UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HENRIQUE VALLE DE LIMA

Controle de admissão para *Network Slicing* considerando recursos de comunicação e computação

Goiânia
2023

UFG

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

[ ] Dissertação [X] 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

Henrique Valle de Lima

3. Título do trabalho

Controle de Admissão para Network Slicing Considerando Recursos de Comunicação e Computação

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 **Sand Luz Corrêa**, **Professor do Magistério Superior**, em 22/05/2023, às 13:53, 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 **Henrique Valle De Lima**, **Discente**, em 22/05/2023, às 14:34, 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 **3767814** e o código CRC **105792F2**.

HENRIQUE VALLE DE LIMA

Controle de admissão para *Network Slicing* considerando recursos de comunicação e computação

Tese apresentada ao Programa de Pós Graduação em Ciência da Computação do Instituto de Informática da Universidade Federal de Goiás, como requisito para obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientadora: Profa. Sand Luz Correa

Co-Orientador: Prof. Kleber Vieira Cardoso

Goiânia
2023

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

Lima, Henrique Valle de
Controle de admissão para Network Slicing considerando recursos de comunicação e computação [manuscrito] / Henrique Valle de Lima. - 2023.
il, 101 f.: il.

Orientador: Prof. Dr. Sand Luz Correa; co-orientador Kleber Vieira Cardoso.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2023.
Bibliografia.
Inclui siglas, abreviaturas, símbolos, gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Network Slicing. 2. 5G. 3. Multi-access edge computing. 4. Controle de Admissão. 5. Multi- Armed Bandits. I. Correa, Sand Luz, orient. II. Título.

CDU 004

UFG

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE TESE

Ata nº **08/2023** da sessão de Defesa de Tese de **Henrique Valle de Lima**, que confere o título de Doutor em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos dez dias do mês de maio de dois mil e vinte e três, a partir das catorze horas, na sala 152 do Instituto de Informática, realizou-se a sessão pública de Defesa de Tese intitulada **"Controle de Admissão para Network Slicing Considerando Recursos de Comunicação e Computação"**. Os trabalhos foram instalados pela Orientadora, Professora Doutora Sand Luz Corrêa (INF/UFG), com a participação dos demais membros da Banca Examinadora: Professor Doutor Kleber Vieira Cardoso (INF/UFG), coorientador; Professor Doutor Joberto Sérgio Barbosa Martins (UNIFACS), membro titular externo; Professor Doutor Cristiano Bonato Both (UNISINOS), membro titular externo; Professor Doutor Antônio Carlos de Oliveira Júnior (INF/UFG), membro titular interno; e Professor Doutor Ronaldo Martins da Costa (INF/UFG), membro titular interno. A participação dos professores Cristiano Bonato Both e Joberto Sérgio Barbosa Martins 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 Tese, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pela Professora Doutora Sand Luz Corrêa, 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 dez dias do mês de maio de dois mil e vinte e três.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **CRISTIANO BONATO BOTH**, **Usuário Externo**, em 10/05/2023, às 17:15, 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 **Ronaldo Martins Da Costa**, **Professor do Magistério Superior**, em 10/05/2023, às 17:16, 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 **Antonio Carlos De Oliveira Junior**, **Professor do Magistério Superior**, em 10/05/2023, às 17:17, 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 **Sand Luz Corrêa**, **Professor do Magistério Superior**, em 10/05/2023, às 17:17, 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 10/05/2023, às 17:18, 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 **Henrique Valle De Lima**, **Discente**, em 10/05/2023, às 17:33, 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 **Joberto Sergio Barbosa Martins**, **Usuário Externo**, em 10/05/2023, às 20:33, 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 **3719005** e o código CRC **2DD05E9E**.

Referência: Processo nº 23070.016623/2023-71 SEI nº 3719005

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

AUTORIZAÇÃO PARA PUBLICAÇÃO DE TESE EM FORMATO ELETRÔNICO

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Controle de admissão para *Network Slicing* considerando recursos de comunicação e computação

Autor(a): Henrique Valle de Lima

Goiânia, 10 de Maio de 2023.

Henrique Valle de Lima – Autor

Sand Luz Correa – Orientadora

Kleber Vieira Cardoso – Co-Orientador

HENRIQUE VALLE DE LIMA

Controle de admissão para *Network Slicing* considerando recursos de comunicação e computação

Tese defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Doutor em Ciência da Computação, aprovada em 10 de Maio de 2023, pela Banca Examinadora constituída pelos professores:

Profa. Sand Luz Correa
Instituto de Informática – UFG
Presidente da Banca

Prof. Kleber Vieira Cardoso
Instituto de Informática – UFG

Prof. Dr. Antônio Carlos de Oliveira Júnior
Instituto de Informática - UFG

Prof. Dr. Ronaldo Martins da Costa
Instituto de Informática - UFG

Prof. Dr. Joberto Sérgio Barbosa Martins
Universidade Salvador - UNIFACS

Prof. Dr. Cristiano Bonato Both
Pós-Graduação em Computação Aplicada - UNISINOS

Henrique Valle de Lima

É Bacharel Engenharia de Computação pela Pontifícia Universidade Católica de Goiás (2011). Durante a graduação, foi bolsista de Iniciação Científica pelo CNPQ, trabalhando no projeto "Estudos de Segregação Residencial na Região Metropolitana de Goiânia: Cruzamento de Dados da Produção Mobiliária com a Tipologia Sócio-Espacial do Observatório das Metrópoles". É pós-graduado pela Faculdade de Tecnologia Senai de Desenvolvimento Gerencial (SENAI-FATESG) em Segurança em Redes de Computadores (2015) e também pela Faculdade Apogeu em Docência no Ensino Superior (2017). Recebeu o título de Mestre em Ciência da Computação (2018) pela Universidade Federal de Goiás (UFG), desenvolvendo a dissertação "Alocação de Recursos em Bandas Não Licenciadas". Durante o mestrado, atuou como bolsista no projeto de pesquisa ATER (Aceleração do Transporte de Dados com o Emprego de Redes de Circuitos Dinâmicos), mantido pela Rede Nacional de Pesquisa (RNP). Atualmente, está fase de finalização do Doutorado em Ciência da Computação, pela Universidade Federal de Goiás (UFG), onde desenvolve a tese "Controle de admissão para *Network Slicing* considerando recursos de comunicação e computação". Tem experiência como docente nas áreas de computação e orientou projetos de Conclusão de Curso (2015) vinculados ao Instituto de Tecnologia de Goiás (ITEGO). Atualmente, é professor no Instituto Federal de Goiás (IFG) e na Universidade Evangélica de Goiás (UniEVANGÉLICA). Possui interesse em projetos de pesquisa relacionados à área de Engenharia/Ciências da Computação, com ênfase em redes sem fio, redes 5G, segurança de redes de computadores, segurança da informação, computação paralela e distribuída, sistemas inteligentes, soluções em aplicações com a tecnologia RFID e sistemas de informação voltados para a educação.

Dedico esta tese primeiramente à Deus, que me sustenta e me guia durante todos os dias da minha vida. Aos meus pais, que me amam incondicionalmente. À minha família e aos meus amigos, que me dedicam todo apoio e carinho necessários para uma vida feliz e completa.

Agradecimentos

Primeiramente, agradeço a Deus pela graça da vida e por tudo que Ele me proporciona. Ele é meu centro e meu alvo, e nele encontro minhas forças diárias. É por causa dele que estou de pé e nunca desisto de lutar.

Quero expressar minha gratidão ao meu pai, Gilson, que é meu espelho, imagem e exemplo. Palavras nunca serão capazes de descrever todo o amor que sinto por você.

Agradeço também à minha mãe, Regimar, por todo o amor, carinho, orientação e paciência que ela tem dedicado a mim durante toda vida. Obrigado por me dar a oportunidade de ser seu filho.

Minhas avós Ivone, Maria e minha bisavó Lia, meu avô Itamar e meu bisavô Ovídio, em memória, que me ensinaram grande parte de tudo o que sei e sou, principalmente, a ser amável, respeitoso e gentil. Dedicar amor ao próximo sem receber nada em troca, é uma lição que aprendi com esses grandes exemplos.

Gostaria de agradecer ao Raffa, meu companheiro, amigo e confidente. Obrigado por me entender nos momentos difíceis da minha vida, por estar ao meu lado e nunca me abandonar. Obrigado por me incentivar a crescer e me tornar uma pessoa melhor.

Agradeço ainda a Cristiane, Silas, Rubens, Pedro e Lucília, uma família linda que Deus me deu, que me acolhe e me dedica tanto amor e carinho que as vezes me pergunto se realmente sou merecedor. Sou mais feliz ao lado de vocês.

Também quero expressar minha gratidão à Profa. Sand Luz Correa, por sua orientação, amizade, paciência e confiança. Obrigado por me ensinar a ser forte e persistir em meu objetivo, por mais difíceis que possam parecer.

Agradeço ao Prof. Kleber Vieira Cardoso, por sua orientação, contribuição e confiança.

Por fim, agradeço ao INF/UFG por proporcionar a realização desse passo importante em minha vida, e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e à Rede Nacional de Pesquisa (RNP) pelo suporte financeiro.

“O Senhor Deus firma os passos de um homem, quando a conduta deste o agrada; ainda que eu tropece, não cairei, pois o Senhor meu Deus me toma pela mão.”

Salmos, 37:23-24,
Bília Sagrada.

Resumo

Lima, Henrique. **Controle de admissão para *Network Slicing* considerando recursos de comunicação e computação**. Goiânia, 2023. 95p. Tese de Doutorado Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

As redes 5G, possibilitaram a aplicação de diversas tecnologias inovadoras e disruptivas como *Network Function Virtualization* (NFV) e *Software-Defined Networking* (SDN). Juntas, essas tecnologias atuam como habilitadoras de *Network Slicing* (NS), transformado a forma de operar, gerenciar e monetizar as redes. Através de conceitos de *Slice-as-a-Service* (SlaaS), as operadoras de telecomunicação podem monetizar a infraestrutura física e lógica, oferecendo *slices* na rede a novos clientes, como indústrias verticais. Esta tese aborda o problema de controle de admissão de inquilinos com uso de NS. Propompos três modelos de controle de admissão para NS (MONETS-OBD, MONETS-OBS e CA-ONS) que consideram recursos de comunicação e computação. Para avaliar os modelos propostos, comparamos os resultados com diferentes algoritmos clássicos da literatura, como o eUCB, o ϵ-*greedy* e o ONETS. Utilizamos dados de diferentes aplicações para enriquecer a análise. Os resultados indicam que as heurísticas MONETS-OBD, MONETS-OBS e CAONS realizam um controle de admissão que se aproxima do conjunto de soluções ideais. Atingimos alta eficiência das heurísticas MONETS-OBD e MONETS-OBS em controlar a admissão dos inquilinos, atingindo taxas de aceitação de requisições de até 99% em alguns casos. Além disso, a heurística CAONS, que utiliza penalizações, além de alcançar taxas de aceitação e recompensa próximas da solução ótima, ainda reduz significativamente o número de violações da capacidade do sistema. Por fim, os resultados evidenciam que o processo de controle de admissão de *slices* deve considerar os recursos de comunicação e os recursos de computação, escassos na borda da rede. Uma solução que considere somente os recursos de comunicação pode levar a interpretações incorretas e inviáveis, superestimando a capacidade dos recursos de computação.

Palavras–chave

Network Slicing, 5G, *Multi-access edge computing*, controle de admissão, Multi-Armed Bandits

Abstract

Lima, Henrique. **Admission control for Network Slicing considering communication and computing resources**. Goiânia, 2023. 95p. PhD. Thesis Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

The 5G networks have enabled the application of various innovative and disruptive technologies such as Network Function Virtualization (NFV) and Software-Defined Networking (SDN). Together, these technologies act as enablers of Network Slicing (NS), transforming the way networks are operated, managed, and monetized. Through the concept of Slice-as-a-Service (SlaaS), telecommunications operators can monetize the physical and logical infrastructure by offering network slices to new customers, such as vertical industries. This thesis addresses the problem of tenant admission control using NS. We propose three admission control models for NS (MONETS-OBD, MONETS-OBS, and CAONS) that consider both communication and computation resources. To evaluate the proposed models, we compare the results with different classical algorithms from the literature, such as eUCB, ϵ-greedy, and ONETS. We use data from different applications to enrich the analysis. The results indicate that the MONETS-OBD, MONETS-OBS, and CAONS heuristics perform admission control that approaches the set of ideal solutions. We achieve high efficiency with the MONETS-OBD and MONETS-OBS heuristics in controlling tenant admission, reaching acceptance rates of up to 99% in some cases. Furthermore, the CAONS heuristic, which employs penalties, not only achieves acceptance and reward rates close to the optimal solution but also significantly reduces the number of capacity violations. Lastly, the results highlight that the process of slice admission control should consider both communication and computation resources, which are scarce at the network edge. A solution that considers only communication resources can lead to incorrect and unfeasible interpretations, overestimating the capacity of computation resources.

Keywords

Network Slicing, 5G, Multi-access edge computing, admission control, Multi-Armed Bandits

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Abreviaturas

3GPP: *3rd Generation Partnership Project*
4G: Redes móveis de 4^a geração
5G: Redes móveis de 5^a geração
6G: Redes móveis de 6^a geração
CN: *Core Network*
D2D: *Device to Device*
eMBB: *Enhanced Mobile Broadband*
ETSI: *European Telecommunications Standards Institute*
eUCB: *enhanced Upper Confidence Bound*
FDD: *Frequency Division Duplex*
IA: Inteligência Artificial
IoT: *Internet of Things*
M-MIMO: *Massive Multiple Input Multiple Output*
M-ONETS: *Online Network Slice Broker with MEC*
M2M: *Machine to Machine*
MaB: *Multi-armed Bandit*
MCC: *Mobile Cloud Computing*
ME: *Mobile Edge*
MEC: *Multi-Access Edge Computing*
MIMO: *Multiple Input Multiple Output*
mMTC: *Massive Machine Type Communication*
mmWave: *Millimeter-wave*
NFV: *Network Function Virtualisation*
NGMN: *Next Generation Mobile Networks*
NS: *Network Slicing*
NSIL: *Network Slice Instance Layer*
ONETS: *Online Network Slice Broker*
OTTs: *Over-The-Top*
PRBs: *Physical Resource Block*
QAM: *Quadrature Amplitude Modulation*

QoE: *Quality of Experience*
QoS: *Quality of Service*
RAN: *Radio Access Network*
RL: *Reinforcement Learning*
SDN: *Software Defined Networking*
SIL: *Service Instance Layer*
SLA: *Service Level Agreement*
SRL: *Slice Resource Layer*
TN: *Transport Network*
UEs: *User Equipments*
URLLC: *Ultra-Reliable e Low Latency Communication*
V2V: *Vehicle-to-vehicle communication's*
VMNOs: *Virtual Mobile Network Operators*
VNE: *Virtual Network Embedding*
VNF: *Virtualized Network Function*

CAPÍTULO 1

Introdução

O advento da Internet trouxe consigo grandes melhorias na forma de comunicação utilizada pelo ser humano. Evoluções como a criação de dispositivos móveis inteligentes, o surgimento de redes sociais e a implementação de aplicações de vídeo por *streaming* impulsionaram fortemente o crescimento acelerado no consumo de dados [Cominardi et al. 2020]. Estimativas de crescimento de demanda de dados, apresentadas em março de 2022 pela Cisco, demonstram que até 2025 teremos um total de 59,4 bilhões de dispositivos móveis e conexões de internet, os quais serão responsáveis por um volume aproximado de 87 exabytes por mês sendo transmitidos por toda rede mundial de telecominucações [CISCO 2022].

Nesse contexto, a rede 5G foi proposta como uma maneira de atenuar o impacto causado pela crescente demanda de conexão gerada, agregando de maneira eficiente, na visão da operadora de telecomunicação, importantes características de flexibilidade e modularização. Essas características podem ser exploradas através do uso de tecnologias habilitadoras das redes 5G, como *Software Defined Networking* (SDN), *Network Function Virtualisation* (NFV) e principalmente *Network Slice* (NS). Além disso, a rede 5G traz consigo novas oportunidades de monetização, diante da oferta de serviços inovadores e especializados à diferentes setores verticais (industrias automotivas, *eHealth*, *Energy-efficient Management Systems*, *Over-The-Top* (OTTs) e outros) [Kukliński e Tomaszewski 2019].

Tecnologias como NFV e SDN estão sendo utilizadas para atender a requisitos de escalabilidade, flexibilidade, agilidade e programabilidade. Nesse contexto, o NFV é encarregado de atender ao conjunto de requisitos das aplicações utilizando computação e tecnologias nativas de nuvem, ao mesmo tempo que oferece suporte para serviços virtualizados [Cao, Kodialam e Lakshman 2014]. Por outro lado, a SDN é focada em tornar programáveis os serviços de conectividade fornecidos pelas redes 5G. Essa tecnologia, em particular, é responsável por direcionar os fluxos de tráfego na rede e gerenciar os recursos disponíveis de forma dinâmica. Dessa maneira, por meio da combinação de NFV e SDN, espera-se obter o máximo de desempenho que as redes 5G podem oferecer [Husni e Bramantyo 2018].

O NS pode ser visto como o habilitador mais básico e certamente mais complexo das redes 5G, e atua ao lado de tecnologias habilitadoras como NFV e SDN, modularizando a rede e construindo, sobre uma mesma infraestrutura física, diversos *slices* customizados de maneira independente, cada um, atendendo às necessidades únicas e particulares de uma aplicação, i.e., aos requisitos específicos de cada serviço - *Service Level Agreement* (SLA). No contexto de redes virtualizadas de telecomunicações, um *slice* é composto por um conjunto, possivelmente encadeado, de *Virtualized Network Function*s (VNFs), criados sob demanda, seguindo o princípios de isolamento dos recursos, de automação, customização, elasticidade, programabilidade e de entrega do serviço *end-to-end* [Rifai e Supriyanto 2017].

Em 5G, cada aplicação possui um conjunto de requisitos específicos de *Quality of Service* (QoS) para uma rede formada por recursos finitos. Dessa maneira, controlar os recursos disponíveis de maneira a melhorar a utilização, e ainda, realizar um controle de admissão de clientes ou inquilinos (*tenants*) visando o aumento do lucro da operadora de maneira eficaz, representam um grande desafio.

Dessa maneira, dois problemas clássicos são amplamente abordados - o controle de admissão de *slices* e a alocação eficiente de recursos para os *slices*. O controle de admissão de *slices* é um processo de verificação e decisão destinado a permitir ou restringir o acesso de inquilinos aos recursos na rede da operadora, considerando um ou mais critérios de retorno. A alocação de recursos, no contexto de uma rede virtualizada de telecomunicações, para um *slice*, pode ser vista como um problema de *Virtual Network Embedding* (VNE), mapeando redes virtuais, criadas para cada inquilino, em redes físicas. Desse modo, aplicar técnicas de controle de admissão de *slices* e alocação de recursos na rede 5G é fundamental para maximizar o lucro da operadora [Feng et al. 2020].

Nesta tese, propomos três soluções *on-the-fly* para o controle da admissão de *slices* adotando um caso de uso da rede 5G. Inicialmente, estudamos o problema clássico de controle de admissão no contexto de *network slicing* quando tradado através de um arcabouço de técnicas do tipo *Multi-armed Bandit* (MaB). Além disso, utilizamos técnicas de *Reinforcement Learning* (RL), em heurísticas com retro-alimentação, para resolver o problema apresentado, considerando-o como um *Multi-armed Bandit* (MaB) com restrições de *lock-up* forçado. Com base em um conjunto de soluções MaB, desenvolvemos os algoritmos *Online Network Slicing with MEC and Overbooking by Demand* (MONETS-OBD) e *Online Network Slicing with MEC and Overbooking by Empirical Rule* (MONETS-OBS), os quais consideram recursos de comunicação e recuros de computação de borda (*Multi-Access Edge Computing* (MEC)) e utilizam amostras de recompensas recebidas historicamente para equilibrar a relação *exploração* x *aproveitamento*. Em seguida, criamos um novo algoritmo, denominado Controle de Admissão Online para RAN Slicing (CAONS), que seleciona o conjunto de *slices* a serem atendidos em cada

rodada, levando em consideração tanto as recompensas recebidas, quanto as penalidades sofridas durante o aprendizado. O algoritmo CAONS fornece o conjunto de soluções em tempo polinomial, tornando-se adequado para soluções *online*. Além disso, comparamos nossas soluções com uma heurística não-guiada, o ϵ-Greedy, uma solução baseada na recompensa instantânea, o *enhanced Upper Confidence Bound* (eUCB), e uma solução estado-da-arte da literatura, o *Online Network Slice Broker* (ONETS). Para uma análise abrangente, foram utilizados perfis de diversas aplicações do mundo real, a fim de enriquecer a avaliação das diferentes soluções propostas. Os resultados obtidos indicam que as heurísticas MONETS-OBD, MONETS-OBS e CAONS são capazes de se aproximar da solução ótima. Destaca-se a eficiência das heurísticas MONETS-OBD e MONETS-OBS no controle de admissão dos inquilinos, alcançando taxas de aceitação de requisições de até 99% em alguns casos. Além disso, a heurística CAONS, que utiliza penalizações, não só atinge taxas de aceitação e recompensa próximas às do algoritmo ótimo, mas também reduz significativamente o número de violações do sistema. Esses resultados reforçam a importância de considerar a penalização como um componente crucial no processo de controle de admissão. Os resultados também evidenciam a necessidade de considerar tanto os recursos de comunicação quanto os recursos de computação na alocação de recursos. É importante destacar que os recursos de borda da rede são particularmente escassos e devem ser levados em consideração. Uma solução que leva em conta apenas recursos de comunicação pode levar a interpretações incorretas e inviáveis, superestimando a capacidade dos recursos de computação. Esses resultados reforçam a importância do desenvolvimento de soluções que integrem de forma eficiente os recursos de comunicação e computação, visando um controle de admissão robusto e eficaz no contexto do *network slicing* nas redes 5G.

