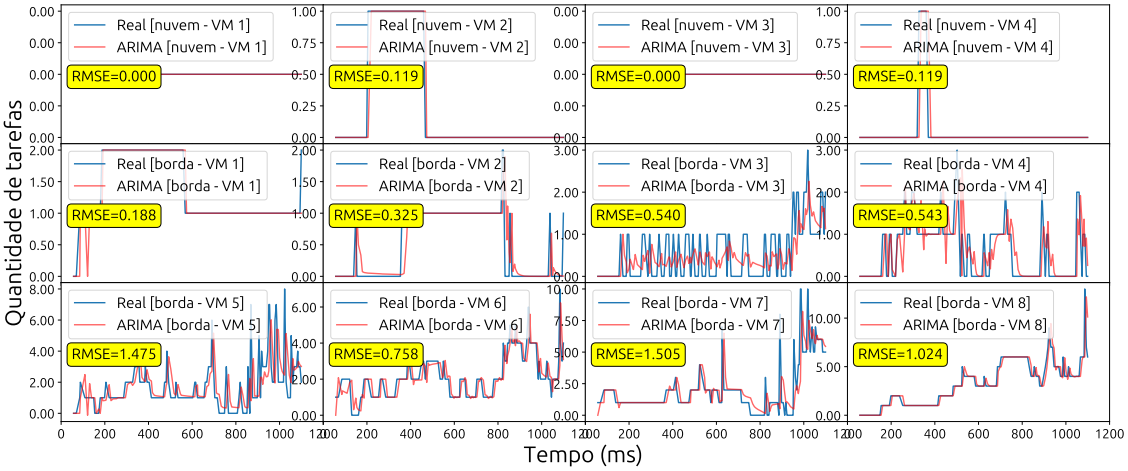


(b) *LOTOS*

Figura 6.4: *Erro Quadrático Médio (RMSE) calculado para predição dos modelos proativos do LATOS e LOTOS. Instância de 2400 dispositivos.*



(a) *LATOS*



(b) *LOTOS*

Figura 6.5: *Erro Quadrático Médio (RMSE) calculado para predição dos modelos proativos do LATOS e LOTOS. Instância de 4600 dispositivos.*

6.4 Considerações finais do capítulo

Neste capítulo, foi proposta uma solução proativa para o problema de orquestração de tarefas, utilizando o modelo preditivo ARIMA para inferir informações relevantes ao orquestrador, com o objetivo de melhorar a qualidade das soluções obtidas. Avaliou-se o uso dessa abordagem proativa junto às soluções *LOTOS* e *LATOS*, sendo possível observar ganhos em relação às abordagens que não incorporam elementos preditivos.

CAPÍTULO 7

Considerações finais

Neste trabalho, foi apresentado um modelo de otimização voltado para a alocação de recursos e a orquestração de tarefas em um ambiente com infraestrutura híbrida de rede e computação, contemplando tanto a borda quanto a nuvem. A proposta se destaca por ser localmente exata, ou seja, a cada instante de tempo, um conjunto de tarefas é orquestrado com o objetivo de maximizar o número de usuários e tarefas atendidos com sucesso. Além disso, o modelo incorpora a minimização dos custos associados à utilização da infraestrutura computacional como um objetivo secundário, assegurando que essa redução de custo não comprometa a qualidade do atendimento à demanda dos usuários.

A solução apresentada enfrenta limitações em termos de escalabilidade, principalmente devido à complexidade do modelo de otimização e do algoritmo utilizado para o cálculo da demanda com base no conjunto de tarefas. Visando mitigar esse problema, foram propostas reformulações da modelagem original, com o objetivo de reduzir o tempo de execução. Além disso, desenvolvemos uma heurística construtiva gananciosa, também voltada à obtenção de maior escalabilidade na resolução das instâncias analisadas.

Também exploramos o uso de abordagens proativas na alocação de recursos, nas quais elementos do ambiente são inferidos por meio de modelos de predição com o objetivo de apoiar a tomada de decisão durante a orquestração de tarefas. Neste trabalho, propomos uma abordagem proativa que prevê a quantidade de tarefas que serão alocadas em uma máquina virtual no futuro, permitindo estimar com maior precisão o tempo de processamento com base na concorrência pelo uso dos núcleos da máquina virtual.