Esta tese está estruturada de forma a apresentar o desenvolvimento do estudo realizado. No Capítulo 2, apresentamos os fundamentos teóricos necessários para o entendimento do trabalho desenvolvido, além de uma revisão dos trabalhos relacionados na área; no Capítulo 3 apresentamos uma visão do problema clássico de controle de admissão de *slices* tratado como um arcabouço de soluções MaB; no Capítulo 4, descrevemos as estratégias de controle de admissão desenvolvidas com o objetivo de oferecer suporte e serviço às indústrias verticais; no Capítulo 5, apresentamos a avaliação de desempenho realizada, com os resultados obtidos e as discussões pertinentes; e finalmente, no Capítulo 6, apresentamos as considerações finais sobre o trabalho desenvolvido e discutimos as perspectivas de trabalhos futuros, destacando as contribuições e as limitações do estudo.

CAPÍTULO 2

Fundamentação Teórica e Trabalhos Relacionados

No presente capítulo, apresentamos uma visão geral sobre os principais fundamentos das tecnologias envolvidas na elaboração deste trabalho. Nas seções a seguir, fornecemos uma visão geral sobre o contexto das redes 5G, bem como as estratégias que possibilitam a implementação da proposta deste trabalho: NS e computação de borda. Discutimos também os principais trabalhos relacionados com o tema de investigação desta tese. Por fim, apresentamos algumas considerações finais.

2.1 Fundamentação Teórica

A evolução das redes móveis de 4^a geração (4G) para o 5G trouxe um paradigma disruptivo: a rede 5G foi concebida para atender serviços com requisitos díspares e desafiadores e acelerar a transformação digital em diversos segmentos verticais. Para alcança esse objetivo, o 5G deve ser um sistema flexível, programável e eficiente, exigindo uma gerência de recurso otimiza e sob demanda em praticamente todos os segmentos da rede. Tal gerenciamento não é uma tarefa simples. No entanto, tecnologias como NS e computação de borda, e soluções baseadas em Inteligência Artificial (IA), discutidas neste capítulo, são essenciais para tornar essa tarefa possível.

2.1.1 Redes 5G

A quinta geração de rede móvel celular, também conhecida como 5G, representa uma evolução significativa em relação às gerações anteriores [3GPP 2019, 3GPP 2020, 3GPP 2021]. A tecnologia 5G permite taxas de dados 100 vezes maiores que as disponíveis no 4G, sendo capaz de conectar um número de dispositivos por km^2 até 100 vezes maior que o 4G LTE. Além disso, o 5G pode apresentar uma latência fim-a-fim, de apenas 5 ms, que é uma fração da latência típica em redes sem fio atuais. Além de oferecer maior velocidade e capacidade de transmissão de dados, as redes 5G também foram projeta-

das para atender demandas socioeconômicas, fornecendo acesso de qualidade à Internet em áreas remotas e ampliando a inclusão digital [Alliance 2016]. Finalmente, o 5G também foi concebido para suportar a execução de uma ampla gama de novas aplicações. Essas aplicações se enquadram em três categorias, cada uma com requisitos muito diferentes: *Enhanced Mobile Broadband* (eMBB), focado na melhoria de banda larga móvel [Popovski et al. 2018]; *Ultra-Reliable e Low Latency Communication* (URLLC), entregando comunicação ultra confiável e de baixa latência [Demmer et al. 2018]; e *Massive Machine Type Communication* (mMTC), possibilitando a comunicação massiva entre dispositivos na rede [Bairagi et al. 2020]. A implantação das redes 5G iniciou-se em 2020 e atualmente está em expansão ao redor do mundo [Wijethilaka e Liyanage 2021].

Apesar de trazerem grandes mudanças em termos de requisitos, casos de uso e tecnologias, os componentes de uma rede 5G permanecem praticamente os mesmos das gerações anteriores, sendo constituídos pela rede de acesso via rádio (*Radio Access Network* (RAN)), rede de transporte (Transport Network (TN)) e rede de núcleo (Core Network (CN). A RAN é responsável pela conexão entre os dispositivos móveis dos usuários (*User Equipment*s (UEs)) e as estações bases. A rede de núcleo permite que os UEs enviem ou recebam tráfego móvel de ou para aplicativos hospedados na rede de dados ou na Internet. Por fim, a rede de transporte é responsável por fornecer conectividade desde o ponto de entrada da rede para o UE até a rede de dados, onde normalmente as aplicações estão hospedadas.

Para alcançar os requisitos almejados, a tecnologia 5G se baseia em um conjungo de tecnologias habilitadoras, incluindo: NS, para possibilitar a coexistência de diversas redes lógicas customizadas para diferentes aplicações sobre a mesma infraestrutura física; computação de borda, para oferecer recursos de computação e armazenamento mais próximos do usuário final e reduzir a latência de comunicação entre os UEs e suas aplicações; *Millimeter-wave* (mmWave), que possibilita a comunicação através de ondas milimétricas, *Massive Multiple Input Multiple Output* (M-MIMO), que atua com múltiplas entradas e múltiplas saídas de comunicação; e *Ultra-Dense Network* (UDN), o qual entrega redes de acesso ultra densas [Ahmad et al. 2018].

NS é uma tecnologia essencial nas redes 5G, permitindo a criação de redes virtuais independentes em uma única rede física. Com NS, é possível fatiar recursos de rede (como largura de banda), processamento e armazenamento de maneira eficiente e flexível. Dessa forma, as operadoras podem oferecer serviços personalizados e com maior qualidade, enquanto maximizam a utilização de seus recursos de rede. Por exemplo, uma operadora pode criar um *slice* de rede personalizado para dispositivos *Internet of Things* (IoT), com requisitos de baixa latência e alta confiabilidade, enquanto outro *slice* pode ser destinado a aplicações de realidade virtual, com requisitos de alta largura de banda e baixa latência [Wijethilaka e Liyanage 2021].

Com cada vez mais dispositivos sem fio conectados na Internet e com o crescimento acelerado da demanda, o espectro disponível para transmissão de dados tem se tornado cada vez mais congestionado. Nesse contexto, a mmWave (MMWV) propõem a utilização de frequências milimétricas como uma solução para esse problema de congestionamento. Com o uso de frequências subutilizadas, entre 30 e 300 GHz, os sistemas de comunicação podem atingir taxas de dados na ordem de Gigabytes por segundo (GB/s) em uma distância de até alguns quilômetros [Naqvi et al. 2020].

Nesse contexto, os sistemas M-MIMO permitem o aumento da eficiência energética nas estações de transmissão e recepção de dados, também conhecidas como estações bases, contribuindo diretamente com um dos objetivos do 5G: redução do consumo de energia nas estações. Uma estação base equipada com M-MIMO conta com um grande número de antenas celulares, que contribuem com melhorias de eficiência espectral e energética, a um baixo custo de processamento e gerenciamento das antenas [Albreem, Juntti e Shahabuddin 2019].

Desde a criação das redes móveis, o processo de divisão e densificação das células representam um dos meios mais eficazes para incremento da capacidade de transmissão e cobertura do sinal. Com as redes 5G, a densificação das células se tornou ainda mais importante para garantir a disponibilidade de largura de banda e de baixa latência necessárias para suportar novas aplicações e serviços. A UDN é caracterizada por ser uma rede de maior densidade de recursos de rádio, que atinge ou até mesmo ultrapassa a densidade dos usuários em uma estação base. Além disso, o conceito de NS permite a criação de *slices* de rede dedicadas para aplicações específicas, possibilitando um melhor gerenciamento de recursos e garantindo a qualidade de serviço para cada *slice*. Com isso, a densificação das células, aliada ao conceito de NS, se torna essencial para a operação de uma rede 5G eficiente e capaz de suportar as demandas de conectividade e velocidade cada vez maiores [Chih-Lin, Kuklinskí e Chen 2020].

Nesse contexto, SDN e NFV são importantes tecnologias habilitadoras de NS. SDN atua desacoplando o plano de controle do plano de dados da rede, permitindo que a organização e o estado da rede sejam centralizados logicamente, possibilitando que a infraestrutura subjacente seja abstraída da visão das aplicações. SDN baseia-se em quatro características principais: a separação do plano de controle do plano de dados, a utilização de um controlador centralizado que tem a visão geral da rede, o uso de interfaces abertas entre o plano de controle e o plano de dados, e por fim, a programabilidade da rede[Zaidi et al. 2018]. Já NFV, por outro lado, permite realizar a virtualização de funções inteiras da rede que antes eram vinculadas ao hardware proprietário, e a execução acontecia apenas em hardware de propósito geral. O principal componente do NFV são as VNFs, funções de rede implementadas em software e instanciadas na forma de máquinas virtuais ou contêineres [Mekikis et al. 2019].

A computação de borda introduz recursos de processamento e/ou armazenamento próximos aos usuários finais para servir aplicações que executam nos UEs, evitando que essas aplicações tenham que ser servidas por servidores remotos que executam em infraestruturas de nuvem. Como consequência, a latência de comunicação entre os UEs e os recursos de borda é reduzida quando comparada com a latência de comunicação com recursos de nuvem. Adicionalmente, como não há necessidade do tráfego dos UEs alcançarem a nuvem, a rede de núcleo pode ser aliviada, reduzindo congestionamentos na saída para a Internet.

A definição do sistema 5G apresentada pelo *3rd Generation Partnership Project* (3GPP) apresenta uma visão sistêmica do mesmo. Nesta visão, o sistema é composto por dois subsistemas principais: o Novo Rádio 5G (*new 5G Radio*) e o Novo Núcleo 5G (*new 5G Core*). O Novo Rádio 5G, encontrado na RAN, é constituído por um conjunto de estações bases, denominadas *Next-Generation Base Station* (gNBs), as quais proveem conectividade aos UEs através da tecnologia *5G New Radio* (NR). Uma gNB pode ser logicamente dividida em três entidades: *Radio Unit* (RU), *Distributed Unit* (DU) e *Centralized Unit* (CU). As funções do protocolo NR executadas em cada uma dessas entidades são determinadas pelo tipo de desagregação (*split*) escolhido, sendo definidos 8 tipos diferentes pelo 3GPP.

A arquitetura do novo Núcleo 5G (CN) é orientada a serviços, permitindo que as funções de rede possam ser implantadas em infraestruturas de computação em nuvem, utilizando tecnologias como NFV e SDN. Além disso, o projeto arquitetural do Núcleo 5G distingue entre funções de rede de plano de controle e plano de usuário, permitindo implantações flexíveis em locais centralizados ou na borda da rede, evolução tecnológica independente e escalabilidade [Henry, Alsohaily e Sousa 2020].

As redes 5G foram projetadas para suportar uma ampla variedade de aplicações que fornecem ao usuário uma experiência aprimorada, com a capacidade de atender às necessidades específicas da indústria vertical, bem como aos tradicionais serviços de comunicação móvel [Alliance 2018]. Para atingir esse objetivo, diversos habilitadores são cruciais. A seguir, descrevemos, com mais detalhes, os dois habilitadores que compõem o foco desta tese: NS e computação de borda.

2.1.2 *Network Slicing*

O conceito de *Network Slice* (NS), proposto no contexto da proposta da *Next Generation Mobile Networks* (NGMN), refere-se à criação e operação de múltiplas redes logicamente independentes sobre uma infraestrutura de telecomunicação compartilhada. Cada rede lógica, denominada *network slice* ou simplesmente *slice*, consiste em um conjunto de VNFs, bem como recursos de computação, armazenamento e de rede para executá-las. Essas funções e recursos formam uma rede lógica para atender, de forma

customizada, as características específicas (e.g., SLA) do serviço representado pelo *slice*. Um *slice* é um conceito fim-a-fim, ou seja, a rede lógica pode atravessar todos os domínios tecnológicos da rede do provedor de telecomunicações, incluindo a rede de acesso - RAN, a rede de transporte - TN e o core da rede - CN.

A implementação de NS traz um novo modelo de negócio para os provedores de telecomunicação, os quais podem alugar seus recursos para novos clientes, geralmente denominados inquilinos (*tenants*). Exemplos de inquilinos incluem *Virtual Mobile Network Operators* (VMNOs) ou indústrias. Esses novos clientes atuam como locatários da infraestrutura e recebem algum nível de controle, acordado com o operador, sobre os recursos alocados. No entanto, a alocação eficiente dos recursos e o isolamento total entre os diferentes inquilinos constituem os principais desafios na realização do conceito de NS. De fato, os *slices* devem ser criadas dinamicamente seguindo os seguintes princípios de: [Afolabi et al. 2018].

- **isolamento**: cada *slice* exige independência de controle e gerenciamento. Isso pode ser obtido através de diferentes recursos físicos, compartilhados e isolados por virtualização, ou recursos compartilhados e isolados por políticas de controle de acesso;
- **automação**: *slices* devem ser criados sob demanda de forma automatizada, com o mínimo de interferência humana;
- **customização**: cada *slices* deve ser criado de forma a atender, de forma customizada, o serviço a ele associado. Esses serviços podem ter diferentes requisitos de desempenho como latência, confiabilidade ou vazão;
- **elasticidade**: os recursos atribuídos a um *slice* podem ser redimensionados (*scale up/down)* para garantir cumprimento do SLAs;
- **programabilidade**: recursos alocados a um *slice* devem ser controlados de forma programática via interfaces abertas;
- **fim-a-fim**: *slices* são criados para dar garantias fim-a-fim a um serviço, isto é, garantido desde o provedor de serviço (inquilino) até os usuários finais. Isto, por sua vez, pode envolver diferentes segmentos de rede, como a RAN, CN e a TN.

A Figura 2.1 apresenta uma infraestrutura onde recursos de rede e computação são compartilhados entre diferentes *slices*, cada um com um propósito específico [Hattachi e Erfanian 2015]. O primeiro *slice*, identificado como *slice 1* na figura, representa um serviço de dados e voz para *smartphones*, caracterizado como um serviço típico da categoria eMBB. Esse tipo de serviço requer várias VNFs ativas e distribuídas ao longo da rede (borda e núcleo). O segundo *slice* é voltado para um serviço de carros autônomos, um caso de uso da categoria URLLC, em que requisitos de segurança, confiabilidade e baixa latência são essenciais. Para atender a esse requisito de baixa latência, as VNFs e

a aplicação do inquilino devem ser instanciadas em servidores de propósito geral localizados na borda da rede (*cloud edge nodes*). Por fim, o terceiro *slice* está associado a um serviço massivo de IoT, um caso de uso da categoria mMTC. Nessa categoria, o conjunto de VNFs é limitado em comparação com os *slices* anteriores. De fato, essa categoria inclui serviços mais simples que não exigem mobilidade e conectividade sempre ativa. É importante notar que, para atender às demandas desses serviços, cada *slice* deve ser projetado de forma específica e com recursos apropriados para cada tipo de aplicação. Desse modo, NS possibilita a configuração de *slices* de forma isolada, com funções específicas para cada serviço, ao mesmo tempo que compartilha a infraestrutura e reduz os custos. O compartilhamento dos recursos deve ser automatizado e flexível, permitindo que se ajuste à demanda atual dos *slices*.

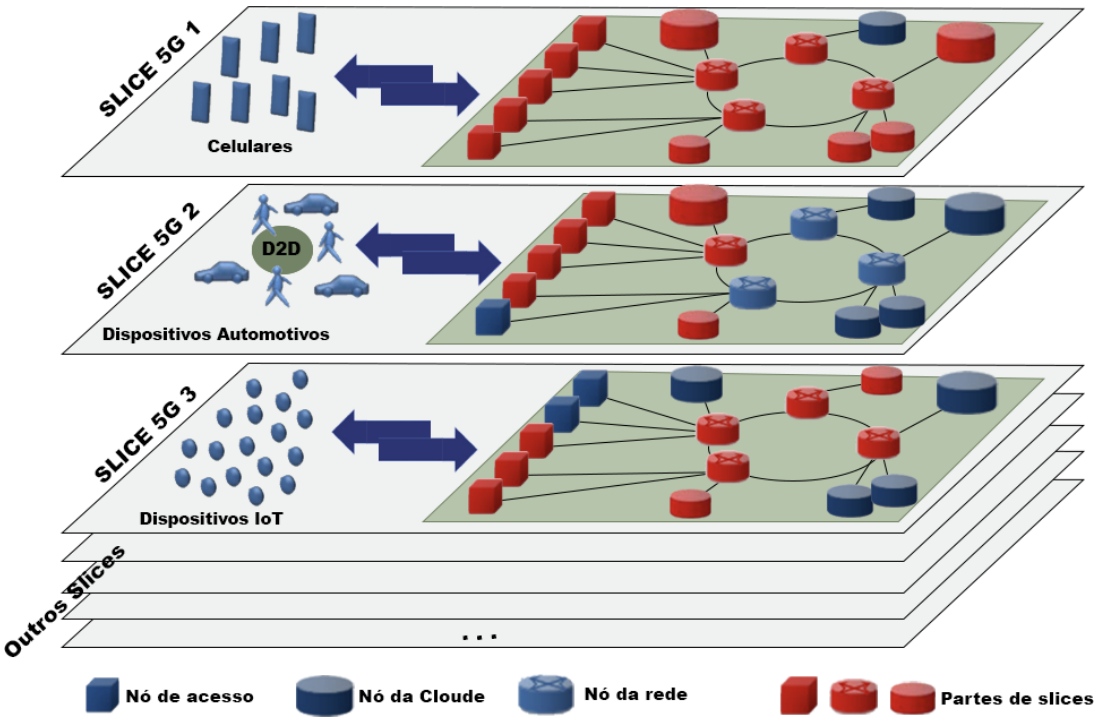


Figura 2.1: Exemplo de *slices* em redes 5G

De acordo com a NGMN, uma arquitetura de *Network Slice* compreende três camadas principais: *Service Instance Layer* (SIL), *Network Slice Instance Layer* (NSIL) e *Slice Resource Layer* (SRL) [Alliance 2018]. A SIL, trata dos serviços a serem suportados. Uma instância de serviço é composta por qualquer serviço que necessite de um *slice* fornecido por um inquilino da infraestrutura. A camada intermediária, NSIL, representa o conjunto de recursos e funções de rede que compõem a rede lógica formada para atender aos requisitos da SIL. As instâncias de *Network Slice* podem ser compartilhadas por vários serviços, assim, um *slice* pode ser visto como instâncias menores da rede, que carregam um conjunto de recursos e funções da rede. Por fim, a camada de mais baixo nível, SRL, compreende todos os recursos físicos da infraestrutura.

Segundo a NGMN [Alliance 2016], o ciclo de vida de um *slice* é dividido em três etapas: comissionamento, operação e descomissionamento. Na etapa de comissionamento, o inquilino da infraestrutura deve solicitar um *slice*, especificando as características da aplicação e os requisitos de QoS. A operadora recebe a solicitação e deve avaliar se existem recursos disponíveis para atender a requisição. É possível que esse processo seja composto por várias etapas de negociação entre o operador e o inquilino. Caso o *slice* seja aceito, a operadora atualizará a lista de *slices* ativos inserindo informações do novo *slice*. Inicia-se então a fase de operação, onde os recursos para o *slice* devem ser dimensionados dinamicamente e de forma elástica, de acordo com a demanda do serviço no período corrente. Por fim, a fase de descomissionamento, pode ser iniciada por dois motivos principais: ausência de recursos na operadora ou finalização do período contratado. Caso o descomissionamento seja iniciado por falta de recursos, os *slices* removidos podem ser movidos para outra rede de propriedade da operadora, caso hajam recursos disponíveis. Caso não hajam recursos em outra rede, as conexões são encerradas e o inquilino informado. Quando o descomissionamento ocorre por finalização do período contratado, as conexões são simplesmente encerradas, e o inquilino é informado.

2.1.3 Computação de Borda

A densificação de UEs na borda da Internet, como, por exemplo, numerosos *smartphones* e a implementação de diversos veículos inteligentes, tem motivado o surgimento de novas aplicações que demandam recursos específicos e ao mesmo tempo, abundantes de processamento e/ou armazenamento. Como grande parte destes dispositivos possuem restrições de recursos e de bateria, uma solução natural é distribuir as funcionalidades dessas novas aplicações entre o UE utilizando recursos de servidores localizados na nuvem. Dessa forma, aplicações que executariam no UE passam a fazer requisições de execução da aplicações em servidores remotos, posicionados na nuvem. Entretanto, servidores na nuvem geralmente encontram-se topologicamente distantes dos UEs e, portanto, algumas requisições realizadas pelos dispositivos normalmente atravessam várias redes até chegar no servidor de nuvem. Enquanto essa divisão é adequada para algumas aplicações, como por exemplo vídeo por *streamming*, para outras, sensíveis à latência (URLLC), o processamento remoto pode ocasionar na degradação da qualidade de experiência (*Quality of Experience* (QoE)) do usuário ou até mesmo inviabilizar o funcionamento correto da aplicação [Cruz, Achir e Viana 2022]. De fato, para aplicações que exibem baixa latência, como indústria 4.0, cirurgia remota, robôs colaborativos, carros autônomos e realidade aumenta e virtual, uma solução mais adequada é servi-las usando recursos de processamento e/ou armazenamento localizados próximos aos usuários finais. Esse paradigma de computação é denominado computação de borda (*Edge Computing*).

A computação de borda é realizada por uma arquitetura distribuída, composta por três camadas principais: a camada de dispositivos, a camada de borda e a camada de nuvem [Ricart-Sanchez et al. 2019]. A Figura 2.2 apresenta uma representação dessa distribuição. A camada de dispositivos é formada pelos UEs, como smartphones, tablets e dispositivos IoT. A camada de borda é composta por servidores, recursos de armazenamento e também recursos de rede que estão localizados mais próximos dos dispositivos de usuário final. Essa camada executa os serviços e aplicações de forma mais eficiente, aproveitando a proximidade com os dispositivos finais. A camada de nuvem fornece recursos de computação e armazenamento para suportar a camada de borda. Essa arquitetura também apresenta outros benefícios, como a redução da carga no núcleo da rede, diminuição do tempo de resposta e melhoria na eficiência dos recursos de rádio [Taleb et al. 2017].

Nesse contexto, uma questão importante é o limite da borda da rede. Recursos de borda podem ser alocados de forma co-localizada com estações bases, ou com algum equipamento da RAN ou ainda na própria CN. À medida que se aproximam dos dispositivos sem fio, no entanto, os recursos de borda tendem a se tornar mais escassos. Portanto, recursos de borda disponíveis em estações bases tendem a ser alocados para aplicações com restrições mais severas de latência como cirurgia remota ou carros autônomos.

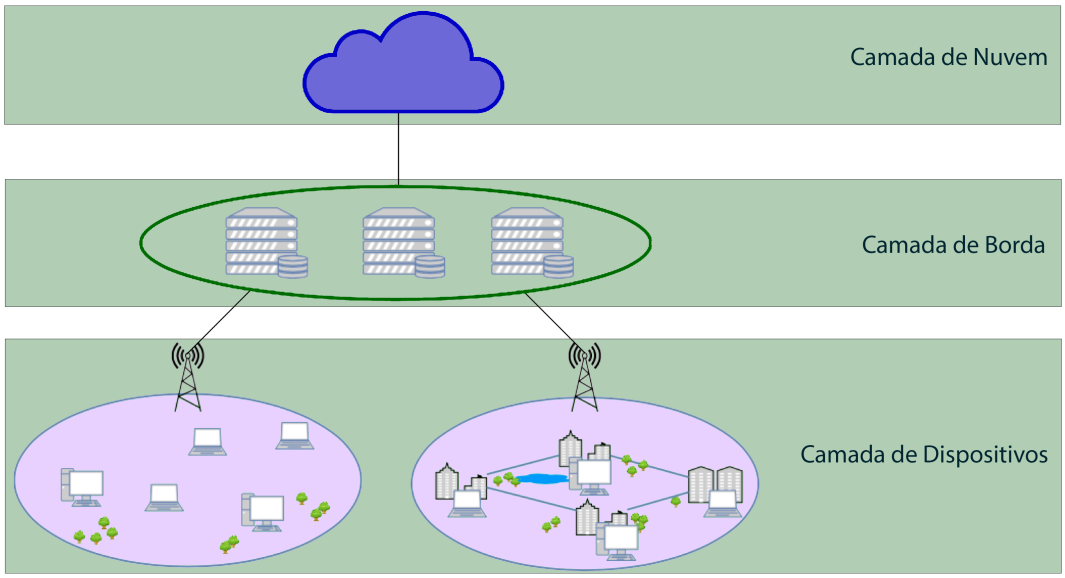


Figura 2.2: Posicionamento dos servidores de borda em meio a infraestrutura de rede

De fato, a computação de borda é compatível com várias tecnologias de rede sem fio, como 5G, Wi-Fi e LTE, e pode ser implementada em ambientes de nuvem pública, privada e híbrida [Kekki et al. 2018]. Além disso, a computação de borda apoia a RAN com uma capacidade de processamento e armazenamento, disponibilizando esses recursos na borda da rede para oferecer um ambiente de tecnologia da informação com latência baixa e alta largura de banda. Diferentemente de outros paradigmas que se baseiam em computação próximos ao usuário final, como a Computação em Névoa (*Fog Computing*)

e *Cloudlets*, a computação de borda é mais integrada à infraestrutura de rede, sendo, portanto, mais consciente da rede que os demais paradigmas. Na prática, isso significa que aplicações que executam na borda podem ter acesso em tempo real a informações de rádio e de rede, como informações de contexto, localidade e qualidade do canal de comunicação com o usuário móvel, permitindo que essas informações sejam usadas para diferenciar portfólios ou introduzir novos serviços [ETSI 2019].

O padrão *European Telecommunications Standards Institute* (ETSI) MEC [MEC 2019] tem sido identificado como a solução de um ambiente aberto e multiplataforma de computação de borda. Esse padrão especifica uma arquitetura de referência para uma plataforma de computação na borda e um rádio aberto e seguro. O ETSI MEC oferece interfaces abertas que facilitam o acesso e o compartilhamento desse ambiente por múltiplos provedores de serviço, bem como a fácil instalação e modificação de serviços.

No entanto, a implantação do padrão ETSI MEC também traz desafios, como a necessidade de garantir a segurança e privacidade dos dados na borda da rede, o gerenciamento de recursos em ambientes distribuídos e a compatibilidade com outras tecnologias de rede existentes. Para lidar com esses desafios, é importante que sejam adotadas soluções adequadas de segurança e gerenciamento de recursos, além de padrões e interfaces abertas para a interoperabilidade com outras tecnologias de rede.

2.2 *Network Slicing* no contexto da Inteligência Artificial

A tecnologia de NS tem ganhado destaque como uma proposta inovadora no atendimento de diferentes requisitos e exigências dos serviços em redes de comunicação. Para garantir a eficiência e o desempenho otimizado do NS, as técnicas de inteligência artificial (IA) têm desempenhado um papel fundamental. Neste texto, algumas das principais técnicas utilizadas em IA para resolver problemas relacionados ao NS são: [Wijethilaka e Liyanage 2021, Han e Schotten 2020]:

1. **Aprendizado de Máquina para Análise de Demanda de Serviço**: Uma das principais atribuições de NS é a alocação eficiente dos recursos de rede para atender à demanda de serviço. O aprendizado de máquina pode ser empregado para analisar padrões históricos de tráfego, comportamento do usuário e requisitos de largura de banda, a fim de prever a demanda futura de serviço. Isso permite uma alocação de recursos mais precisa e dinâmica, garantindo que os recursos sejam dimensionados adequadamente para cada fatia de rede.
2. **Otimização de Recursos com Algoritmos Genéticos**: Os algoritmos genéticos são uma técnica poderosa para otimização de problemas complexos. No contexto de NS, esses algoritmos podem ser aplicados para otimizar a alocação de recursos,

como largura de banda, capacidade de processamento e latência, levando em consideração as restrições e as necessidades específicas de cada fatia de rede. Os algoritmos genéticos exploram uma abordagem de seleção natural e evolução para encontrar soluções ótimas ou próximas do ótimo, contribuindo para uma alocação de recursos eficiente e adaptativa.

3. **Redes Neurais para Gerenciamento de SLA**: O gerenciamento efetivo de SLA é essencial para garantir a qualidade e o desempenho de *slices* na rede. Nesse sentido, as redes neurais podem ser empregadas para analisar dados em tempo real, monitorar o desempenho da rede e identificar potenciais violações de SLA. Essas redes podem aprender padrões complexos de tráfego e comportamento da rede, permitindo a tomada de decisões rápidas e precisas para garantir que os SLAs sejam cumpridos.
4. **Aprendizado por Reforço para Orquestração de Rede**: A orquestração eficiente de recursos é fundamental em NS, pois envolve a coordenação de diferentes elementos de rede para criar e gerenciar as fatias de rede. O aprendizado por reforço pode ser utilizado para aprender políticas de tomada de decisão adaptativas que otimizem a orquestração de recursos, levando em consideração as metas de desempenho e os requisitos de serviço. Essa abordagem permite a automação inteligente da orquestração, melhorando a eficiência operacional e a agilidade na gestão de NS.
5. **Algoritmos de Clusterização para Segmentação de Serviços**: A segmentação eficiente dos serviços é uma parte fundamental de NS, permitindo que diferentes *slices* de rede sejam configurados de maneira a atender aos requisitos específicos de cada tipo de serviço. Alguns exemplos como os algoritmos de clusterização k-means, podem ser aplicados para agrupar serviços semelhantes com base em suas características e requisitos. Isso facilita a alocação de recursos adequados e a configuração personalizada de cada *slice* de rede, garantindo que cada serviço seja atendido de maneira otimizada.

As técnicas de IA têm desempenhado um papel crucial na resolução de problemas relacionados a NS. Desde o uso de aprendizado de máquina para previsão de demanda até o emprego de algoritmos genéticos para otimização de recursos, e de redes neurais para gerenciamento de SLA até o uso de algoritmos de clusterização para segmentação de serviços, as abordagens de IA oferecem soluções eficientes para os desafios de NS [Chochliouros et al. 2020]. É importante observar que, as proposições discutidas contam com focos de aplicação em diferentes seguimentos de NS. Dessa maneira, considerando-se que esta tese propõe técnicas de controle de admissão de *slices*, adotamos a utilização de técnicas de aprendizado por reforço como base para a solução proposta.

2.3 Trabalhos Relacionados

Diante das complexidades inerentes à implementação de NS em sistemas de telecomunicações, é crucial que sejam adotadas soluções eficientes e eficazes. Além disso, é importante entender que o ciclo de vida de um *slice* compreende quatro etapas cruciais: i) preparação da rede, ii) instanciação do *slice*, iii) execução e iv) descomissionamento. É importante observar que o escopo a solução proposta nesta tese atua na etapa ii) da instanciação do *slice*, explorando os desafios e as soluções inovadoras no processo de configuração e criação de *slices* personalizados. A Figura 2.3 apresenta uma representação do ciclo de vida de um *slice*.

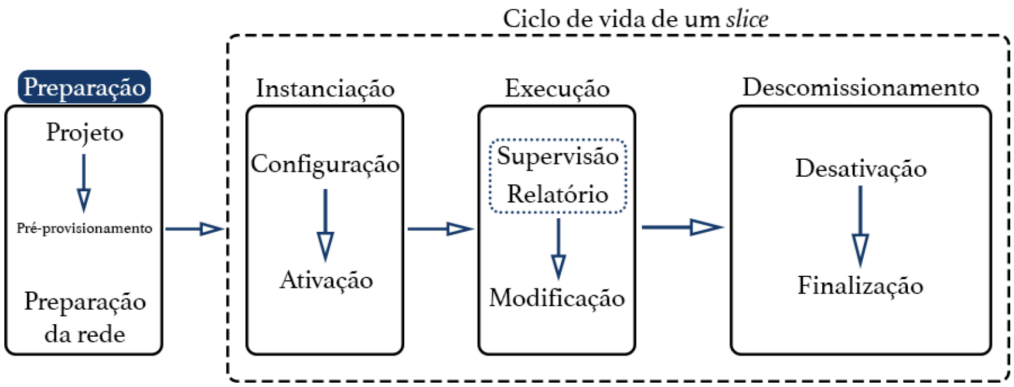


Figura 2.3: Ciclo de vida de um *slice*

Na primeira etapa, de preparação da rede, é necessário configurar e adaptar a infraestrutura de rede para suportar a criação e operação dos *slices*. A preparação da rede envolve a definição de políticas de alocação de recursos, como largura de banda, latência e capacidade de processamento. Além disso, é necessário configurar os elementos de rede, como roteadores, switches e controladores, para que possam acomodar os *slices* de maneira eficiente.

Na segunda etapa, de instanciação, um *slice* específico é criado para atender a requisitos de comunicação específicos. O objetivo é fornecer um serviço de rede personalizado, ajustado às necessidades dos usuários ou aplicativos atribuídos ao *slice*. Essa etapa envolve a especificação dos parâmetros do *slice*, como requisitos de largura de banda, qualidade de serviço (QoS) e políticas de segurança.

Na terceira etapa, de execução o *slice* é ativado e começa a fornecer serviços de rede conforme definido em seus parâmetros. Os usuários e aplicativos atribuídos ao *slice* podem utilizar a infraestrutura de rede alocada para suas necessidades específicas. Durante a execução, é fundamental monitorar o *slice* para garantir que esteja operando de acordo com as expectativas, avaliando seu desempenho e a utilização eficiente dos recursos alocados. Além disso, é nessa etapa que os princípios de elasticidade do *slice* podem ser executados.

Por fim, na quarta etapa, de descomissionamento, quando um *slice* não é mais necessário, seja devido a alterações nos requisitos de comunicação ou à conclusão de uma tarefa específica, ele pode ser desativado e removido da infraestrutura de rede. O descomissionamento envolve a liberação de recursos alocados para o *slice*, garantindo que sejam disponibilizados para outros usos.

Compreender o ciclo de vida de um *slice* é de extrema importância para posicionar e contextualizar o trabalho desenvolvido nesta tese de doutorado. O ciclo de vida do *slice* abrange todas as etapas necessárias para a criação, operação e desativação de fatias virtuais da rede, proporcionando serviços personalizados e isolados para diferentes usuários ou aplicativos. Ao entender as nuances e os desafios em cada etapa do ciclo de vida do *slice*, podemos identificar as lacunas existentes e propor soluções inovadoras e eficientes.

Ao considerar a instanciação do *slice* como o foco principal deste trabalho, é crucial entender as implicações dessa etapa específica no contexto mais amplo do ciclo de vida. A instanciação do *slice* determina como os recursos de rede serão alocados e configurados para atender aos requisitos específicos de comunicação. Uma instanciação adequada garante que os *slices* sejam criados de forma eficiente, levando em consideração a disponibilidade dos recursos e as demandas dos usuários ou aplicativos.

Dessa maneira, realizamos uma pesquisa acerca das soluções propostas nos últimos anos sobre a fase de instanciação de um *slice*, avaliando as características individuais, vantagens e desvantagens. A adoção de NS como uma forma de aumentar a rentabilidade financeira das operadoras de telecomunicações é uma tendência crescente, especialmente à medida que as redes 5G se consolidam. No entanto, podemos encontrar dois problemas clássicos abordados no arcabouço de soluções:

1. Controle de Admissão de *slices*: decidir, entre um conjunto de possíveis inquilinos da industria vertical, quem deve ser atendido, de maneira a maximizar a recompensa recebida pela operadora. De acordo com o ciclo de vida do *slices* definido pela NGMN, essa decisão deve ser feita na fase de comissionamento;
2. Orquestração de Recursos: uma vez que o conjunto de inquilinos a serem atendidos foi selecionado, criar, disponibilizar e monitorar *slices* de maneira a atender as solicitações dos inquilinos, respeitando as demandas implícitas à cada serviço. De acordo com o ciclo de vida do *slices* definido pela NGMN, a orquestração de recursos ocorre na fase de comissionamento e operação.

Dessa maneira, agrupamos as soluções nas duas categorias citadas, a fim de torná-las mais compreensíveis e facilmente aplicáveis às situações específicas de cada operadora.

2.3.1 Controle de Admissão de *Slices*

Com o advento de técnicas de virtualização e da divisão dos recursos de rede, as redes 5G, tem permitido que o compartilhamento da infraestrutura se torne ainda mais flexível, proporcionando um aumento significativo na multiplexação e no isolamento dos recursos [Foukas et al. 2017]. NS, permite que as operadoras criem redes virtuais sobre uma infraestrutura física, de maneira a compartilhar os recursos e, ao mesmo tempo, fornecer isolamento e segurança para os inquilinos e para os usuários do sistema [Zhang et al. 2017]. Dessa maneira, esse novo modelo de rede possibilita que as operadoras adaptem a infraestrutura da rede de maneira a atender às necessidades específicas de seus clientes, permitindo que eles forneçam serviços personalizados e otimizados para diferentes tipos de aplicativos e clientes. Com isso, as operadoras têm a oportunidade de aumentar a rentabilidade financeira, alugando parte da infraestrutura para inquilinos da indústria vertical, melhorando a experiência do usuário [Zhang 2019] [Li et al. 2017].

Em [Han et al. 2015], os autores abordam o problema de controle de admissão em uma rede com inquilinos heterogêneos. O sistema proposto é modelado com múltiplas filas, aproximando-se de um modelo Markoviano. Nele são impostas métricas de desempenho, como justiça entre diferentes locatários ou entre diferentes tipos de divisão da rede, fornecendo uma otimização baseada em utilidade. Os autores avaliam o desempenho da solução em comparação com soluções legadas.

Em [Jiang, Condoluci e Mahmoodi 2016], os autores propõem um novo mecanismo de controle de admissão baseado em heurística que explora dois níveis de prioridade: entre *slices* e intra *slices*. Uma vez que os recursos de rede são atribuídos a um inquilino, uma subdivisão é criada para atender cada usuário do inquilino de forma personalizada. De acordo com a decisão do controle de admissão, a tarefa de alocação de recursos é realizada com o objetivo de maximizar a qualidade da experiência dos usuários (QoE) dentro de cada *slice*, considerando a prioridade entre os *slice*.

Caballero et al. [Caballero et al. 2018] observaram que a existência de inquilinos com demandas elásticas e inquilinos com demandas não elásticas cria um contraste na alocação de recursos. Para solucionar esse problema, os autores propõem um modelo de controle de admissão de *slices* baseado em pesos atribuídos às solicitações, que descarta um ou mais inquilinos do sistema quando não houver recursos disponíveis. Os autores consideram apenas o segmento da RAN no trabalho. No entanto, embora o modelo proposto forneça uma boa visão geral do sistema, os autores utilizam técnicas de otimização. Essas, por sua vez, não escalam a solução à medida que a quantidade de parâmetros de entrada do problema aumenta, tornando-se impraticáveis numa implantação real. Isso se diferencia de nosso modelo, onde usamos técnicas de RL capazes de capturar a aleatoriedade inerente a esse tipo de problema, tornando a alocação de recursos mais eficiente e adaptável às mudanças nas demandas dos inquilinos.

No estudo de Samdanis et al. [Samdanis, Costa-Perez e Sciancalepore 2016], os autores discutem os principais habilitadores do compartilhamento de recursos em redes multi-inquilinos, baseados em demanda e sem intervenção humana, construídos sobre a arquitetura de gerenciamento e compartilhamento proposta pelo 3GPP. Eles propõem um esquema de controle de admissão de *slices* para o segmento da RAN, alocando recursos específicos à provedores de serviço e clientes da indústria vertical, considerando uma função de utilização do sistema por um período de tempo especificado. No entanto, eles não analisam a alocação de recursos de borda (i.e., computação e/ou armazenamento), que é igualmente importante e limitante para o compartilhamento de recursos em redes multi-inquilinos. A alocação eficiente de recursos de borda é crucial para garantir a latência, a largura de banda e a capacidade de processamento necessárias para atender às demandas das aplicações críticas em tempo real.

Por outro lado, os autores em Bakri [Bakri, Brik e Ksentini 2021] apresentam um mecanismo de controle de admissão que permite a alocação dinâmica de recursos em redes compostas por *slices* de características variadas. O objetivo é maximizar a satisfação dos usuários e garantir o cumprimento do SLA das aplicações. O esquema proposto realiza a tarefa de controle de admissão com eficiência, decidindo se deve aceitar ou rejeitar pedidos de *slices* com base nas taxas de dados e nos recursos disponíveis no sistema. Além disso, os recursos são alocados dinamicamente de acordo com a carga de tráfego da rede, o que possibilita uma melhor utilização dos recursos disponíveis. No entanto, os autores não consideram a possibilidade de os usuários terem demandas de recursos variáveis ao longo do tempo, o que pode levar a uma subutilização ou sobrecarga de recursos em determinados momentos.