As soluções apresentadas foram avaliadas em comparação com trabalhos da literatura, considerando critérios como o atendimento aos usuários, as características específicas de cada aplicação e os custos associados à infraestrutura de computação. Os resultados demonstraram a superioridade da nossa proposta quando aplicada em um ambiente simulado, garantindo maior taxa de execução de tarefas, menor custo de uso da infraestrutura e menor tempo de serviço.

Por fim, este trabalho evidenciou os benefícios da adoção de abordagens proativas para a alocação de recursos, que se mostraram mais eficazes quando comparadas às soluções originais propostas. Em particular, devido à natureza localmente exata da nossa

abordagem, observou-se que a integração de uma estratégia proativa contribuiu para o aumento da taxa de atendimento das tarefas, elevando ainda mais a eficiência da nossa proposta.

7.1 Contribuições

Considerando todos os pontos já citados, e o objetivo geral de investigar estratégias de orquestração de tarefas de modo a melhorar parâmetros de QoS, otimizar a utilização de recursos e minimizar os custos de uso da infraestrutura, é plausível assumir que o referido trabalho avança o estado da arte, em especial fazendo as seguintes contribuições:

- Modelagem e implementação de solução de orquestração de tarefas localmente exata. Localmente pois consideramos um conjunto de tarefas (que representam a demanda dos usuários) a cada rodada de execução do modelo. Exata pois utilizamos programação matemática para modelar e resolver o problema.
- Consideração de latência, e custo como elementos do modelo de decisão. O primeiro representando uma restrição a ser atendida e o segundo como objetivo a ser otimizado. Embora sejam importantes na tomada de decisão, não foram considerados tais parâmetros nos trabalhos relacionados mapeados.
- Integração de um gerador de carga baseado em uma aplicação real de realidade mista chamada MR-Leo [38].
- Remodelagem do problema proposto com o objetivo de reduzir o tempo de execução da estratégia. Tais remodelagens geram soluções aproximadas ao modelo original em contrapartida ao aumento da eficiência.
- Desenvolvimento de uma heurística construtiva gananciosa que aloca recursos priorizando os serviços com maiores restrições, selecionando servidores na borda ou na nuvem com menor custo de utilização.
- Implementação de um modelo de predição baseado em ARIMA, que utiliza dados temporais para estimar a quantidade de tarefas alocadas em uma VM em um futuro próximo.
- Desenvolvimento de uma abordagem proativa que incorpora o modelo de predição como parâmetro no cálculo do tempo de processamento de uma tarefa, ampliando a quantidade de informações disponíveis para o modelo de alocação e contribuindo para a prevenção de falhas.
- Publicação em repositório aberto de todos os experimentos realizados na pesquisa e apresentados neste trabalho [1].

[1] https://github.com/LABORA-INF-UFG/mestrado-LKL-2025

7.2 Publicações

A partir dos avanços obtidos com essa pesquisa, o seguinte artigo foi aceito para publicação no SBRC no ano de 2025:

- FRAGA, L.; PINTO, L.; CARDOSO, K. **Efficient task orchestration including mixed reality applications in a combined cloud-edge infrastructure**. In: Anais do XLIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, p. 294–307, Porto Alegre, RS, Brasil, 2025. SBC.

Através de colaborações durante o período de mestrado, foram publicados os seguintes artigos, sem relação com a pesquisa realizada no mestrado, onde o referido pesquisador foi coautor nas seguintes publicações:

- ESPER, J. P.; DE S. FRAGA, L.; VIANA, A. C.; CARDOSO, K. V.; CORREA, S. L. **+tour: Recommending personalized itineraries for smart tourism**. Computer Networks, 260:111118, 2025.
- DIAS, G.; FRAGA, L.; CARDOSO, K.; CORREA, S. **Dtmcash: explorando redundância de viewports de usuários para otimização de streaming de vídeo 360º**. In: Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, p. 421–434, Porto Alegre, RS, Brasil, 2024. SBC.
- ROMUALDO, H.; FRAGA, L.; CONCEIÇÃO, P.; ROCHA, F.; CARDOSO, K. **Otimização da associação entre estações base e equipamentos de usuário com auxílio de aprendizado federado para suporte a realidade aumentada móvel**. In: Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, p. 616–629, Porto Alegre, RS, Brasil, 2024. SBC.