Por fim, os autores Sciancalepore et al. [Sciancalepore et al. 2022] apresentam um *broker* de *slices* capaz de arbitrar e atender aos requisitos de *slices* heterogêneos dos inquilinos, maximizando os ganhos de multiplexação dos recursos da rede. A solução proposta é uma corretagem online de *slices* para o segmento da RAN, denominada ONETS, que utiliza o modelo do problema decisório clássico do Bandido Multi-Armado para analisar a disponibilidade de recursos de comunicação para alocação de inquilinos no sistema. O ONETS utiliza a quantidade de recursos, em termos de *Physical Resource Block* (PRBs) disponibilizados pela operadora, o tempo de duração da reserva e a quantidade de recursos efetivamente utilizada pelo inquilino para formar as decisões de aceite de *slices*. Por meio de leilão, o *broker online* analisa os recursos de rede da operadora, buscando a maximização dos ganhos da operadora ao alugar *slices* na infraestrutura proprietária. Os resultados apresentados demonstram a eficácia da solução, com a garantia de que os resultados sub-ótimos se aproximam dos resultados esperados no melhor caso. Contudo, os autores não consideram um recurso igualmente importante no conjunto de análise: o recurso de borda.

Neste trabalho, utilizamos técnicas de RL para permitir o controle de admissão de clientes na infraestrutura da operadora. Além disso, nossos modelos consideram a distribuição das solicitações através do tempo e incluem uma outra dimensão para os dados de entrada: os recursos computacionais de borda. Reconhecemos que, embora os recursos de comunicação sejam essenciais, eles não devem ser os únicos parâmetros analisados pelos tomadores de decisão. É crucial considerar os recursos de computação, pois uma análise limitada aos recursos de comunicação pode superestimá-los, resultando em uma solução inviável. Como os demais trabalhos relacionados, neste trabalho, nos focamos apenas no segmento da RAN. A Tabela 2.1 apresenta um resumo das características dos principais trabalhos que tocam o problema do controle de admissão de *slices*.

Tabela 2.1: Características e soluções adotadas nos principais trabalhos relacionados

Solução adotada	Caballero et al.	Samdanis et al.	Jiang et al.	ONETS	MONETS CAONS
Controle de Admissão	X	X	X	X	X
Aprendizado por Reforço		X	X	X	X
Decisão *on-the-fly*		X		X	X
Satisfação de usuários	X		X	X	X
Recursos de Comunicação	X	X	X	X	X
Recursos de Computação					X
Segmento da RAN	X	X	X	X	X

2.3.2 Orquestração de Recursos

À medida que *slices* são admitidos na rede através da utilização das técnicas de controle de admissão abordadas no capítulo anterior, surge uma outra classe de problemas - a orquestração de recursos de rede. Esta classe de problemas busca garantir o SLA das aplicações, considerando as demandas históricas geradas pela utilização do próprio *slice*, bem como investiga como os recursos serão redistribuídos dentro do *slice*. Embora não seja especificamente o foco desta tese, os problemas de orquestração de recursos estão intimamente ligados aos problemas de controle de admissão de *slices* e podem influenciar as decisões de admissão de inquilinos baseadas em previsões do futuro com base em dados históricos.

Nesse sentido, He e Song [He e Song 2015] propõem uma estrutura de otimização da utilização de recursos através de um controlador que considera a alocação ideal entre o conjunto de aplicativos analisados e a quantidade de recursos disponíveis. Em uma direção semelhante, Devlic et al. [Devlic et al. 2017] implementam o NESMO (*Network Slicing Management and Orchestration Framework*), um *framework* para automatização da implantação, configuração e gerenciamento de uma rede automatizada do tipo *end-to-end* (E2E).

Katsalis, Nikaein e Huang [Katsalis, Nikaein e Huang 2018] apresentam um novo orquestrador de recursos para uma rede virtualizada. Os autores consideram diferentes ofertas de serviços em diferentes segmentos de negócios, o que exige a configuração individual das funções de rede de acordo com as requisições de cada aplicação.

Afolabi et al. [Afolabi et al. 2020] propõem um sistema de orquestração multi-domínio, escalável e E2E. As iterações entre os componentes do orquestrador são modeladas com o auxílio da teoria de filas nas simulações apresentadas. O sistema proposto é composto por um algoritmo de escalonamento dinâmico, responsável por dimensionar os recursos do sistema ao mesmo tempo em que atende as solicitações de criação de novos *slices* com requisitos inerentes à aplicação.

Os trabalhos relacionados à orquestração de recursos em redes virtuais têm como objetivo garantir a qualidade de serviço das aplicações e melhorar a eficiência da utilização de recursos. Diversos estudos têm proposto estratégias para solucionar os problemas de orquestração em diferentes contextos, como redes virtuais multi-domínio, redes virtualizadas e redes E2E.

Os estudos revisados demonstram que a orquestração de recursos é um desafio complexo e multidimensional, que exige a consideração de diversos fatores, como demanda histórica, alocação de recursos, oferta de serviços e configuração individual de funções de rede. Além disso, a utilização de técnicas de otimização e controle, como os sistemas baseados em aprendizado de máquina, pode ser uma estratégia promissora para a solução dos problemas de orquestração de recursos.

Os avanços na orquestração de recursos têm permitido a criação de redes mais flexíveis, eficientes e adaptáveis às demandas das aplicações. Ainda assim, há desafios a serem superados, como a integração de diferentes domínios de rede e a garantia de que a orquestração de recursos seja realizada de forma justa e equitativa para todos os usuários. E apesar de não ser o foco central desta tese, a orquestração de recursos possui grande relevância para o problema de controle de admissão, podendo influenciar diretamente nas decisões de admissão de inquilinos. Assim, as soluções propostas pelos trabalhos relacionados ajudam a avançar na direção de uma abordagem mais abrangente e automatizada para o gerenciamento eficiente de recursos em ambientes de redes virtualizadas.

2.4 Consideraçãoes Finais

Neste capítulo, fornecemos uma definição sobre as redes 5G, a técnica de *Network Slicing* e a computação de borda. Nos concentramos em fornecer as definições mais utilizadas sobre as tecnologias citadas, seguindo os padrões de referência. Discutimos os principais trabalhos relacionados ao escopo de desenvolvimento utilizado nesta tese. Apesar de utilizarem, em alguns casos, técnicas de RL para tratar o problema de controle de admissão, os trabalhos relacionados não inserem dados de aplicações do mundo real em suas análises, conduzindo o conjunto de testes com valores arbitrários. Além disso, os trabalhos relacionados não consideram uma dimensão igualmente importante na decisão - os recursos de computação. No Capítulo 3 estudamos o problema clássico de admissão de *slices* como um arcabouço de algoritmos do tipo MaB. Posteriormente, no Capítulo 4 tratamos as lacunas identificadas nos trabalhos relacionados, formulando um problema de controle de admissão que utiliza dados de inquilinos do mundo real, ao passo que considera recursos tanto recursos de comunicação, quanto de computação no conjunto da análise.

CAPÍTULO 3

Controle de Admissão de *Slices* e Arcabouço do Bandido Multi-Armado

Nesta seção, estudamos inicialmente o problema de controle de admissão de *slices* no segmento da RAN sob o arcabouço do Bandido Multi-Armado (MaB). Neste capítulo, levamos em consideração apenas recursos de comunicação. Estudamos três soluções disponíveis na literatura para tratar esse problema, a saber: ϵ-Greedy, eUCB e ONETS. As duas primeiras correspondem à soluções clássicas utilizadas no arcabouço MaB, enquanto a última foi proposta especificamente para o problema no contexto de NS.

3.1 Modelo do Sistema

Neste trabalho, consideremos que um provedor de serviços de telecomunicações (operador) torna sua infraestrutura disponível para clientes externos (inquilinos), como por exemplo, clientes da indústria vertical. O operador é o proprietário da infraestrutura da rede. Consideramos apenas o segmento da RAN, o qual contém recursos de rádio, provido por uma estação base. Assumimos que a estação base oferece suporte para fatiamento de recursos (*network slices*) e possui uma capacidade total C dada em números de PRBs.

Consideramos $\mathcal{I} = \{1, 2, \cdots, |\mathcal{I}|\}$ um conjunto de inquilinos que se inscrevem previamente junto ao operador antes de submeter uma requisição para criação de *slice*. Seja $\mathcal{S} = \{1, 2, \cdots, |\mathcal{S}|\}$ o conjunto de modelos (*templates*) de *slices* disponibilizados pelo operador. Cada inquilino $i \in \mathcal{I}$ pode requisitar um *slice* $s \in \mathcal{S}$ que melhor represente as características de seu serviço. Cada *template* de *slice* $s = \{R^{(s)}; L^{(s)}\}$ compreende uma quantidade de recursos de radio (dado em número de PRBss), representada por $R^{(s)}$, e o tempo de duração do *slice* (expresso em segundos), representado por $L^{(s)}$.

Seja $\mathcal{T} = \{1, 2, \cdots, |\mathcal{T}|\}$ o conjunto de instantes de tempo em que um mecanismo de controle de admissão de *slices* toma decisão sobre quais requisições aceitar. Cada inquilino $i \in \mathcal{I}$ pode solicitar um *slice* $s \in \mathcal{S}$ em um instante de tempo $t \in \mathcal{T}$. Denotamos tal requisição por $r_{i,t}^{(s)}$. Modelamos a distribuição das requisições no tempo de um inquilino $i \in \mathcal{I}$ como uma variável aleatória, de forma que o tempo entre as chegadas, representado por Δ_t, é distribuído exponencialmente com taxa ϕ_i. Obtemos ϕ_i de uma distribuição de Pareto, com média ρ e desvio padrão ζ.

Consideramos que um inquilino $i \in \mathcal{I}$ oferece um único tipo de serviço, por exemplo: jogos online, vídeo por *streaming*, geolocalização e mapas, etc.. Existem diferentes *templates* de *slices* para o mesmo serviço. Tais *templates* diferem entre si pela quantidade de recursos alocados, a qual pode variar entre máxima, média e mínima. Um inquilino $i \in \mathcal{I}$ sempre requisita um *slice* para o mesmo tipo de serviço. No entanto, a quantidade de recursos requisitados pode variar a cada requisição. Assumimos também que cada inquilino pode solicitar um único *slice* a cada instante de tempo t, ou seja, novas solicitações de *slices* não podem ser aceitas enquanto $L^{(s)}$ do inquilino i estiver em vigor.

Definimos o problema do controle de admissão de *slices* da seguinte forma: A cada instante de tempo t, a operadora decide se aceita ou rejeita, de forma online, um conjunto de requisições para criação de novos *slices*. O objetivo é maximizar a multiplexação dos *slices*, e ao mesmo tempo, cumprir as garantias acordadas (SLAs) em *slices* concedidos anteriormente e ainda em vigor (ativos).

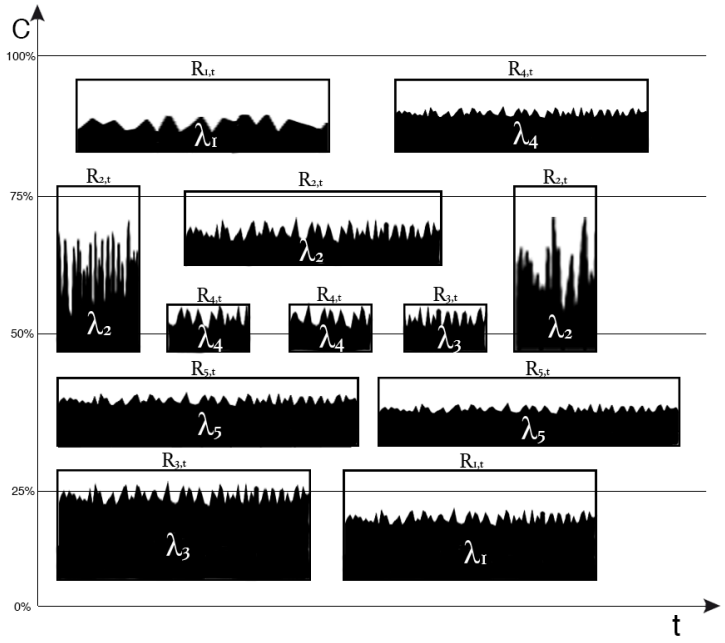


Figura 3.1: Esquema de política de seleção.

A Figura 3.1 ilustra um exemplo da política de seleção. A política de seleção utilizada decide conceder ao inquilino selecionado a quantidade de recursos de rede necessários ao tipo de *slice* acordado, ou seja $R_{i,t}^{(s)}$. É possível que um inquilino seja selecionado, mas não tenha manifestado interesse por locar *slices* na infraestrutura da operadora, nesse caso, não será retornada recompensa pela ação. Definimos a função de custo $\lambda_{i,t}$ como o número de PRBs efetivamente utilizados na rodada t pelo inquilino i, de forma que:

$$\lambda_{i,t} \leq R_{i,t}^{(s)} \leq C \tag{3-1}$$

O orçamento total para admissão de *slices*, ou seja, C, representa a capacidade total de transmissão de dados na rede. Esse orçamento define a região de admissibilidade, já que não podem ser alocados mais *slices* do que a capacidade C suporta.

Ao aceitar a requisição para criação de um *slice* para um inquilino i no instante de tempo t, o operador recebe recompensas pela utilização dos recursos de rede. A Equação 3-2 expressa a recompensa recebida, em termos de recursos de rede, por aceitar uma requisição do inquilino i no tempo t.

$$\eta_{i,t} = \alpha \frac{R_{i,t}^{(s)}}{C} + (1-\alpha)\frac{R_{i,t}^{(s)} - \lambda_{i,t}}{R_{i,t}^{(s)}}, \tag{3-2}$$

Particularmente, a recompensa $\eta_{i,t}$ leva em consideração a quantidade de recursos solicitada, bem como o ganho de multiplexação, ou seja, a proporção entre o que foi realmente usado (segundo termo da Equação 3-2) e o que está sendo solicitado (primeiro termo da Equação 3-2). De um lado, inquilinos cujos recursos atribuídos passam a maior parte do tempo subutilizados são preferíveis àqueles que utilizam completamente seus recursos. Isso ocorre porque o operador pode realocar os recursos não utilizados naquele momento pelo inquilino para atender mais requisições no sistema, empreendendo uma estratégia de *overbooking*. Por outro lado, inquilinos que requisitam (e pagam) uma quantidade maior de recursos também são de interesse do operador. O peso $\alpha \in [0,1]$ na Equação 3-2 expressa esse compromisso. A Tabela 3.1 apresenta um resumo das variáveis utilizadas em nosso modelo.

Tabela 3.1: Variáveis utilizadas em nosso modelo.

$\mathcal{I}$	Conjunto de inquilinos
$\mathcal{T}$	Conjunto de estampas de tempo onde são feitas decisões
C	Capacidade total disponível para transmitir dados na rede
$\eta_{i,t}$	Valor de recompensa recebida, em termos de recursos de rede, pela aceitação da requisição do inquilino i no instante de tempo t
$R_{i,t}^{(s)}$	Quantidade de recursos de rede disponibilizada para o inquilino i no instante de tempo t
$\lambda_{i,t}$	Quantidade de recursos de rede efetivamente utilizada pelo inquilino i no instante de tempo t
α	Peso atribuído à quantidade de recursos de rede solicitada

3.2 Formulação do Problema

Para todo inquilino $i \in \mathcal{I}$ e para todo instante de tempo $t \in \mathcal{T}$, definimos a seguinte variável de decisão:

- $x_{i,t} \in \{0,1\}$, a qual assume o valor 1, quando o a requisição de criação de *slice* do inquilino i é aceita em t, e o valor 0, caso contrário.

Podemos, então, formular a admissão de *slices* como um problema dado por:

$$\textit{maximizar} \quad \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{I}} \eta_{i,t} x_{i,t} \tag{3-3}$$

sujeito a:

$$\sum_{i \in \mathcal{I}} \lambda_{i,t} x_{i,t} \leq C, \forall t \in \mathcal{T} \tag{3-4}$$

$$x_{i,t} \geq x_{i,t-1} \mathbb{F}(t - t_i^{start} \leq L_i^{(s)}) \tag{3-5}$$

$$x_{i,t} \in \{0,1\}, \forall i \in \mathcal{I}, \forall t \in \mathcal{T} \tag{3-6}$$

A função objetivo 3-3 incentiva a aceitação das requisições. A restrição em 3-4 garante que a quantidade de PRBs utilizada não exceda a capacidade. Em 3-5, impomos a seleção do inquilino i enquanto durar a reserva do *slice* s previamente concedido. Denotamos por t_i^{start} a rodada em que a requisição s do inquilino i foi aceita, e $\mathbb{F}$ é uma função indicadora que fornece o valor 1, se a condição for satisfeita, e o valor 0, caso contrário. Por fim, em 3-6 estabelecemos a variável binária de decisão.

3.3 Bandido Multi-Armado

RL é uma técnica de aprendizado de máquina composta por um agente que aprende quais ações tomar em um ambiente, de maneira a maximizar uma recompensa ao longo do tempo. A aplicação de RL em NS é vasta, tocando objetivos como otimizar a alocação de recursos de rede para atender às diferentes demandas de *slices*, ou realizar controle de admissão de inquilinos no sistema.

Para aplicar RL em NS, é necessário definir um ambiente, um agente e uma função de recompensa. O ambiente é caracterizado pela rede virtualizada, em que as fatias de rede são criadas. O agente é responsável por tomar as ações, neste caso, decisões sobre aceitar ou rejeitar solicitações de *slices* de rede. Por fim, a função de recompensa é o objetivo que o agente deve maximizar, que pode ser definido em termos de desempenho de rede, satisfação do usuário, custo, entre outros.

O agente utiliza uma política para selecionar as ações com base no estado atual do ambiente. Em soluções clássicas, o estado inclui informações sobre a demanda de tráfego, a disponibilidade de recursos de rede e as características de cada *slices* de rede. A política é treinada usando RL, possibilitando que o agente aprenda a tomar as ações que maximizam a recompensa acumulada ao longo do tempo.

O treinamento do agente pode ser feito por meio de simulações em um ambiente controlado, ou pode ser feito em um ambiente real, onde o agente pode interagir com a rede em tempo real. Durante o treinamento, o agente explora diferentes ações para aprender qual é a melhor para cada estado do ambiente. Com o tempo, o agente é levado a convergir para uma política ótima que maximiza a recompensa acumulada. Após o treinamento, o agente pode ser implantado na rede real para alocar recursos de rede para as diferentes fatias de rede de forma dinâmica e automática. O agente pode monitorar continuamente o ambiente para ajustar suas ações com base nas mudanças nas demandas de tráfego ou na disponibilidade de recursos.

Desse modo, RL quando aplicado em NS pode contribuir na otimização e na alocação de recursos de rede, de maneira a atender às demandas de diferentes *slices*. O agente aprende a tomar ações por meio de treinamento em um ambiente controlado, em tempo real e convergir para uma política ótima que maximiza a recompensa. O RL pode aumentar a satisfação do usuário, melhorar o desempenho da rede e reduzir o custo.

Nessa vertente, uma das soluções de RL frequentemente utilizadas para problemas de alocação de recursos e controle de admissão em *slices* são os algoritmos Bandido Multi-armado (MaB). Em *Network Slicing*, um MaB pode ser usado para determinar qual *slice* de rede deve receber mais recursos em um determinado momento para maximizar a satisfação do usuário ou outra métrica de desempenho.

No problema exploratório MaB clássico, um jogador tem diversas máquinas caça-níquel à disposição. Cada máquina caça-níquel retorna uma recompensa obtida através de uma distribuição de probabilidade desconhecida. O jogador pode escolher uma máquina por vez. Em cada decisão, o jogador pode: (1) explorar e observar a recompensa retornada por uma máquina caça-níquel ainda não escolhida ou (2) aproveitar mantendo-se com a máquina caça-níquel que retornou a maior recompensa já vista em rodadas anteriores. O objetivo final é maximizar a recompensa recebida após um certo número finito de rodadas.

A complexidade de um problema estocástico de exploração MaB é determinada pelo número total de máquinas caça-níquel, em conjunto com a diferença entre a recompensa esperada e a recompensa ideal[Mahajan e Teneketzis 2008]. Estratégias como o eUCB são frequentemente utilizados na literatura para resolver o problema MaB sem garantia da recompensa ideal [Black 2005] [Ontanón 2013] [Guan et al. 2020].

Sciancalepore et al. [Sciancalepore et al. 2022] mostram que o controle de admissão de *slices*, modelada como um problema de decisão, pode ser mapeada para um problema do tipo MaB com algumas variações: (i) múltiplos jogadores, (ii) orçamento limitado e (iii) períodos de *lock-up*, devido à restrição imposta em 3-5. Cada inquilino i é modelado como uma máquina caça-níquel, a qual, se escolhida no tempo t, retorna uma recompensa. Como mais de um inquilino pode ter uma requisição pendente no tempo t, o jogador pode escolher múltiplos inquilinos ao mesmo tempo, desde que respeitando o orçamento disponível.

O problema do bandido multi-armado com restrição de *lock-up* é uma variação do problema do bandido multi-armado, em que o agente enfrenta uma restrição adicional em suas escolhas. A restrição determina que uma vez que um braço é escolhido, ele deve ser mantido fixo por um número fixo de tentativas antes que o agente possa mudar para outro braço. Matematicamente, o problema do bandido multi-armado com restrição de *lock-up* pode ser formulado como segue:

Suponha que existam k braços (ou opções) disponíveis para o agente, numerados de 1 a k. Cada braço i tem uma distribuição de probabilidade desconhecida P_i sobre as recompensas r_i que podem ser obtidas ao escolher este braço. O objetivo do agente é maximizar a recompensa total R_T obtida após T tentativas, onde T é um número que determina o tamanho da janela de tomada de decisões. Durante o processo de escolha, o agente enfrenta uma restrição de *lock-up*: uma vez que um braço i_t for escolhido na tentativa t, ele deve ser mantido fixo durante as próximas L tentativas, antes que o agente possa mudar para outro braço. Formalmente, a recompensa R_T do agente é dada por:

$$R_T = \sum_{t=1}^{T} r_{i_t,t} \tag{3-7}$$

onde, i_t é o braço escolhido pelo agente na tentativa t, e $r_{i_t,t}$ é a recompensa obtida pelo agente ao escolher o braço i_t na tentativa t.

O problema para o agente é encontrar uma política de seleção de braços que maximize a recompensa esperada, considerando a restrição de *lock-up*. O agente explora os braços para obter informações sobre as distribuições de probabilidade e, ao mesmo tempo, considera as restrições de *lock-up* em suas escolhas. Uma possível abordagem é utilizar um algoritmo de RL adaptado para a restrição de *lock-up*.

No algoritmo adaptado de RL, assim como em MaB com restrição de *lock-up*, o agente enfrenta a restrição adicional de que uma vez que uma ação é tomada, ela deve ser mantida fixa por um número fixo de etapas de tempo antes que o agente possa mudar para outra ação. Dessa maneira, esse algoritmo pode ser adaptados para fornecer um caminho de solução para o problema abordado nesta tese, como demonstramos no Capítulo 4.

3.3.1 Algoritmo Enhanced-UCB (eUCB)

Enhanced Upper Bound Confidence (eUCB) é um algoritmo do conjunto de soluções de aprendizado por reforço, que pode ser adaptado para otimizar o desempenho de redes no contexto de NS. O objetivo do eUCB é alocar recursos, de maneira eficiente, maximizando o desempenho geral da rede. Ao invés de realizar escolhas arbitrárias sobre qual braço da máquina caça-níquel deve ser puxado em cada rodada, o algoritmo eUCB baseia-se no equilíbrio entre exploração e aproveitamento para tomar decisões, guiados por um termo de confiança. Esse termo garante que o *slice* selecionado possua o maior potencial de melhorar o desempenho geral da rede, levando em consideração o desempenho passado de outros *slices*. Para isso, o eUCB utiliza uma fórmula que combina a média da recompensa histórica recebida, com o termo de confiança acumulado.

Uma vez selecionado o *slice* a ser aceito, o eUCB determina a quantidade de recursos que devem ser alocados para o *slice* em questão. Essa alocação é realizada de forma eficiente, levando em consideração tanto o desempenho passado do *slice* quanto a quantidade de recursos disponíveis na rede. A alocação de recursos é feita por meio de um processo Markoviano que considera a recompensa histórica e a quantidade de recursos alocados para demais *slices*. À medida que reúne conhecimento do ambiente, o algoritmo deixa de ser exploratório, e passa a intensificar a solução preferindo a ação com maior recompensa estimada.

O eUCB é uma abordagem adaptativa e dinâmica, o que significa que ele é capaz de ajustar as alocações de recursos à medida que o desempenho dos *slices* mudam ao longo do tempo. Essa abordagem torna o eUCB uma solução viável em diferentes ambientes de rede, como redes de acesso sem fio e redes de data center.

O modelo clássico proposto em [Garivier e Cappé 2011] serve como referência para solução. Durante o processo de execução, o algoritmo clássico coleta a recompensa obtida em cada máquina caça-níquel escolhida durante cada tentativa, e infere a média da distribuição estatística $\bar{\theta}_i$. A função de densidade é parametrizada pela média, ou seja, $\mu(\theta_i) = \bar{\theta}_i$, desse modo, quanto maior o número de tentativas, maior a precisão nas informações da distribuição. Além disso, o algoritmo insere um peso G_{alg} ($\sqrt{\frac{2\ log\ t}{W_i}}$) que atenua a influência negativa das decisões aleatórias. O G_{alg} trabalha dando preferências de seleção à inquilinos selecionados mais vezes em rodadas anteriores, e que podem retornar maiores recompensas na relação de alocação/utilização de recursos junto a operadora.

No problema formulado na Seção 3.1, várias máquinas caça-níquel podem ser selecionadas, desde que obedecendo a restrição de orçamento C. Além disso, se uma máquina caça-níquel foi selecionado em uma rodada anterior, e o período de reserva ainda estiver em vigência, o algoritmo deve obrigatoriamente selecionar essa máquina caça-níquel para a próxima rodada. Uma versão do algoritmo eUCB para o problema de admissão de *slices* é descrita no Algoritmo 3.1.

Algoritmo 3.1: Algoritmo eUCB com garantia de orçamento fixo.

Entrada: I, T, C.

1 **Inicialização:** $W_i = 0$, $\bar{\theta}_i(0) = 0 \in \Theta$; $\forall$ i $\in$ I, L $\leftarrow 0$
2 **forall** $i \in I$ **do**
3 GET v_i
4 UPDATE $\bar{\theta}_i(0)$
5 $W_i = W_i + 1$
6 **end**
7 **forall** $t \in T$ **do**
8 $\hat{\theta}_i(t) = \bar{\theta}_i(t) + \sqrt{\frac{2\ log\ t}{W_i}}$
9 $R = \{i\} \leftarrow$ **Problema** D-ONLINE-SLICING(C, $L(t)$, $\Theta(t)$)
10 **forall** $i \in R$ **do**
11 GET v_i
12 UPDATE $\bar{\theta}_i(t)$
13 $W_i = W_i + 1$
14 **end**
15 UPDATE $L(t)$
16 **end**

As linhas de 1-6 inicializam o algoritmo. Nessa etapa inquilinos podem expressar seus interesses por *slices*. A linha 9 resolve de forma otimizada uma versão instantânea do problema (D-ONLINE-SLICING), assumindo como entradas o orçamento total C, os períodos de reserva em vigência ($L(t)$) e as médias empíricas das distribuições de recompensa dos inquilinos. A versão instantânea desse problema reduz significativamente a complexidade do problema, em comparação ao modelo ótimo apresentado na Seção 3.2, pois persegue apenas a maximização da recompensa pontual em t. A saída do problema é constituída por um conjunto de inquilinos. Nas linhas de 10 - 14, as recompensas, obtidas com os inquilinos escolhidos na linha 8, são atualizadas. Na linha 15, os períodos de reserva atuais são atualizados para as próximas iterações.

3.3.2 Algoritmo ϵ-Greedy

No problema de controle de admissão de slices, O algoritmo ϵ-Greedy tem o objetivo de selecionar o inquilino do sistema para receber *slices* de rede de maneira a garantir, ao mesmo tempo, que os outros inquilinos recebam recursos suficientes para manter um desempenho satisfatório. O algoritmo funciona em duas etapas: i) seleção de *slices* e ii) alocação de recursos.

Algoritmo 3.2: ϵ-Greedy - Algoritmo de seleção do conjunto de inquilinos atendidos, com base na probabilidade de seleção ϵ.

Entrada: I, T, C.
Inicialização: $W_i = 0$, $\bar{\theta}_i(0) = 0 \in \Theta$; $\forall$ i $\in$ I, L $\leftarrow 0$
forall $i \in I$ **do**
    GET v_i
    UPDATE $\bar{\theta}_i(0)$
    $W_i = W_i + 1$
end
forall $t \in T$ **do**
    $\epsilon = \min\{ 1 , \frac{b|I|}{d^2 t} \}$
    while $(C - B \leq 0)$ ***and*** $(I \neq \emptyset)$ **do**
        if $L(t) \neq \emptyset$ **then**
            $\hat{I} \leftarrow L$
        end
        else
            GET $z \in [0,1]$
            if $z \leq \epsilon$ **then**
                $\hat{I}$: arg max $\hat{\theta}_i(t)$
            end
            else
                $\hat{I}$: rand (I/L)
            end
        end
        if $B + \lambda_i \leq C$ **then**
            $R \leftarrow R \cup \hat{I}$
            $B = B + \lambda_i$
        end
    end
    forall $i \in R$ **do**
        GET v_i
        UPDATE $\bar{\theta}_i(t)$
        $W_i = W_i + 1$
    end
    UPDATE $L(t)$
    $B = 0$, $Z = 0$
end

Na primeira etapa, o ϵ-Greedy seleciona o inquilino que receberá *slices* na rede. A seleção do inquilino é feita de acordo com a probabilidade de ser escolhido aleatoriamente ou selecionado com base no retorno passado do inquilino. A probabilidade de escolha aleatória é controlada por um parâmetro "epsilon"(ϵ , enquanto a escolha, com base no desempenho passado, é determinada pelo algoritmo que acompanha o retorno dos inquilinos do sistema.

Se o inquilino selecionado for aquele com o melhor retorno passado, então o algoritmo alocará recursos para esse inquilino. Caso contrário, a política selecionará outro inquilino para receber *slices* no sistema. Esse processo de seleção e alocação é repetido várias vezes, ajustando a alocação de recursos no sistema à medida que o desempenho muda. O pseudo-código é apresentado no Algoritmo 3.2.

Uma dependência linear de ϵ com o tempo decorrido t guia a política de seleção a explorar com mais intensidade durante as primeiras rodadas, enquanto intensifica o aproveitamento ao longo da evolução das rodadas. Define-se $\epsilon = \frac{b|I|}{d^2 t}$ onde $d \in 0{,}1$ e b são valores arbitrários. O Algoritmo 3.2 selecionará, com probabilidade ϵ, as melhores máquinas caça-níquel em busca da maximização da recompensa (linha 16), enquanto selecionará aleatoriamente (linha 19), satisfazendo (em ambos os casos) a restrição orçamentária limitante (linhas 22-26).

3.3.3 Algoritmo ONETS

A complexidade computacional é uma preocupação fundamental em muitos campos da ciência da computação, especialmente em problemas de otimização e aprendizado de máquina. Embora o algoritmo eUCB atinja bom desempenho na solução para admissão de *slices*, ele resolve um problema de Programação Linear Inteira a cada rodada. Na prática, embora seja amplamente utilizado, a complexidade computacional do eUCB é exponencial, sendo um obstáculo para utilização em sistemas de tempo real. Além disso, Tran-Thanh et al. [Tran-Thanh et al. 2012] provaram que o eUCB quando aplicado à problemas do tipo MaB com restrição de orçamento são NP-Hard.

Uma forma de reduzir a complexidade do algoritmo eUCB é substituir a resolução do problema D-ONLINE-SLICING, a cada rodada, pela seleção de K máquinas caça-níquel, repeitando a restrição de orçamento e de forma que a média da distribuição empírica seja maximizada. O trabalho de [Sciancalepore et al. 2022], denominado ONETS propõe uma solução para essa questão.

O ONETS é um sistema que reduz a complexidade computacional do eUCB, tornando-o adequado para aplicações em tempo real em sistemas de redes. O sistema utiliza uma abordagem de seleção de braço com base em aprendizado por reforço, que permite a seleção de uma ação ótima, com base em experiências anteriores e sem exigir uma análise completa de todas as opções disponíveis.

Algoritmo 3.3: ONETS - Algoritmo de seleção do conjunto de inquilinos atendidos, garantindo orçamento fixo.

Entrada: K, I, T, C.
Inicialização: $B = 0$, $Z = 0$, $L \leftarrow \emptyset$, $W_i = 0$, $\bar{\theta}_i(0) = 0 \; \forall i \in I$
forall $i \in I$ **do**
    GET v_i
    UPDATE $\bar{\theta}_i(0)$
    $W_i = W_i + 1$
end
forall $t \in T$ **do**
    $\hat{\theta}_i(t) = \bar{\theta}_i(t) + \sqrt{\frac{2 \; log \; t}{W_i}}$
    while $n \leq K$ **do**
        if $L(t) \neq \emptyset$ **then**
            $\hat{i} \leftarrow L$
        end
        else
            $\hat{i}$: arg max $\hat{\theta}_i(t)$
        end
        if $B + \lambda_i \leq C$ **then**
            $R \leftarrow R \cup \hat{i}$
            $B = B + \lambda_i$
            $n = n + 1$
        end
    end
    forall $i \in R$ **do**
        GET v_i
        UPDATE $\bar{\theta}_i(t)$
        $W_i = W_i + 1$
    end
    UPDATE $L(t)$
    $B = 0; Z = 0, n = 0$
end

Com o ONETS, o processo de seleção de braço é aprimorado, permitindo que o sistema encontre a solução ideal de maneira mais rápida e eficiente. A limitação da análise em exatamente K máquinas selecionadas reduz a complexidade da solução em $\mathcal{O}(K)$. Uma versão do algoritmo ONETS para o problema de admissão de *slices* é descrita no Algoritmo 3.3.

O algoritmo executa inicialmente uma rodada de inicialização, onde os inquilinos apresentam o interesse por utilização de *slices* na infraestrutura da operadora (linhas 2-5). O parâmetro $\bar{\theta}_i$ (linha 8) acumula a média história das recompensas acumuladas pelo inquilino e servem como norteador para o aprendizado do algoritmo. A cada rodada, o algoritmo ONETS seleciona os inquilinos que já estão com uma reserva vigente, respeitando a restrição de *lock-up* (linhas 10-12). Após essa operação, se a quantidade de recursos já alocada for menor que os orçamentos disponíveis, escolhe-se, dentre os demais inquilinos, aqueles que contém os maiores valores para $\hat{\theta}_i(t)$ e que ainda respeitem a restrição de orçamento (linhas 13-21). Por fim, os valores de recompensa e quantidade atendimento são computados (linhas 22-25), a fila de *lockup* é atualizada (linha 27) e as variáveis de controle reinicializadas (linha 28). O algoritmo encerra ao fim do período de simulação T e retorna uma lista de inquilinos a serem atendidos em cada instante de tempo $t \in T$.

3.4 Considerações Finais

Neste capítulo, apresentamos o problema de controle de admissão de inquilinos na infraestrutura da operadora. Abordamos o problema como um *Multi-armed bandit* e estudamos soluções clássicas disponíveis na literatura, os algoritmos eUCB, ϵ-Greedy e ONETS. No Capítulo 5, apresentamos três evoluções do algoritmo ONETS para tratar o problema de controle de admissão de *slices* considerando recursos de comuniação e computação de borda.

CAPÍTULO 4

Controle de Admissão para *slices* Ciente de Recursos de Rede e de Processamento

Neste capítulo, inicialmente apresentamos o modelo de sistema utilizado neste trabalho. Posteriormente, descrevemos o problema de controle de admissão para NS que considera recursos de comunicação e computação. Por fim, apresentamos uma solução exata e soluções baseadas em aprendizado por reforço para o problema em questão.

4.1 Modelo do Sistema

Neste trabalho, propomos um modelo de sistema que leva em consideração as características das soluções apresentadas no Capítulo 3, onde o operador de serviços de telecomunicações torna sua infraestrutura disponível para inquilinos. Contudo, ao tratar do segmento da RAN, além de considerar os recursos de rádio, providos por uma estação base, consideramos também os recursos de processamento, provido por um servidor de borda. Além disso, consideramos o acúmulo histórico das recompensas recebidas, utilizamos fatores de reserva visando minimizar o número de violações na rede e ainda, inserimos a penalização no sistema diante de escolhas que incorrem quantidades elevadas de violações. Dessa maneira, cada estação base além possuir uma capacidade total de C em PRBs de comunicação, conta ainda com uma capacidade total M de computação em servidores de borda, expressa em CPU *ticks*.

O conjunto de inquilinos $\mathcal{I} = \{1, 2, \cdots, |\mathcal{I}|\}$, o tipo de *slice* $\mathcal{S} = \{1, 2, \cdots, |\mathcal{S}|\}$, a janela de tempo sobre a qual o mecanismo de controle de admissão toma decisões, $\mathcal{T} = \{1, 2, \cdots, |\mathcal{T}|\}$, e o modelo de distribuição de solicitações de *slices* são modelados e representados de forma semelhante ao apresentado no Capítulo 3. Em relação ao *template* de *slice* $s \in \mathcal{S}$, além da quantidade de PRBs necessária para criar um *slice* do tipo s ($R^{(s)}$) e a duração do *slice* ($L^{(s)}$), representamos também a quantidade de recursos de computação de borda necessária para criar um *slice* do tipo s, representada por $Q^{(s)}$, ou seja, $s = \{R^{(s)}; Q^{(s)}; L^{(s)}\}$. Consideramos ainda que inquilino $i \in \mathcal{I}$ oferece um único tipo de serviço entre: (i) jogos online, (ii) vídeo por *streaming*, (iii) geolocalização e mapas.

Dessa forma, definimos o problema do controle de admissão da seguinte forma: a cada instante de tempo t, a operadora decide se aceita ou rejeita, de forma online, um conjunto de requisições para criação de novos *slices*, considerando três dimensões: (i) recursos de comunicação, (ii) recursos de computação e (iii) duração do *slice*. O objetivo é maximizar a multiplexação dos *slices* e, ao mesmo tempo, cumprir os níveis de serviços (SLAs) acordados em *slices* já instanciados e ativos. A Figura 4.1 ilustra um exemplo da política de seleção que decide conceder ao inquilino selecionado a quantidade de recursos, tanto de rede, quanto de computação, necessários ao tipo de *slice* acordado. Na figura, as três dimensões consideradas, variam de acordo com as características do *slice*. A dimensão de largura define por quanto tempo um *slice* deve ficar ativo quando for aceito. A dimensão de altura define a quantidade de recursos de comunicação necessários para o estabelecimento do *slice*. E, por fim, a dimensão de profundidade, define a quantidade de recursos de processamento necessária para atendimento da aplicação.

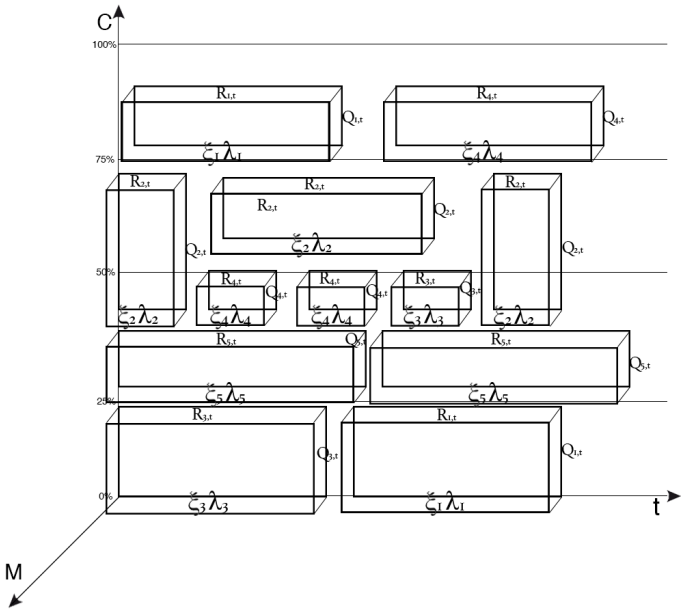


Figura 4.1: Esquema de política de seleção.

Em nosso modelo, definimos as funções de custo $\lambda_{i,t}$ e $\xi_{i,t}$, respectivamente, como o número de PRBs e a quantidade de CPU *ticks* utilizados em t pelo inquilino i, de forma que:

$$\lambda_{i,t} \leq R_{i,t}^{(s)} \leq C \tag{4-1}$$

$$\xi_{i,t} \leq Q_{i,t}^{(s)} \leq M \tag{4-2}$$

Definimos ainda a Equação 4-3, que expressa a recompensa recebida, em função da quantidade de recursos de rede solicitada e a efetivamente utilizada no tempo t. De maneira similar, a Equação 4-4 expressa a recompensa recebida, em função da quantidade de recursos de computação solicitada e a efetivamente utilizada no tempo t.

$$\eta_{i,t} = \alpha \frac{R_{i,t}^{(s)}}{C} + (1-\alpha)\frac{R_{i,t}^{(s)} - \lambda_{i,t}}{R_{i,t}^{(s)}}, \tag{4-3}$$

$$\kappa_{i,t} = \sigma \frac{Q_{i,t}^{(s)}}{M} + (1-\sigma)\frac{Q_{i,t}^{(s)} - \xi_{i,t}}{Q_{i,t}^{(s)}}, \tag{4-4}$$

Nas Equações 4-3 e 4-4, α e σ são parâmetros que atribuem pesos à quantidade de recursos solicitada e o ganho de multiplexação. A Tabela 4.1 apresenta um resumo das variáveis utilizadas em nosso modelo.