Para contribuir com o avanço na pesquisa científica, todos os artigos publicados apresentam repositório público contendo toda a experimentação realizada nos referidos trabalhos. A Tabela 7.1 apresenta o endereço dos repositórios:

Tabela 7.1: *Trabalhos publicados e aceitos.*

Artigo	Repositório
[13]	https://github.com/LABORA-INF-UFG/paper-LLK-2025
[29]	https://github.com/LABORA-INF-UFG/paper-HLPFK-2024
[10]	https://github.com/LABORA-INF-UFG/paper-GLKS-2024
[12]	https://github.com/LABORA-INF-UFG/plusTour

7.3 Trabalhos futuros

Para que esta pesquisa continue avançando e produzindo artefatos relevantes à comunidade acadêmica, são sugeridos os seguintes tópicos para trabalhos futuros:

- Como demonstrado no Capítulo 6, a utilização de uma abordagem proativa permite obter ganhos em problemas de alocação de recursos, já que os modelos preditivos servem como uma base valiosa de informações durante a tomada de decisão. Deste modo, uma investigação mais profunda em relação a utilização de abordagens proativas pode permitir o desenvolvimento de soluções de orquestração mais eficientes. Neste sentido uma das lacunas a serem tratadas em trabalhos futuros se refere a qual elemento do modelo de orquestração melhor se beneficia da utilização de soluções preditivas. Neste trabalho nos consideramos apenas a estimativa da quantidade de tarefas que um nó de processamento poderia ter no futuro. Mas é possível utilizar abordagens preditivas para obter informações relevantes para vários elementos de uma estratégia de orquestração. Além disso, neste trabalho nos utilizamos ARIMA como modelo preditor, embora trabalhos futuros possam se beneficiar através da investigação de outros modelos de aprendizado como *Long Short-Term Memory* (LSTM) ou *Graph Neural Network* (GNN).
- Outro ponto interessante a ser tratado em pesquisas futuras é a utilização de meta-heurísticas como alternativa a soluções baseadas em programação matemática no processo de orquestração de tarefas. Algoritmos genéticos *Genetic Algorithm* (GA), recozimento simulado *Simulated Annealing* (SA), busca adaptativa randomizada gulosa *Greedy Randomized Adaptive Search Procedure* (GRASP) são apenas alguns exemplos de técnicas eficazes na geração de aproximações.

Referências

[1] ABDAH, H.; OTHERS. **Three-Tier Fuzzy-based Orchestration in MEC**. In: *IEEE Global Communications Conference (GLOBECOM)*, p. 1–6, 2021.

[2] ABDULAZEEZ, D. H.; ASKAR, S. K. **A novel offloading mechanism leveraging fuzzy logic and deep reinforcement learning to improve iot application performance in a three-layer architecture within the fog-cloud environment**. *IEEE Access*, 12:39936–39952, 2024.

[3] ALGHAMDI, M.; ALAM, A.; NALLANATHAN, A.; CHERIF, A. **Dynamic clustering-based task orchestrator in mobile edge computing**. In: *2024 International Wireless Communications and Mobile Computing (IWCMC)*, p. 1613–1618, 2024.

[4] ALMUTAIRI, J.; ALDOSSARY, M. **A novel approach for IoT tasks offloading in edge-cloud environments**. *Journal of Cloud Computing*, 10(1):28, Apr. 2021.

[5] AWS. **AWS Calculator.**, 2025. https://calculator.aws/#/createCalculator, 2025-01-08.

[6] BUYYA, R.; OTHERS. **Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities**. In: *2009 International Conference on High Performance Computing & Simulation*, p. 1–11, 2009.

[7] CATTRYSSE, D. G.; VAN WASSENHOVE, L. N. **A survey of algorithms for the generalized assignment problem**. *European Journal of Operational Research*, 60(3):260–272, 1992.