Tabela 4.1: Variáveis utilizadas em nosso modelo.

$\mathcal{I}$	Conjunto de inquilinos
$\mathcal{T}$	Conjunto de estampas de tempo onde são feitas decisões
C	Capacidade total disponível para transmitir dados na rede
M	Capacidade total disponível para processamento de dados na borda da rede
$\eta_{i,t}$	Valor de recompensa recebida, em termos de recursos de rede
$\kappa_{i,t}$	Valor de recompensa recebida, em termos de recursos de computação
$R_{i,t}^{(s)}$	Quantidade de PRBs requisitados pelo inquilino i no tempo t
$\lambda_{i,t}$	Quantidade de PRBs efetivamente utilizados pelo inquilino i em t
$Q_{i,t}^{(s)}$	Quantidade de CPU *ticks* requisitados pelo inquilino i no tempo t
$\xi_{i,t}$	Quantidade de CPU *ticks* efetivamente utilizados pelo inquilino i em t
α	Peso atribuído à quantidade de recursos de rede solicitada
σ	Peso atribuído à quantidade de recursos de computação de borda solicitada

4.2 Formulação do Problema

Nesta seção, formulamos o problema do controle de admissão de *slices* na infraestrutura do operador como um problema de programação linear. Consideramos que para todo inquilino $i \in \mathcal{I}$ e para todo instante de tempo $t \in \mathcal{T}$, exista uma variável de decisão representada por:

- $x_{i,t} \in \{0,1\}$, a qual assume o valor 1, quando o a requisição de criação de *slice* do inquilino i é aceita em t, e o valor 0, caso contrário.

Dessa maneira, o problema da admissão de *slices* pode ser formulado como:

$$\textit{maximizar} \qquad \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}(\eta_{i,t}+\kappa_{i,t})x_{i,t} \tag{4-5}$$

sujeito a:

$$\sum_{i\in\mathcal{I}}\lambda_{i,t}x_{i,t}\leq C,\forall t\in\mathcal{T} \tag{4-6}$$

$$\sum_{i\in\mathcal{I}}\xi_{i,t}x_{i,t}\leq M,\forall t\in\mathcal{T} \tag{4-7}$$

$$x_{i,t}\geq x_{i,t-1}\mathbb{F}(t-t_i^{start}\leq L_i^{(s)}) \tag{4-8}$$

$$x_{i,t}\in\{0,1\},\forall i\in\mathcal{I},\forall t\in\mathcal{T} \tag{4-9}$$

A função objetivo 4-5 incentiva a aceitação das requisições dos inquilinos, considerando recursos de comunicação e computação de borda. A restrição em 4-6 garante que a quantidade de PRBs utilizada não exceda a capacidade disponível. Em 4-7 garantimos que a quantidade de CPU *ticks* utilizada não exceda a capacidade disponível. Em 4-8 impomos a seleção do inquilino i enquanto durar a reserva do *slice* s previamente concedido. Denotamos por t_i^{start} a rodada em que a requisição s do inquilino i foi aceita, e $\mathbb{F}$ é uma função indicadora que fornece valor 1, se a condição entre colchetes for satisfeita, e o valor 0, caso contrário. Por fim, em 4-9, estabelecemos a variável binária de decisão.

4.3 Proposta

O problema apresentado na Seção 4.2 é NP-difícil em sua versão como problema de decisão: determinar se existe solução viável para a qual o valor da função objetivo 4-5 é maior que um dado valor constante arbitrário. De fato, verificarmos que o problema é NP-difícil, fazendo uma redução do problema do controle de admissão de *slice*, formulado, considerando o problema do bandido multi-armado com restrição de *lock-up* [Komiyama, Sato e Nakagawa 2013], apresentado e discutido no Capítulo 3.

Neste contexto, a técnica de RL pode ser utilizada baseando-se em *feedbacks* constantes com o ambiente. Através do treinamento do agente de RL, é possível adaptar os *slices* em tempo real, levando em consideração as mudanças no tráfego de dados, o surgimento de novas demandas e a disponibilidade de recursos computacionais. Dessa maneira, é possível obter uma rede mais flexível e eficiente, que se adapta às necessidades do usuário e maximiza o desempenho da rede como um todo.

Como mencionado no Capítulo 2, cada *slice* deve ser projetado para atender a um conjunto específico de requisitos, i.e., latência, largura de banda, segurança, e outros, permitindo a oferta de *slices* personalizados para diferentes tipos de aplicações. No entanto, a criação de *slices* eficientes depende não somente da capacidade da rede física,

conforme apresentado no Capítulo 3, mas também da capacidade de processamento dos servidores de borda. A insuficiência de capacidade de processamento pode resultar em latências de processamento mais elevadas, interrupções de serviço e outros problemas. De modo análogo, uma rede subdimensionada não será capaz de acomodar o número de *slices* necessários para atender à demanda, o que pode prejudicar a qualidade do serviço oferecido aos usuários.

Dessa maneira, para preencher essa lacuna, propomos os algoritmos MONETS-OBD, MONETS-OBS e CAONS, algoritmos de RL, modelados como problemas do bandido multi-armado com restrição de *lock-up* para o controle de admissão *slices*, que consideram tanto recursos de comunicação quanto de computação para tomada de decisões. Os algoritmos são apresentados nas Seções 4.3.1, 4.3.1 e 4.3.3.

4.3.1 Algoritmo MONETS-OBD

Inicialmente, propomos uma extensão da solução apresentada por [Sciancalepore et al. 2022], denominada *MEC Online Network Slicing with Overbooking* (MONETS-OBD). Em MONETS-OBD utilizamos uma versão do algoritmo do bandido multi-armado com restrições de *lock-up* para controlar a admissão de *slices* na infraestrutura do operador, respeitando os orçamentos de comunicação (C) e computação (M) disponíveis. O pseudo-código é apresentado no Algoritmo 4.1.

No algoritmo MONETS-OBD, propomos a resolução do problema de controle de admissão de *slices* a cada rodada t, pela seleção limitada de K máquinas caça-níquel, cada uma representando um inquilino i. Essa medida prioriza o atendimento dos inquilinos que historicamente retornam as melhores recompensas e diminui a complexidade computacional do algoritmo para O(K). O objetivo é maximizar a média da distribuição empírica da função objetivo. Esse algoritmo se distancia dos demais estudados no Capítulo 3 considerando os recursos computacionais no conjunto de análise, além de trabalhar com a média histórica acumulada da utilização dos recursos, o que confere maior direcionamento ao agente no processo de escolha de inquilinos.

No algoritmo, inicializamos as variáveis na linha 1. As linhas 2-6, correspondem à etapa de treinamento em que todos os inquilinos demonstram interesse pela utilização de *slices*. Para cada rodada, na linha 8, a recompensa v_i é computada em consideração à recompensa média fornecida por i até o momento ($\bar{\theta}_i(t)$). O termo $\sqrt{\frac{2\ log\ t}{W_i}}$ é introduzido para dar peso maior para médias de recompensa obtidas em janelas mais longas. Nas linhas 10-12, inquilinos cujo *slices* já estão admitidos e ativos recebem o recurso acordado anteriormente. Caso não haja recurso disponível suficiente para atendimento desses inquilino, o inquilino não será atendido e não retornará nenhuma recompensa. Se após o processamento de todos os inquilinos com *slices* ativos ainda houver recursos (orçamento) disponíveis, busca-se, entre os inquilinos restantes, aqueles que possuam as maiores

recompensas médias ($\bar{\theta}_i(t)$), respeitando-se o limite de restrição de orçamento, até K inquilinos. As linhas 16-21 implementam a estratégia de *overbooking*, contabilizando os recursos alocados para i não pela quantidade prevista no *template* de *slices*, i.e., $R_{i,t}^{(s)}$ e $Q_{i,t}^{(s)}$, mas pelo número médio de PRBs e de CPU *ticks* utilizados por i até t, i.e., $\hat{\lambda}_i$ e $\hat{\xi}_i$.

4.3.2 Algoritmo MONETS-OBS

Apesar de promissora, a estratégia apresentada na seção 4.3.1, que aloca recursos no sistema considerando a utilização média dos recursos (i.e., $\hat{\lambda}_i$ e $\hat{\xi}_i$), pode resultar em violações de SLA. Análises e dados numéricos para esta afirmação são fornecidos no Capítulo 5. Desse modo, para atenuar esse efeito, propomos o *Online Network Slicing with MEC and Overbooking by Empirical Rule* (MONETS-OBS), que utiliza a regra empírica do desvio padrão para contabilizar os recursos alocados.

Em MONETS-OBS utilizamos um fator de reserva dos recursos de rede e computacionais baseado-nos na regra empírica do desvio padrão (φ), aplicado-a em ma distribuição normal com média μ. Esta regra é baseada na propriedade de que a maioria dos dados em uma distribuição normal se concentra em torno da média, e quanto mais longe da média, menos provável é a ocorrência de um determinado valor. Pela regra, cerca de 68% dos dados em uma distribuição normal estão concentrados no intervalo $[\mu-\varphi,\mu+\varphi]$. Cerca de 95% dos dados estão concentrados no intervalo $[\mu-2\varphi,\mu+2\varphi]$, enquanto cerca de 99,7% dos dados estão concentrados no intervalo $[\mu-3\varphi,\mu+3\varphi]$.

Desse modo, denotamos por φ_{λ_i} e φ_{ξ_i} o desvio padrão do número médio de PRBs e do número médio de CPU *ticks* utilizados por i, respectivamente. O algoritmo MONETS-OBS é similar ao Algoritmo 4.1, exceto pelas linha 16, 18 e 19 onde $\hat{\lambda}_i$ é substituída por $\hat{\lambda}_i+(3\times\varphi_{\lambda_i})$ e $\hat{\xi}_i$ é substituída por $\hat{\xi}_i+(3\times\varphi_{\xi_i})$. No algoritmo, utilizamos 3 unidades do desvio padrão para abranger o conjunto de 99,7% dos casos simulados. Ao utilizar a regra do desvio padrão, podemos reduzir de forma considerável, o número de violações de SLA. O pseudo-código é apresentado no Algoritmo 4.2.

4.3.3 Algoritmo CAONS

Os algoritmos de RL utilizados nesta tese fazem parte de um ferramental de aprendizado de máquina que permitem que um agente interaja com um ambiente e aprenda a tomar decisões a fim de maximizar uma recompensa. O agente recebe informações sobre o estado do ambiente e executa ações que podem mudar o estado do ambiente. A recompensa é fornecida pelo ambiente em resposta às ações do agente. Contudo, os algoritmos de RL não trabalham apenas dessa maneira. Os algoritmos de RL contam com duas subcategorias principais a saber: os algoritmos que não usam penalidade e os que utilizam. A principal diferença entre os dois é a maneira como lidam com os erros e as consequências.

Algoritmo 4.1: M-ONETS-OBD - Algoritmo de seleção do conjunto de inquilinos, com garantia de orçamento fixo e período de reserva.

Entrada: K, I, T, C, M.

Inicialização: $B = 0$, $Z = 0$, $L \leftarrow \emptyset$, $W_i = 0$, n=0, $\bar{\theta}_i(0) = 0 \; \forall i \in I$

forall $i \in I$ **do**
  $v_i \leftarrow \eta_{i,0} + \kappa_{i,0}$
  ATUALIZE $\bar{\theta}_i(0)$
  $W_i = W_i + 1$
end
forall $t \in T$ **do**
  $\hat{\theta}_i(t) = \bar{\theta}_i(t) + \sqrt{\frac{2 \; log \; t}{W_i}}$
  while $n \leq K$ **do**
    if $L(t) \neq \emptyset$ **then**
      $\hat{i} \leftarrow L(t)$
    end
    else
      $\hat{i}$: arg max $\hat{\theta}_i(t)$
    end
    if $B + \hat{\lambda}_i \leq C$ ***and*** $Z + \hat{\xi}_i \leq M$ **then**
      $R \leftarrow R \cup \hat{i}$
      $B = B + \hat{\lambda}_i$
      $Z = Z + \hat{\xi}_i$
      $n = n + 1$
    end
  end
  forall $i \in R$ **do**
    $v_i \leftarrow \eta_{i,t} + \kappa_{i,t}$
    ATUALIZE $\bar{\theta}_i(t)$
    $W_i = W_i + 1$
  end
  ATUALIZE $L(t)$
  $B = 0$; $Z = 0$; $n = 0$
end

Algoritmo 4.2: MONETS-OBS - Algoritmo de seleção do conjunto de inquilinos, com redução de violação de orçamento e de SLA.

Entrada: K, I, T, C, M.

Inicialização: $B = 0$, $Z = 0$, $L \leftarrow \emptyset$, $W_i = 0$, n=0, $\bar{\theta}_i(0) = 0 \ \ \forall\, i \in I$

forall $i \in I$ **do**
  $v_i \leftarrow \eta_{i,0} + \kappa_{i,0}$
  ATUALIZE $\bar{\theta}_i(0)$
  $W_i = W_i + 1$
end
forall $t \in T$ **do**
  $\hat{\theta}_i(t) = \bar{\theta}_i(t) + \sqrt{\frac{2\ log\ t}{W_i}}$
  while $n \leq K$ **do**
    if $L(t) \neq \emptyset$ **then**
      $\hat{i} \leftarrow L(t)$
    end
    else
      $\hat{i}$: arg max $\hat{\theta}_i(t)$
    end
    if $B + \hat{\lambda}_i + (3 \times \varphi_{\lambda_i}) \leq C$ ***and*** $Z + \hat{\xi}_i + (3 \times \varphi_{\xi_i}) \leq M$ **then**
      $R \leftarrow R \cup \hat{i}$
      $B = B + \hat{\lambda}_i + (3 \times \varphi_{\lambda_i})$
      $Z = Z + \hat{\xi}_i + (3 \times \varphi_{\xi_i})$
      $n = n + 1$
    end
  end
  forall $i \in R$ **do**
    $v_i \leftarrow \eta_{i,t} + \kappa_{i,t}$
    ATUALIZE $\bar{\theta}_i(t)$
    $W_i = W_i + 1$
  end
  ATUALIZE $L(t)$
  $B = 0$;
  $Z = 0$;
  $n = 0$
end

Algoritmos de RL que não usam penalidade são aqueles que não penalizam o agente pela tomada de ações erradas ou desfavoráveis. Em vez disso, os algoritmos recompensam o agente a cada ações correta ou favorável. Essa abordagem incentiva o agente a explorar o ambiente para encontrar ações que levem a recompensas positivas. Os Algoritmos 4.1 e 4.2 fazem parte deste grupo.

Por outro lado, os algoritmos de RL que usam penalidade para o aprendizado incidem em decréscimos na recompensa do agente diante de ações desfavoráveis. Essa penalidade é aplicada quando o agente toma uma ação que leva a uma recompensa distante da recompensa esperada. O uso de penalidade no processo de treinamento do algoritmo incentiva o agente a evitar ações que levem a consequências sub-ótimas. Dessa maneira, geralmente, os algoritmos de RL que utilizam penalidade acabam convergindo mais rapidamente. Além disso, a penalidade pode ser usada para reforçar o conjunto de comportamentos desejados, ajudando a moldar a política de decisão do agente. Contudo, o uso de penalidade em RL deve ser controlado, pois a aplicação de penalizações excessivas pode levar a uma política muito conservadora, ao passo que penalizações insuficientes podem levar a uma política muito agressiva.

Através da análise dos pontos de utilização de penalidades em RL, foi possível propor o algoritmo CAONS (Controle de Admissão On-line para RAN *Slicing*), que utiliza as penalidades como um dos elementos centrais de seu processo de aprendizagem. O CAONS é uma evolução dos algoritmos MONETS-OBD e MONETS-OBS, propostos nesta tese, com adição de submetas ou subobjetivos que orientam o agente em direção à soluções mais eficientes e precisas. CAONS utiliza um fator de penalidade para identificar e associar as diferentes classes de tarefas que o agente deve realizar. O algoritmo permite utilizar do conhecimento prévio sobre o ambiente, aumentando a eficácia da aprendizagem e a capacidade de generalização do agente. O pseudo-código de CAONS é apresentado no Algoritmo 4.3. CAONS permite que o agente aprenda a lidar com situações imprevistas, pois as submetas permitem que o agente volte a se concentrar em um objetivo diferente, caso uma solução não seja encontrada inicialmente.

Inicializamos as variáveis na linha 1. Nas linhas de 2-6 inquilinos apresentam interesse na utilização de *slices*. Na linha 8 a recompensa privilegia janelas maiores de reserva. Na linha 10 verificamos a fila de *lock-up*, selecionando inquilinos ativos em rodadas anteriores. A capacidade de atendimento é verificada na linha 12. Caso não hajam recursos para inquilinos em *lock-up*, o agente será penalizado pela realização da escolha desfavorável. A penalização ocorre em 3 unidades do valor esperado para recompensa, forçando o agente a evitar a seleção de inquilinos com alta variância e baixo retorno e rodadas futuras próximas. Na linha 20 selecionamos novos inquilinos para atendimento, considerando a média histórica acumulada. Por fim, entre 27-31 atualizamos os dados da distribuição de recompensa dos inquilinos acumulada até o momento da execução.

Algoritmo 4.3: CAONS - Algoritmo de seleção do conjunto de inquilinos, utilizando penalizações.

Entrada: K, I, T, C, M.

Inicialização: $B = 0$, $Z = 0$, $L \leftarrow \emptyset$, $W_i = 0$, $\bar{\theta}_i(0) = 0 \; \forall i \in I$

forall $i \in I$ **do**
  $v_i \leftarrow \eta_{i,0} + \kappa_{i,0}$
  ATUALIZE $\bar{\theta}_i(0)$
  $W_i = W_i + 1$
end
forall $t \in T$ **do**
  $\hat{\theta}_i(t) = \bar{\theta}_i(t) + \sqrt{\frac{2 \; log \; t}{W_i}}$
  while $n \leq K$ **do**
    if $L(t) \neq \emptyset$ **then**
      $\hat{i} \leftarrow L(\mathrm{t})$
      if $B + \lambda_i \leq C$ ***and*** $Z + \xi_i \leq M$ **then**
        $R \leftarrow R \cup \hat{i}, \; B = B + \lambda_i, \; Z = Z + \xi_i$
      end
      else
        $p_{i,t} = 3(\eta_{i,t} + \kappa_{i,t})$
      end
    end
    else
      $\hat{i}$: arg max $\hat{\theta}_i(t)$
      if $B + \lambda_i \leq C$ ***and*** $Z + \xi_i \leq M$ **then**
        $R \leftarrow R \cup \hat{i}, \; B = B + \lambda_i, \; Z = Z + \xi_i$
      end
    end
    $n = n + 1$
  end
  forall $i \in R$ **do**
    $v_i \leftarrow \eta_{i,t} + \kappa_{i,t}$ - $p_{i,t}$
    ATUALIZE $\bar{\theta}_i(t)$
    $W_i = W_i + 1$
  end
  ATUALIZE $L(t)$
  $B = 0; Z = 0, n = 0$
end

O algoritmo CAONS é uma abordagem promissora para o RL, pois combina a utilização de penalidades com a orientação do agente em direção a submetas específicos. Essa abordagem aumenta a eficácia da aprendizagem, a capacidade de generalização do agente e permite que ele aprenda a lidar com situações imprevistas de forma mais eficiente.

4.4 Considerações Finais

Neste capítulo, apresentamos a formulação matemática do problema de admissão de *slices* em um provedor de telecomunicação levando em consideração recursos de rede e de computação de borda. Apresentamos três algoritmos, baseados em aprendizado por reforço, para a solução do problema. No próximo capítulo, apresentaremos a avaliação de desempenho realizada e discutimos os resultados obtidos.

CAPÍTULO 5

Avaliação Experimental

Neste capítulo, apresentamos a avaliação de desempenho para os algoritmos apresentados nos Capítulos 3 e 4. Descrevermos os cenários e a metodologia utilizada na avaliação experimental, detalhando a infraestrutura adotada, as aplicações e os cenários. Por fim, discutimos os resultados obtidos.

5.1 Características da Infraestrutura

Na configuração utilizada nesta tese, consideramos que a RAN é composta por um conjunto de dispositivos, softwares, conexões e protocolos, que juntos, permitem a comunicação de dados entre os mais diferentes tipos de dispositivos. Uma rede de comunicação pode ser vista como um conjunto de dispositivos eletrônicos que são conectados fisicamente ou sem fio, com o objetivo de compartilhar recursos [Singh e Kumbhani 2020]. Devido as características de organização em camadas, a infraestrutura de uma RAN pode ser dividida em partes menores, possibilitando que cada uma execute funções específicas. A camada física, por exemplo, lida com a transmissão de dados e a conectividade física entre os dispositivos. A camada de rede é responsável por gerenciar o tráfego de dados e as rotas utilizadas para a entrega dos pacotes. A camada de aplicação, presente no topo do modelo, é responsável pela interação entre usuários e serviços.

Além disso, as redes 5G trouxeram consigo uma ampla gama de aplicações, que requerem características específicas de latência, transmissão e processamento. Dessa maneira, é fundamental que exista uma medida da capacidade de transmissão de dados ao longo infraestrutura das redes móveis que esteja diretamente relacionada à unidade de transmissão. Como mencionado anteriormente, neste trabalho utilizamos a unidade PRBs para descrever e quantificar os recursos de rede oferecidos por uma estação base. Escolhemos a unidade de medida pois PRBs são blocos de recursos físicos tradicionalmente utilizados pelas redes móveis para alocar e transmitir dados em uma rede celular. PRBs são uma medida fundamental no controle de admissão para a transmissão de dados em redes móveis. Eles são usados para determinar a quantidade de largura de banda que é necessária para transmitir dados de maneira eficiente entre dispositivos móveis e estações base [Wu et al. 2015].

Em 5G, a infraestrutura das redes móveis deve ser projetada de maneira a acomodar a transmissão de PRBs de maneira eficiente e confiável. Desse modo, as estações bases nesse tipo de rede são responsáveis por gerenciar PRBs de diferentes dispositivos na rede, garantindo que a largura de banda seja alocada de maneira justa e eficiente [Hwang e Park 2017].

Além disso, as redes móveis 5G também devem ser capazes de monitorar a alocação de PRBs para garantir que a rede esteja operando de maneira ideal. Isso envolve a medição da qualidade do sinal, o monitoramento da utilização de PRBs e a identificação e resolução de problemas que possam afetar a eficiência da transmissão de dados. Os sistemas *Long Term Evolution* (LTE) suportam diferentes larguras de banda em canais distintos. Nesse tipo de sistema, um PRBs ocupa cerca de 180 KHz [Wang, Rosa e Pedersen 2010]. Isso significa que, em um sistema LTE de largura de banda igual a 40 MHz, temos alocados aproximadamente 216 PRBs [3GPP 2009].

Nesse tipo de sistema, a taxa de transferência é representada pelo número de símbolos transmitidos por unidade de tempo. A taxa de transmissão pode ser expressa em bits por segundo, dada a quantidade de bits que um símbolo pode representar. Consideremos um sistema LTE que utiliza *Frequency Division Duplex* (FDD) de 40 MHz, com 216 PRBs. Em cada PRBs são representados 168 símbolos por milissegundo (ms). Nesse sistema existem 36.288/ms símbolos, ou 36.288.000 símbolos por segundo. Se a modulação utilizada for de 64 *Quadrature Amplitude Modulation* (QAM), correspondente à representação de 6 bits por símbolo, a taxa de transferência desse sistema será de 36.288 x 6 = 217,728 megabits por segundo (Mbps), para uma única antena [Hwang e Park 2017].

Visando aumentar a capacidade de transmissão, os sistemas LTE utilizam não apenas uma, mas múltiplas antenas de transmissão e recepção, explorando a propagação multicaminho. A esse sistema é dado o nome de Multiple Input Multiple Output (MIMO). Um sistema do tipo 4 MIMO retorna uma taxa de transferência de 217,728 x 4 = 870,912 Mbps. É importante ressaltar que em LTE, cerca de 25% da taxa total de transmissão é utilizada para controle e sinalização, portanto, a taxa efetiva de transferência, nesse caso, será de aproximadamente 648 Mbps [Hwang e Park 2017]. Essa configuração de 216 PRBs é a utilizada para a estação base de referência considerada neste trabalho.

Para quantificar os recursos de computação oferecidos por um servidor de borda, usamos o conceito de CPU *ticks*. Um *tick* corresponde à menor unidade de medida de tempo de uma CPU em um sistema de computação [Gunther 2007]. Geralmente os sistemas computacionais implementam contadores internos que contabilizam a quantidade de *ticks* consumidos. Essa informação é importante pois até mesmo data, hora e várias outras funções do sistema operacional são derivadas desse contador. A quantidade de *ticks* executadas em um sistema depende diretamente do próprio sistema e de suas configurações de hardware. No Linux, por exemplo, são executados 10.000 *ticks* por ms [Weisbecker 2013].

5.2 Características das Aplicações

O caso de uso adotado nesta tese inclui algumas aplicações das redes 5G, elas foram selecionadas por possuírem características únicas que as diferenciam da maioria das aplicações que operam nas anteriores das redes móveis. Dentre as características, podemos destacar a necessidade de alta velocidade, latência reduzida, maior capacidade de rede e conectividade massiva de dispositivos [Medeiros et al. 2017].

A alta velocidade das redes 5G permite a transmissão de grandes quantidades de dados em tempo real, possibilitando o desenvolvimento de novas aplicações em áreas como realidade virtual e aumentada, jogos online, telemedicina e veículos autônomos. A latência reduzida permite uma resposta rápida e em tempo real a comandos e solicitações de dispositivos conectados. Isso é fundamental para aplicações como jogos online, drones e veículos autônomos. A conectividade massiva de dispositivos permite a conexão simultânea de um grande número de dispositivos, incluindo dispositivos de baixa potência e dispositivos que não exigem alta largura de banda e possibilitando a implementação de soluções de IoT em larga escala [Medeiros et al. 2017].

Diante dessa ampla gama de aplicações com características intrínsecas e ímpares, estimar a quantidade de processamento necessária para atender a uma demanda na rede é uma tarefa desafiadora. A quantidade de CPU necessária para processar um *gigabit per second* (Gbps) de dados em uma aplicação de vídeo por *streaming*, por exemplo, não é a mesma quantidade de CPU que seria requerida por uma aplicação de jogos ou de mapas [Malandrino et al. 2020].

Para fornecer uma alternativa a esse problema, Malandrino et al. [Malandrino et al. 2020] analisam um conjunto de dados reais, extraídos de *traces* de redes celulares posicionadas em três grandes cidades americanas, a saber: Atlanta, Los Angeles e San Francisco. Particularmente, os autores analisam dados de *traces* reais, com o objetivo de propor distribuições matemáticas que descrevam e recriem a demanda de dados com a quantidade de processamento necessário para cumprir o propósito da aplicação.

Segundo [Malandrino et al. 2020], cerca de 66% do tráfego observado no conjunto de dados pode ser originário de inquilinos que ofertam serviços de vídeo por *streamming*. Outros 15% são originários de inquilinos que prestam serviços de jogos online. Por fim, aproximadamente 19% correspondem a inquilinos que prestam serviço de mapas e geolocalização. A partir dos dados os autores analisam ainda, como a quantidade de recursos de comunicação requerida impacta na quantidade de recursos de computação, para cada uma das aplicações citadas. Essas relações são apresentadas na Equação 5-1.

$$CPU_{ticks} = \begin{cases} 0,25 * trafego_{Mbit} + 6,76 & para\ video \\ 161,38 * trafego_{Mbit} + 1675,03 & para\ jogos \\ 67,44 * trafego_{Mbit} - 7,53 & para\ mapas \end{cases} \quad (5\text{-}1)$$

Dessa maneira, nesta tese utilizamos o modelo de perfis de tráfego proposto por Malandrino et al. [Malandrino et al. 2020] para caracterizar os serviços providos por inquilinos em nosso modelo. Dessa forma, assumimos três tipos diferentes de inquilinos no nosso sistema: (1) os que oferecem serviço de vídeo por *streaming*, (2) os que proveem serviços de jogos online e (3) os que oferecem serviços de geolocalização e mapas. A Tabela 5.1 apresenta as proporções de cada um desses tráfegos no tráfego total da estação base de referência adotada nesta tese. Essas proporções são baseadas nos valores observados em [Malandrino et al. 2020]. A coluna PRBs na Tabela 5.1 representa a quantidade de PRBs utilizada por cada tipo de tráfego dado sua proporção. A última coluna dessa tabela representa o volume de tráfego, em Mbps, gerado por cada aplicação (serviço), também segundo sua proporção.

Tabela 5.1: Perfil de consumo de recursos de comunicação por inquilinos do sistema

Aplicação	Percentual	PRBs	Mbps
Vídeo	66%	142	426
Jogos	15%	33	99
Mapas	19%	41	123
Total	100%	216	648

5.3 Características dos *Templates* de *Slices*

Dada a Tabela 5.1 e as relações apresentada na Equação 5-1, podemos calcular a quantidade de CPU *ticks* necessária para processar cada tipo de tráfego considerado na Tabela 5.1. A Tabela 5.2 apresenta os valores obtidos.

Tabela 5.2: Perfil dos possíveis inquilinos do sistema

Aplicação	Percentual	PRBs	Mbps	CPU *ticks*
Vídeo	66%	142	426	113,26
Jogos	15%	33	99	17.651,65
Mapas	19%	41	123	8.287,5
Total	100%	216	648	26.052,50

A Tabela 5.2 demonstra que serviços de vídeo transferem massivamente mais dados que serviços de jogos online. Portanto, inquilinos de vídeo tendem a consumir grande parte da largura de banda nas redes de comunicação. Por outro lado, a quantidade

de recursos de CPU consumida por serviços de jogos e mapas supera a quantidade requerida por serviços de vídeo. Isso é razoável, uma vez que o processamento de vídeo envolve basicamente a leitura de um arquivo do disco e sua alimentação em um fluxo de rede. Serviços de jogos e mapas, por sua vez, necessitam de realizar operações complexas, como manter o controle do status do jogo ou renderizar os mapas.

Como mencionamos anteriormente, consideramos diferentes *templates* de *slices* para o mesmo serviço. Particularmente, em cada tipo de serviço (video por *streamming*, jogos online e mapas), consideramos três tipos de *templates*: Templates de *Demanda Máxima*, Templates de *Demanda Mínima* e Templates de *Demanda Variável*. Esses *templates* diferem entre si pela quantidade de recursos oferecida. *Templates* de *Demanda Máxima* são configurados com a maior quantidade de recursos de rede e de computação disponível para o serviço. Por exemplo, os parâmetros $R_{i,t}^{(s)}$ e $Q_{i,t}^{(s)}$ do *template Demanda Máxima* para o serviço de vídeo são configurados com os valores 426 Mbps e 113,26 CPU *ticks*, respectivamente. *Templates* de *Demanda Mínima* são configurados com a menor quantidade de recursos de rede e de computação disponível para o serviço. Por fim, *Templates* de *Demanda Variável* são configurados com uma quantidade de recursos de rede e de computação entre os valores máximos e mínimos. A Tabela 5.3 apresenta a quantidade de recurso de comunicação e de computação necessários para possibilitar a execução das aplicações dos variados tipo de serviços analisados. Utilizar diferentes tipos de *templates* para os diferentes serviços pode aumentar a diversidade dos cenários analisados em nossos experimentos.

Tabela 5.3: Possíveis *slices templates* disponibilizados à inquilinos

Aplicação	Demanda Mínima		Demanda Média		Demanda Máxima	
	Mbps	CPU ticks	Mbps	CPU ticks	Mbps	CPU ticks
Vídeo	142	42,26	284	77,76	426	113,26
Jogos	33	7.000,57	66	12.326,11	99	17.651,65
Mapas	41	2.757,51	82	5.522,55	123	8.287,59

5.4 Cenários e Experimentos

Para avaliar os algoritmos apresentados nos Capítulos 3 e 4, definimos um conjunto de experimentos, como descrito: i) consideramos um cenário composto de uma *Base Station* (BS) e um servidor MEC, ii) utilizamos os valores apresentados na Tabela 5.4 e iii) geramos 12 cenários de experimentação. Cada cenário envolve um tipo de serviço (Video, Jogos, Mapa ou Variado) e um tipo de *template* de *slices* (*Demanda Máxima*, *Demanda Mínima* ou *Demanda Variada*). Nos cenários variados, todos os serviços (Vídeo, Jogos, Mapas e Variado) são considerados, sendo o tipo de serviço associado ao inquilino sorteado com o uso de uma distribuição uniforme.

Tabela 5.4: Cenários Avaliados

Nome	Tipo de Serviço	*Template* Utilizado
Cenário de Video 1	Vídeo	*Demanda Variada*
Cenário de Video 2	Vídeo	*Demanda Máxima*
Cenário de Video 3	Vídeo	*Demanda Mínima*
Cenário de Jogos 1	Jogos	*Demanda Variada*
Cenário de Jogos 2	Jogos	*Demanda Máxima*
Cenário de Jogos 3	Jogos	*Demanda Mínima*
Cenário de Mapas 1	Mapas	*Demanda Variada*
Cenário de Mapas 2	Mapas	*Demanda Máxima*
Cenário de Mapas 3	Mapas	*Demanda Mínima*
Cenário Variado 1	Vídeo, Jogos e Mapas	*Demanda Variada*
Cenário Variado 2	Vídeo, Jogos e Mapas	*Demanda Máxima*
Cenário Variado 3	Vídeo, Jogos e Mapas	*Demanda Mínima*

Tabela 5.5: Parâmetros da simulação

Parâmetros	Valores		Parâmetros	Valores
$\|\mathcal{I}\|$	10		α	0,5
$\|\mathcal{T}\|$	10000		σ	0,5
C	216 PRBs (40 MHz)		b (ϵ-Greedy)	10
M	26.052,50 CPU *ticks*		d (ϵ-Greedy)	0,1
μ	10% a 90% de $R_{i,t}^{(s)}$		ω	1
ρ	100		K	10
ζ	0,2		L	100, 500 ou 1000

Em todos os cenários, geramos as requisições de *slices* a partir de uma distribuição exponencial, com parâmetros ρ e ζ fixos, que determinam a frequência das ocorrências de requisições. A utilização dos recursos de rede de um inquilino i em t ($\lambda_{i,t}$) é gerada a partir das distribuições gaussiana, lognormal e uniforme com média μ e desvio padrão ω. Para criar diferentes cenários de tráfego efetivo, μ vaira entre 10% a 90% do parâmetro $R_{i,t}^{(s)}$, definido no *template* de *slice* associado ao inquilino i. A utilização efetiva dos recursos computacionais, $\xi_{i,t}$, é obtida a partir de $\lambda_{i,t}$, usando as relações descritas na Equação 5-1. A Tabela 5.5 resume os parâmetros utilizados na avaliação.

A cada rodada $t \in \mathcal{T}$, a política de seleção implementada analisa o melhor conjunto de inquilinos para serem aceitos. Para efeitos de comparação, implementamos também a formulação ótima dos problemas apresentados nos Capítulos 3 e 4. Utilizamos a ferramenta CPLEX [CPLEX 2018] para implementar o modelo matemático para solução de referência e a linguagem de programação Python para os demais algoritmos. Realizamos 30 execuções para cada cenário descrito na Tabela 5.4. Os resultados apresentados representam as médias obtidas ao longo de 30 execuções, que fornecem 95% de confiança acerca dos resultados apresentados [Sim e Reid 1999].

5.5 Resultados dos Algoritmos Clássicos de MaB

Nesta seção, analisamos os resultados obtidos pelos algoritmos apresentados no Capítulo 3. Os algoritmos levam em consideração os recursos de comunicação necessários para atendimento das aplicações de inquilinos. Nos resultados, os algoritmos são representados nos gráficos como ÓTIMO, eUCB, ONETS e ϵ-Greedy.

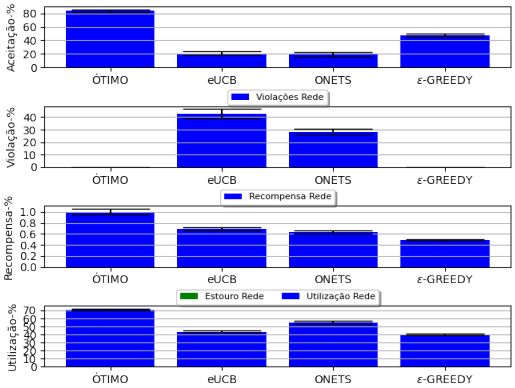
5.5.1 Cenário de Vídeo

Apresentamos na Figura 5.1 os resultados para os cenários Vídeo 1, Vídeo 2 e Vídeo 3 para o problema estudado no Capítulo 3. Nesses cenários, todos os inquilinos que requisitam criação de *slices* oferecem serviços de vídeos. Entretanto, os *templates* requisitados variam em cada cenário. No cenário Vídeo 1, os inquilinos requisitam *templates* de vídeo com demanda variada, ou seja, valores de demanda que variam entre a demanda máxima e a mínima. Portanto, o conjunto de requisições será heterogêneo. No cenário Vídeo 2, todos os inquilinos requisitam *templates* de vídeo com demanda Máxima. Logo, esse cenário apresenta a maior demanda em termos de recursos de rede, em relação aos demais cenários de vídeo. No cenário Vídeo 3, todos os inquilinos requisitam *templates* de vídeo com demanda mínima. Logo, esse cenário apresenta a menor demanda em termos de recursos de rede, em relação aos demais cenários de vídeo.

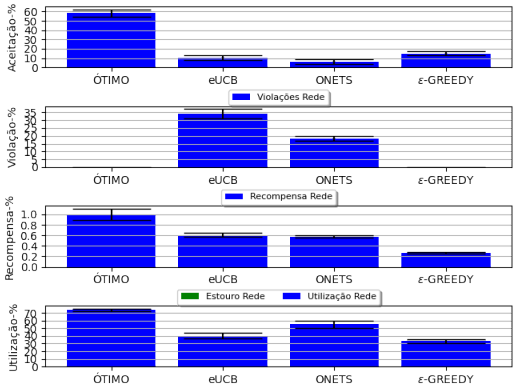
As Figuras 5.1(a), 5.1(b) e 5.1(c) apresentam a taxa de aceitação média, a taxa de violação média, a recompensa média e a taxa de utilização média da infraestrutura para os cenários Vídeo 1, Vídeo 2 e Vídeo 3, respectivamente. Em cada figura, a taxa de aceitação média é ilustrada no gráfico superior, seguida da taxa de violação média e da recompensa média, enquanto a taxa de utilização média é mostrada no gráfico inferior. Nos gráficos de utilização, os resultados obtidos pelos algoritmos clássicos de MaB são mostrados em duas barras empilhadas. A barra superior (em verde) corresponde ao percentual médio da violação da capacidade dos recursos computacionais, quando e somente se ocorrer. Enquanto que a barra inferior (em azul) representa o percentual médio de utilização dos recursos computacionais.

Analisando a Figura 5.1(a), observamos que a maior taxa de aceitação é obtida pelo ÓTIMO, que acomoda em torno de 80% das requisições. As menores taxas de aceitação são obtidas no cenário de demanda máxima (5.1(c)), que forçam uma utilização massiva dos recursos de comunicação. O algoritmo ϵ-Greedy acomoda em torno de 50%, enquanto eUCB e ONETS acomodam em torno de 20% e 21%, respectivamente. A Figura apresenta ainda que ÓTIMO e ϵ-Greedy não violam as requisições de SLA das aplicações, ao passo que, do número total de requisições atendidas, eUCB e ONETS violam aproximadamente 40% e 29%, respectivamente. Como esperado, nos três cenários avaliados, o algoritmo ÓTIMO alcança as maiores taxas médias de recompensas, seguido

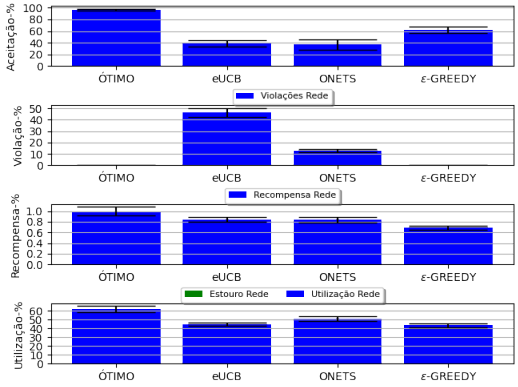
do algoritmo eUCB. Considerar a recompensa recebida por um modelo ótimo de otimização matemática é fundamental porque a maioria dos problemas de otimização tem como objetivo encontrar a melhor solução possível dentre as várias alternativas possíveis. A recompensa representa uma medida de desempenho do modelo, que, em nosso caso, deve ser maximizada, buscando atingir o objetivo da otimização. Portanto, avaliar a recompensa recebida pelo modelo ótimo é crucial para determinar se o resultado da otimização atende aos critérios desejados. Além disso, a recompensa é uma métrica importante de ser utilizada para comparar os resultados no modelo ótimo com os diferentes modelos propostos e selecionar aquele que fornece a melhor solução possível para o problema em questão. Nesse contexto, o algoritmo ONETS obtém recompensas muito próxima à recompensa recebida pelo algoritmo ϵ-Greedy. Analisando os gráficos de utilização de recursos, observamos que, devido às características implícitas à aplicação de vídeo, a utilização dos recursos de rede é grande nos cenários de vídeo. Em geral, para os cenários de vídeo, o cenário de demanda mínima (Figura 5.1(c)) permite melhor utilização dos recursos e maiores recompensas que os demais cenários.