[8] COSTA, B.; BACHIEGA JR, J.; DE CARVALHO, L. R.; ARAUJO, A. P. **Orchestration in fog computing: A comprehensive survey**. *ACM Computing Surveys (CSUR)*, 55(2):1–34, 2022.

[9] DARADKEH, T.; AGARWAL, A.; ZAMAN, M.; S, R. M. **Analytical modeling and prediction of cloud workload**. In: *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, p. 1–6, 2021.

[10] DIAS, G.; FRAGA, L.; CARDOSO, K.; CORREA, S. **Dtmcash: explorando redundância de viewports de usuários para otimização de streaming de vídeo 360º**. In: *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 421–434, Porto Alegre, RS, Brasil, 2024. SBC.

[11] DONG, S.; TANG, J.; ABBAS, K.; HOU, R.; KAMRUZZAMAN, J.; RUTKOWSKI, L.; BUYYA, R. **Task offloading strategies for mobile edge computing: A survey**. *Computer Networks*, 254:110791, 2024.

[12] ESPER, J. P.; DE S. FRAGA, L.; VIANA, A. C.; CARDOSO, K. V.; CORREA, S. L. **+tour: Recommending personalized itineraries for smart tourism**. *Computer Networks*, 260:111118, 2025.

[13] FRAGA, L.; PINTO, L.; CARDOSO, K. **Efficient task orchestration including mixed reality applications in a combined cloud-edge infrastructure**. In: *Anais do XLIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 294–307, Porto Alegre, RS, Brasil, 2025. SBC.

[14] IFTIKHAR, S.; GILL, S. S.; SONG, C.; XU, M.; ASLANPOUR, M. S.; TOOSI, A. N.; DU, J.; WU, H.; GHOSH, S.; CHOWDHURY, D.; OTHERS. **Ai-based fog and edge computing: A systematic review, taxonomy and future directions**. *Internet of Things*, 21:100674, 2023.

[15] ISLAM, A.; DEBNATH, A.; GHOSE, M.; CHAKRABORTY, S. **A survey on task offloading in multi-access edge computing**. *Journal of Systems Architecture*, 118:102225, 2021.

[16] JAMIL, M. N.; OTHERS. **Workload Orchestration in Multi-Access Edge Computing Using Belief Rule-Based Approach**. *IEEE Access*, 11:118002–118023, 2023.

[17] JANARDHANAN, D.; BARRETT, E. **Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models**. In: *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, p. 55–60, 2017.

[18] KALPANA, P.; ALMUSAWI, M.; CHANTI, Y.; SUNIL KUMAR, V.; VARAPRASAD RAO, M. **A deep reinforcement learning-based task offloading framework for edge-cloud computing**. In: *2024 International Conference on Integrated Circuits and Communication Systems (ICICACS)*, p. 1–5, 2024.

[19] KONG, L.; TAN, J.; HUANG, J.; CHEN, G.; WANG, S.; JIN, X.; ZENG, P.; KHAN, M.; DAS, S. K. **Edge-computing-driven internet of things: A survey**. *ACM Comput. Surv.*, 55(8), Dec. 2022.

[20] LI, W.; YANG, G.; WANG, B.; ZHANG, Q.; HU, K.; PAN, C.; NI, Q. **Adaptive two-stage task offloading based on meta reinforcement learning for mobile edge computing**. *The Journal of Supercomputing*, 81(6):1–33, 2025.

[21] LIU, B.; GUO, J.; LI, C.; LUO, Y. **Workload forecasting based elastic resource management in edge cloud**. *Computers & Industrial Engineering*, 139:106136, 2020.

[22] LUO, Q.; HU, S.; LI, C.; LI, G.; SHI, W. **Resource scheduling in edge computing: A survey**. *IEEE Communications Surveys & Tutorials*, 23(4):2131–2165, 2021.

[23] MAZZOLA, J. B.; NEEBE, A. W. **Resource-constrained assignment scheduling**. *Operations Research*, 34(4):560–572, 1986.

[24] MECHALIKH, C.; TAKTAK, H.; MOUSSA, F. **Pureedgesim: A simulation framework for performance evaluation of cloud, edge and mist computing environments**. *Computer Science and Information Systems*, 18(1):43–66, 2021.

[25] MILGRAM, P.; KISHINO, F. **A taxonomy of mixed reality visual displays**. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[26] NOWAK, T. W.; OTHERS. **Verticals in 5G MEC-Use Cases and Security Challenges**. *IEEE Access*, 9:87251–87298, 2021.

[27] PEIXOTO, M. J. P.; AZIM, A. **Design and Development of a Machine Learning-Based Task Orchestrator for Intelligent Systems on Edge Networks**. *IEEE Access*, 11:33049–33060, 2023.

[28] PENTICO, D. W. **Assignment problems: A golden anniversary survey**. *European Journal of Operational Research*, 176(2):774–793, 2007.

[29] ROMUALDO, H.; FRAGA, L.; CONCEIÇÃO, P.; ROCHA, F.; CARDOSO, K. **Otimização da associação entre estações base e equipamentos de usuário com auxílio de aprendizado federado para suporte a realidade aumentada móvel**. In: *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, p. 616–629, Porto Alegre, RS, Brasil, 2024. SBC.

[30] SHARIF, Z.; JUNG, L. T.; RAZZAK, I.; ALAZAB, M. **Adaptive and priority-based resource allocation for efficient resources utilization in mobile-edge computing**. *IEEE Internet of Things Journal*, 10(4):3079–3093, 2023.

[31] SILVA FILHO, M. C.; OLIVEIRA, R. L.; MONTEIRO, C. C.; INÁCIO, P. R. M.; FREIRE, M. M. **Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness**. In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, p. 400–406, 2017.

[32] SONMEZ, C.; OTHERS. **EdgeCloudSim: An environment for performance evaluation of Edge Computing systems**. In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, p. 39–44, 2017.

[33] SONMEZ, C.; OTHERS. **Fuzzy Workload Orchestration for Edge Computing**. *IEEE Transactions on Network and Service Management*, 16(2):769–782, 2019.

[34] SONMEZ, C.; OTHERS. **Machine Learning-Based Workload Orchestrator for Vehicular Edge Computing**. *IEEE Transactions on Intelligent Transportation Systems*, 22(4):2239–2251, 2021.

[35] SPEICHER, M.; HALL, B. D.; NEBELING, M. **What is mixed reality?** In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, p. 1–15, New York, NY, USA, 2019. Association for Computing Machinery.

[36] THEODOROPOULOS, T.; VIOLOS, J.; TSANAKAS, S.; LEIVADEAS, A.; TSEPERS, K.; VARVARIGOU, T. **Intelligent proactive fault tolerance at the edge through resource usage prediction**. *ITU Journal on Future and Evolving Technologies*, 3(3):761–778, 2022.

[37] TOCZÉ, K.; NADJM-TEHRANI, S.; KOBUSINSKA, A. **A taxonomy for management and optimization of multiple resources in edge computing**. *Wirel. Commun. Mob. Comput.*, 2018, Jan. 2018.

[38] TOCZÉ, K.; OTHERS. **Performance Study of Mixed Reality for Edge Computing**. In: *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC'19, p. 285–294, New York, NY, USA, 2019. Association for Computing Machinery.

[39] TOCZÉ, K.; OTHERS. **Characterization and modeling of an edge computing mixed reality workload**. *Journal of Cloud Computing*, 9(1):46, Dec. 2020.

[40] TOCZÉ, K.; NADJM-TEHRANI, S. **ORCH: Distributed Orchestration Framework using Mobile Edge Devices**. In: *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, p. 1–10, 2019.

[41] WANG, H.; OTHERS. **A Survey on the Metaverse: The State-of-the-Art, Technologies, Applications, and Challenges**. *IEEE Internet of Things Journal*, 10(16):14671–14688, 2023.

[42] YAMAK, P. T.; YUJIAN, L.; GADOSEY, P. K. **A comparison between arima, lstm, and gru for time series forecasting**. In: *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI '19, p. 49–55, New York, NY, USA, 2020. Association for Computing Machinery.

[43] YAMANSAVASCILAR, B.; BAKTIR, A. C.; SONMEZ, C.; OZGOVDE, A.; ERSOY, C. **Deepedge: A deep reinforcement learning based task orchestrator for edge computing**. *IEEE Transactions on Network Science and Engineering*, 10(1):538–552, 2023.

[44] ZHENG, T.; OTHERS. **Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing**. *Journal of Cloud Computing*, 11(1):3, Jan. 2022.

APÊNDICE A

Lista de notações e definições

A seguir serão listadas as notações utilizadas neste trabalho e suas definições, separadas pelos capítulos onde foram introduzidas inicialmente.

A.1 Capítulo 3

Conjuntos:

- $\mathcal{U}$: Conjunto de usuários;
- $\mathcal{A}$: Conjunto de pontos de acesso;
- $\mathcal{T}$: Conjunto de tarefas;
- $\mathcal{F}$: Conjunto de configurações. Ou conjunto de combinações de tarefas;
- $\mathcal{V}$: Conjunto de máquinas virtuais;
- $\mathcal{E}$: Conjunto de máquinas virtuais na borda;
- $\mathcal{C}$: Conjunto de máquinas virtuais na nuvem;
- $\mathcal{T}^{v}_{ordenado}$:Conjunto de tarefas ordenadas de acordo com $\delta^{t,u}_{inicial}$;
- $\mathcal{T}^{v}_{concorrente}$: Conjunto de tarefas que competem por recursos de processamento na máquina virtual v;
- $\mathcal{T}^{v}_{legado}$: Conjunto de tarefas legadas na máquina virtual v.

Elementos:

- u: Um usuário em $\mathcal{U}$;
- a: Um ponto de acesso em $\mathcal{A}$;
- t: Uma tarefa em $\mathcal{T}$;
- f: Uma configuração em $\mathcal{F}$. Ou uma combinação de tarefas;
- v: Uma máquina virtual em $\mathcal{V}$.

Variável de decisão

- x^{f}_{v}: Variável de decisão que representa a alocação das tarefas em f na máquina virtual v.

Tarefas:

- ρ_v^f: Utilização da CPU demandada pelas tarefas em f na máquina virtual v;
- τ_v^f: Memória RAM demandada pelas tarefas em f na máquina virtual v;
- ϕ_u^t: Quantidade de núcleos necessários para processar a tarefa t gerada pelo usuário u;
- ω_u^t: Quantidade de instruções da tarefa t gerada pelo usuário u;
- Δ_u^t: Limite de tempo máximo para execução da tarefa t gerada pelo usuário u;
- α_u^t: Quantidade de dados em *upload* da tarefa t gerada pelo usuário u;
- β_u^t: Quantidade de dados em *download* da tarefa t gerada pelo usuário u;
- $\delta_{inicial}^{t,u}$: Estimativa do tempo em que a tarefa t gerada pelo usuário u, irá concluir o *upload* dos dados para a máquina virtual;
- $\delta_{final}^{t,u}$: Tempo de processamento final da tarefa t;
- $\Gamma_{com}^{t,u,v}$: Tempo de *download* e *upload* da tarefa t gerada pelo usuário u na máquina virtual v;
- $\Gamma_{proc}^{t,u,v}$: Tempo de processamento da tarefa t gerada pelo usuário u na máquina virtual v;
- $\Gamma_{esp}^{t,u}$: Tempo que uma tarefa t gerada pelo usuário u precisa esperar até a formação de um lote;
- δ_{atual}: Variável que realiza o registro do tempo atual de acordo com os cálculos realizados no Algoritmo 3.1;
- δ_{dif}: Variável que armazena o período de tempo em que um conjunto de tarefas concorrem por recursos;
- cpt: Fração de uso dos núcleos da máquina virtual v quando as tarefas no conjunto $T_{concorrente}$ estão sendo processadas simultaneamente;
- cap_u^i: Capacidade de processamento em MIPS disponível para a tarefa t gerada pelo usuário u quando as tarefas no conjunto $T_{concorrente}$ estão sendo processadas simultaneamente.

Maquinas virtuais:

- P_v: Porcentagem de CPU residual disponível na máquina virtual v;
- T_v: Memória RAM residual disponível na máquina virtual v;
- Φ^v: Quantidade de núcleos da máquina virtual v;
- Ω^v: Capacidade de processamento em MIPS da máquina virtual v;
- c^v: Custo monetário de utilização da máquina virtual v.

Rede:

- $B_{wlan}^{u,a(t)}$: Largura de banda alocada para cada dispositivo no ponto de acesso a em que a tarefa t foi gerada, do enlace $WLAN$;

- $B_{wan}^{u,a(t)}$: Largura de banda alocada para cada dispositivo no ponto de acesso a em que a tarefa t foi gerada, do enlace *WAN*;
- B_{man}: Largura de banda do enlace MAN;
- ζ_{ul}^{man}: Limite de tarefas que podem ser transmitidas no enlace MAN em *upload*;
- ζ_{dl}^{man}: Limite de tarefas que podem ser transmitidas no enlace MAN em *download*;
- ζ_{a}^{wlan}: Limite residual de usuários que podem utilizar o enlace WLAN no ponto de acesso a;
- ζ_{a}^{wan}: Limite residual de usuários que podem utilizar o enlace WAN no ponto de acesso a;
- Γ_{prop}^{wan}: Atraso de propagação do enlace WAN;
- Γ_{prop}^{man}: Atraso de propagação do enlace MAN;
- λ: Estimativa da quantidade de tarefas por segundo utilizando o enlace MAN;
- η_{ul}: Tamanho médio dos dados enviados no enlace em *upload*;
- η_{dl}: Tamanho médio dos dados enviados no enlace em *download*.

Funções:

- $M(f,t,u)$: Retorna 1 se a tarefa t, gerada pelo usuário u, estiver incluída no conjunto f;
- $W_{wlan}(x_v^f,a)$: Número de usuários transmitindo dados com destino à máquina virtual v, pelo enlace WLAN, no ponto de acesso a, de acordo com a variável x_v^f;
- $W_{wan}(x_v^f,a)$: Número de usuários transmitindo dados com destino à máquina virtual v, pelo enlace WAN, no ponto de acesso a, de acordo com a variável x_v^f;
- $W_{man}(x_v^f)$: Número de tarefas transmitindo dados através do enlace MAN, de acordo com a variável de decisão x_v^f;
- $A(f)$: retorna a quantidade total de tarefas em f;
- $T(f,v)$: Tempo total para que a máquina virtual v processe a carga de trabalho das tarefas em f.

A.2 Capítulo 4

Variável de decisão:

- x_v^t: Variável de decisão que indica a alocação da tarefa t na maquina virtual v.

Tarefas:

- τ_v^t: Memória RAM demandada pela tarefa t na máquina virtual v.

Rede:

- $B_{wlan}^{a(t)}$: Largura de banda disponível no enlace WLAN do ponto de acesso a;

- $B_{wan}^{a(t)}$: Largura de banda disponível no enlace WAN do ponto de acesso a;
- B_{man}^{t}: Largura de banda no enlace MAN alocada para a tarefa t.

Funções:

- $W(t,u,a)$: Retorna 1 se a tarefa t, gerada pelo usuário u, tenha sido originada no ponto de acesso a. Retorna 0 caso contrário;
- $C(x_v^f, T(x_v^f), c^v)$ ou $C(x_v^t, T(x_v^t), c^v)$: O maior custo entre todas as soluções possíveis;
- $P(x_v^t, \mathcal{T}_{legado}, \phi_u^t, \Phi^v)$: A capacidade de processamento disponível na máquina virtual v considerando a concorrência de recursos inclusive com tarefas legadas;
- $T(x_v^t)$: Tempo total para que a máquina virtual v processe a carga de trabalho da tarefa t.

A.3 Capítulo 5

Conjuntos:

- $\mathcal{T}_{ordenado}$: Conjunto de tarefas ordenadas de acordo com $\Gamma_{esp}^{t,u} - \Delta_u^t$;
- $\mathcal{V}_{ordenado}$: Conjunto de máquinas virtuais ordenadas de acordo com c^v.