(a) *Inquilinos de vídeo com demanda variada.*



(b) *Inquilinos de vídeo com demanda máxima*



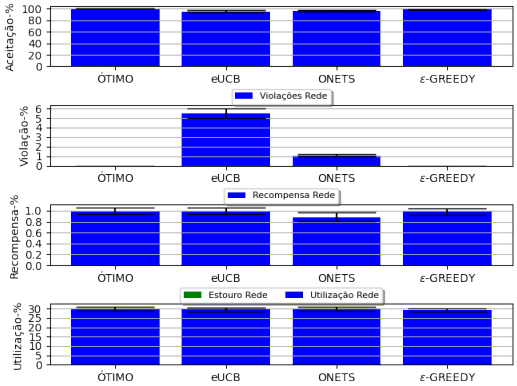
(c) *Inquilinos de vídeo com demanda mínima.*

Figura 5.1: Avaliação de desempenho dos algoritmos estudados para um conjunto de inquilinos do tipo Vídeo.

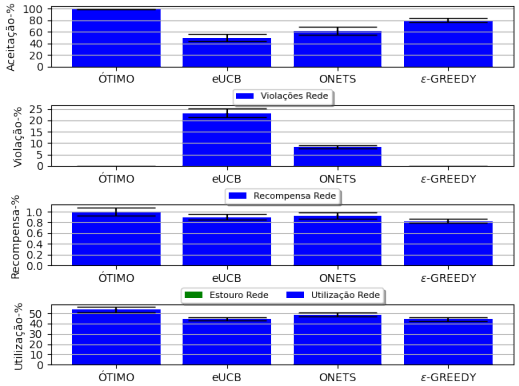
5.5.2 Cenário de Mapas

Na Figura 5.2 apresentamos os resultados para os cenários Mapas 1, Mapas 2 e Mapas 3 para o problema estudado no Capítulo 3. Nesses cenários, todos os inquilinos que requisitam criação de *slices* oferecem serviços de mapa. Entretanto, os *templates* requisitados variam em cada cenário. No cenário Mapas 1, os inquilinos requisitam *templates* de mapas com demanda variada (valores entre a demanda máxima e a demanda mínima). No cenário Mapas 2, todos os inquilinos requisitam *templates* de mapa com demanda Máxima, havendo maior demanda em termos de recursos de comunicação, em relação aos demais cenários de mapa. No cenário Mapas 3, todos os inquilinos requisitam *templates* de mapa com demanda mínima. Logo, esse cenário apresenta a menor demanda em termos de recursos de comunicação, em relação aos demais cenários de mapa.

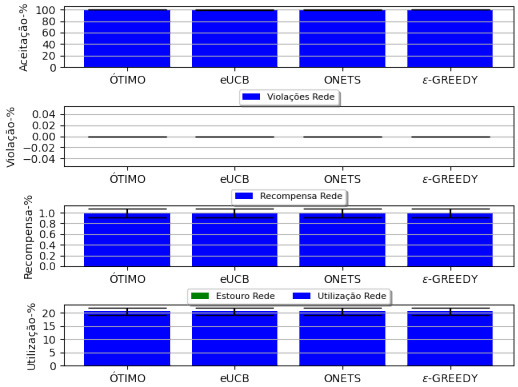
As Figuras 5.2(a), 5.2(b) e 5.2(c) apresentam as taxas médias de aceitação, as taxas médias de violação, as recompensas médias e as taxas médias de utilização da infraestrutura para os cenários Mapas 1, Mapas 2 e Mapas 3, respectivamente. Analisando os algoritmos estudados neste trabalho, observamos novamente que a taxa de aceitação média das soluções atinge em torno de 89% para o ÓTIMO, para os cenários com demandas variadas e mínimas, ao passo que para cenários com demanda Máxima, o modelo foi capaz de acomodar 100% das solicitações recebidas. eUCB, ONETS e ϵ-Greedy apresentam taxas de aceitação de 76%, 79% e 83% para demandas variada e mínima, respectivamente. As Figuras apresentam ainda que apenas ONETS viola o SLA das aplicações para cenários de demanda variada e mínima, ao passo que ONETS e eUCB incorrem em violações em cenário de demanda máxima. A recompensa média obtida pelo ÓTIMO é maior que as obtidas pelos algoritmos estudados, nos três cenários considerados. Analisando o gráfico de utilização de recursos, observamos que, deferentemente dos cenários de Vídeo, os cenários de Mapas necessitam de uma quantidade menor de recursos de computação. Podemos observar que, no cenário de maior demanda, a utilização dos recursos chega próximo de 50% para os algoritmos estudados, enquanto que em cenários de demanda variada e mínima, a utilização não ultrapassa o limiar de 30%.



(a) *Inquilinos de mapas com demanda variada.*



(b) *Inquilinos de mapas com demanda máxima.*



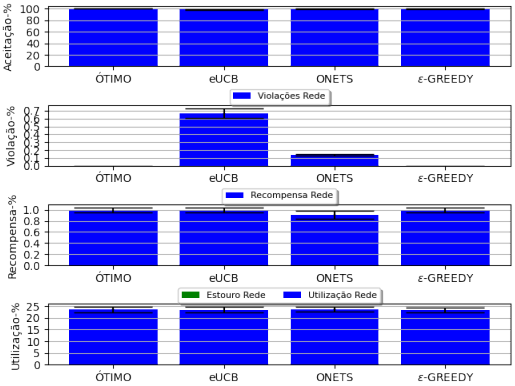
(c) *Inquilinos de mapas com demanda mínima.*

Figura 5.2: Avaliação de desempenho dos algoritmos estudados para um conjunto de inquilinos do tipo Mapas.

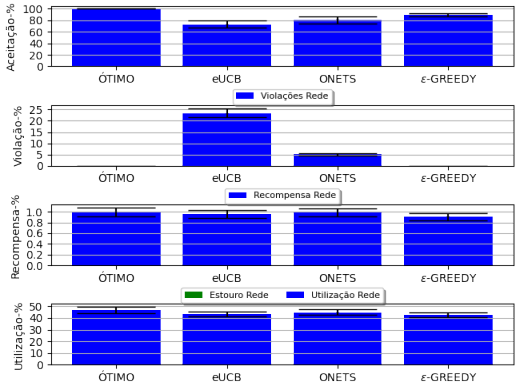
5.5.3 Cenário de Jogos

Na Figura 5.3 apresentamos os resultados para os cenários Jogos 1, Jogos 2 e Jogos 3. Nesses cenários, todos os inquilinos oferecem serviços de jogos. Entretanto, os recursos dos *templates* requisitados variam em cada cenário. No cenário Jogos 1, os inquilinos requisitam *templates* de jogos com demanda variada. No cenário Jogos 2, todos os inquilinos requisitam *templates* de jogos com demanda Máxima, e no cenário Jogos 3, todos os inquilinos requisitam *templates* de jogos com demanda mínima.

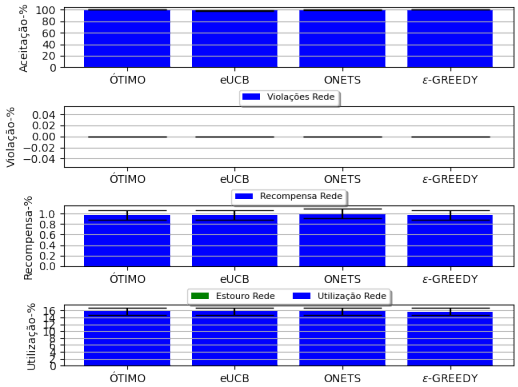
As Figuras 5.3(a), 5.3(b) e 5.3(c) apresentam a taxa de aceitação, violação, recompensa e a utilização dos recursos de comunicação. Analisando os algoritmos estudados nesta tese, observamos que a taxa de aceitação varia entre 78% e 85%, atingindo uma satisfação mais alta quando comparada aos cenários anteriores analisados. Para o cenário Jogos, apenas ONETS incorre em violações de SLA das aplicações, em torno de 10% para todos os casos analisados. A recompensa média obtida por eUCB, ONETS e ϵ-Greedy se aproximam no ÓTIMO. Por fim, todos os algoritmos utilizaram em torno de 30% dos recursos computacionais, para todos os cenários analisados.



(a) *Inquilinos de jogos com demanda variada.*



(b) *Inquilinos de jogos com demanda máxima.*



(c) *Inquilinos de jogos com demanda mínima.*

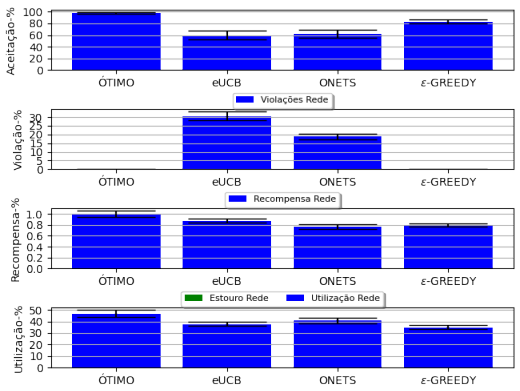
Figura 5.3: Avaliação de desempenho dos algoritmos estudados para um conjunto de inquilinos do tipo Jogos.

5.5.4 Cenário Variado

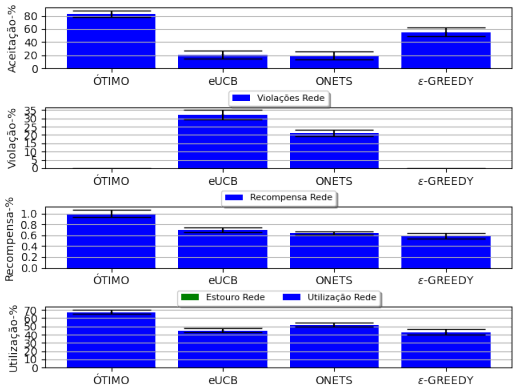
Na Figura 5.4 apresentamos os resultados para os cenários Variado 1, 2 e 3. Nesses cenários, inquilinos que requisitam criação de *slices* oferecem serviços variados. Os *templates* requisitados variam em cada cenário. No cenário Variado 1, inquilinos requisitam *templates* com demanda variada. Esse é o cenário com maior heterogeneidade entre todos. No cenário Variado 2, inquilinos requisitam *templates* de jogos com demanda Máxima, e em Variado 3, inquilinos requisitam *templates* de jogos com demanda mínima.

As Figuras 5.4(a), 5.4(b) e 5.4(c) apresentam a aceitação, violações, recompensa e a utilização média da infraestrutura, respectivamente. Analisando os algoritmos, podemos observar que em um cenário de inquilinos variados com demanda máxima, a taxa de aceitação de eUCB e ONETS é de 20%, enquanto que ÓTIMO e ϵ-Greedy alcançam 80% e 57%, respectivamente. Essa diferença na recompensa é adicionada ao número de violações de eUCB e ONETS, próximo a 35% e 25%, respectivamente. Além disso, os algoritmos estudados alcançam no máximo 70% da recompensa do ÓTIMO. Esses resul-

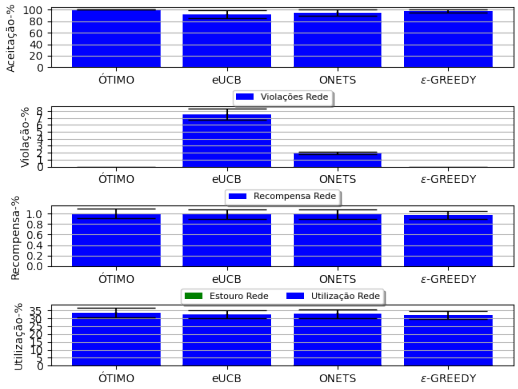
tados são importantes pois ilustram cenários mais próximos dos reais, onde a demanda por recursos de comunicação é bastante heterogêneo.



(a) *Inquilinos variados com demanda variada.*



(b) *Inquilinos variados com demanda máxima.*



(c) *Inquilinos variados com demanda mínima.*

Figura 5.4: Avaliação de desempenho dos algoritmos para inquilinos de serviços variados.

5.6 Resultados dos Algoritmos Propostos

Nesta seção, realizamos a análise dos resultados dos algoritmos propostos no Capítulo 4, que levam em consideração tanto os recursos de comunicação quanto os recursos de computação necessários para atender as demandas de transmissão e processamento das aplicações dos inquilinos no sistema. Os gráficos apresentam a comparação entre os resultados dos algoritmos ÓTIMO, MONETS-OBD, MONETS-OBS e CAONS. Ademais, realizamos uma comparação dos resultados obtidos com o algoritmo ONETS e discutimos nossas conclusões.

5.6.1 Cenário Vídeo

Nesta seção, apresentamos os resultados obtidos para os cenários de Vídeo 1, Vídeo 2 e Vídeo 3, referentes ao problema formulado no Capítulo 4. A Figura 5.5 ilustra cada um dos três cenários mencionados. Na subfigura 5.5(a), todos os inquilinos que requisitam *slices* oferecem serviços de vídeo por *streaming* com demandas variando de máxima a mínima. Esse cenário apresenta a maior heterogeneidade entre todos. Na subfigura 5.5(b), todos os inquilinos apresentam a maior demanda da classe de serviço de vídeo. Por fim, na subfigura 5.5(c), todos os inquilinos apresentam a menor demanda da classe de serviço de vídeo.

Cada sub figura é composta por quatro gráficos. O primeiro apresenta a taxa de aceitação média de inquilinos no sistema. O segundo, apresenta a taxa média de violação das capacidades máximas de recursos de comunicação e computação. O gráfico 3 apresenta a recompensa média recebida, e o último gráfico mostra a taxa média de utilização dos recursos de comunicação e computação. Os resultados de recompensa para os algoritmos propostos são exibidos em duas barras empilhadas, representando a recompensa média recebida pela utilização dos recursos de computação e comunicação em verde claro e verde escuro, respectivamente. É importante mencionar que a demanda instantânea dos inquilinos do sistema é gerada por meio de três distribuições distintas: gaussiana, uniforme e lognormal, agregando características de aleatoriedade aos diferentes tipos de inquilinos analisados. Essas distribuições descrevem a demanda instantânea do sistema exercida pelos inquilinos que receberam *slices*.

Ao analisarmos a Figura 5.5(a), podemos constatar que o algoritmo ONETS apresentou a maior taxa de aceitação, conseguindo acomodar quase 100% das requisições para as três distribuições de demanda analisadas. Em contrapartida, MONETS-OBD e MONETS-OBS tiveram as menores taxas de aceitação, embora tenham ficado muito próximas das obtidas pelo ÓTIMO. CAONS, que implementa penalização, obteve resultados mais próximos ao ÓTIMO. Além disso, é importante observar que a demanda instantânea gerada pela distribuição lognormal obteve as menores taxas de aceitação para todos os algoritmos analisados. Esse fato pode ser explicado pelo comportamento dessa distribuição, que apresenta uma cauda longa, tornando inviável a alocação de novos inquilinos quando grande parte dos recursos está em uso.

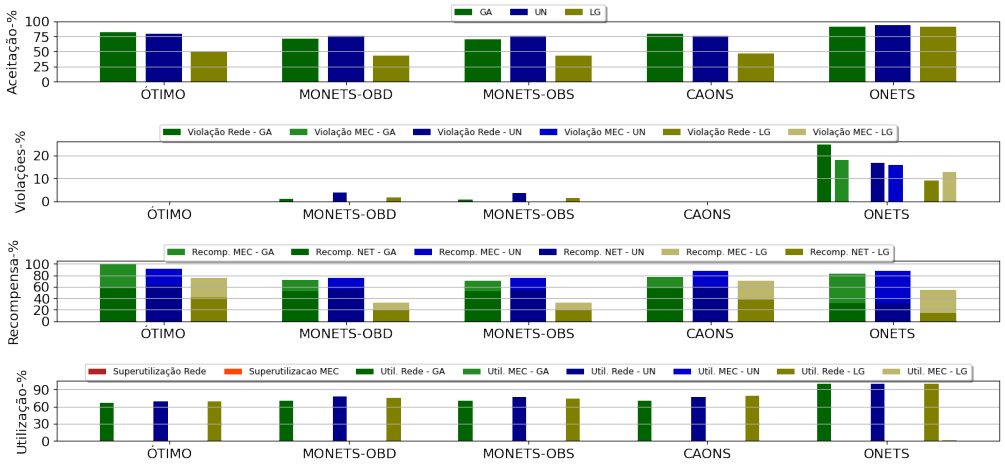
No gráfico subsequente, podemos perceber que ÓTIMO e CAONS não geraram violações da capacidade de recursos no sistema, diferentemente de MONETS-OBD e MONETS-OBS, que apresentaram uma violação média dos recursos de comunicação de 4% e 5%, respectivamente, para uma distribuição de demanda uniforme. ONETS, por outro lado, apresentou a maior taxa de violação dos recursos de comunicação, chegando a quase 23%, 17% e 9% para as distribuições gaussiana, uniforme e lognormal, respectivamente. Nos gráficos de recompensa, podemos perceber que o ÓTIMO alcança

as maiores recompensas do sistema para as três distribuições analisadas. Além disso, CAONS e ONETS são os algoritmos que mais se aproximam dos resultados obtidos pelo ÓTIMO, seguidas de MONETS-OBD e MONETS-OBS. No entanto, vale ressaltar que, para atingir recompensas mais altas, ONETS violou os recursos várias vezes, conforme demonstrado anteriormente. No gráfico de utilização, podemos observar que ONETS alcançou a maior utilização do sistema, com valores próximos de 92% para as três distribuições de demanda analisadas. ÓTIMO, por sua vez, apresentou as menores taxas de utilização do sistema, seguido por CAONS, MONETS-OBD e MONETS-OBS.

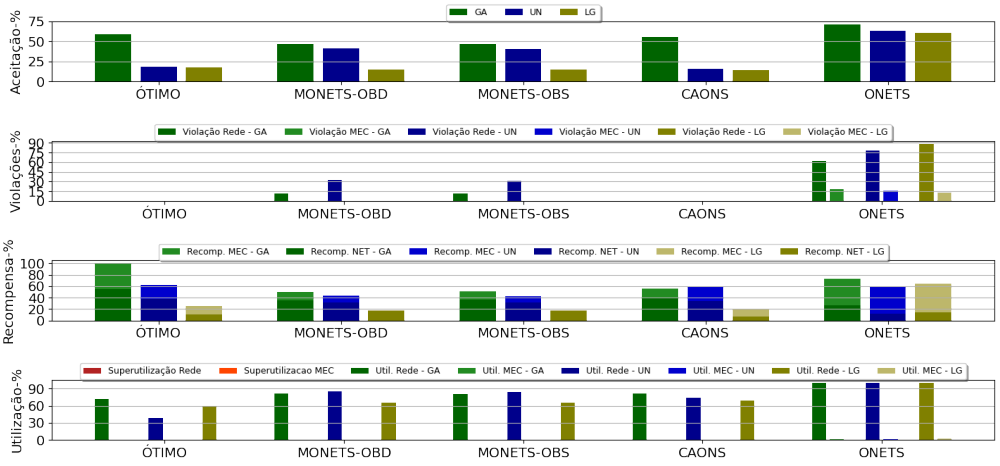
Ao analisar a Figura 5.5(b), podemos observar que ONETS apresenta a maior taxa de aceitação, seguido por ÓTIMO, CAONS, MONETS-OBD e MONETS-OBS. No entanto, é importante destacar que, nesse cenário específico, as taxas de aceitação são as menores dentre o conjunto de inquilinos do tipo vídeo. Isso ocorre porque a alta demanda gerada pelos inquilinos que sempre utilizam a demanda máxima inviabiliza, em parte, a estratégia de *overbooking* adotada pelos algoritmos, reduzindo a taxa de aceitação. De maneira semelhante, nos cenários com a distribuição lognormal representando a distribuição instantânea, também obtivemos as menores taxas de aceitação. Esse resultado pode ser explicado pelo comportamento da distribuição, que possui uma cauda longa, o que inviabiliza o aceite de novos inquilinos enquanto grande parte dos recursos estiver alocada.

No subgráfico seguinte, de violações, podemos perceber que, mais uma vez, ÓTIMO e CAONS não incorrem em violações no sistema, ao contrário de MONETS-OBD e MONETS-OBS, que resultam em uma violação média dos recursos de comunicação de 4% e 5%, respectivamente, para uma distribuição de demanda uniforme. Já o ONETS atinge níveis maiores de violações, próximos de 88% para os recursos de comunicação em uma distribuição lognormal. Esse resultado ajuda a demonstrar o motivo pelo qual ONETS alcança recompensas maiores que o ÓTIMO, chegando a 63%, enquanto o ÓTIMO alcança 27% para a distribuição lognormal. Observando o gráfico de utilização, podemos perceber que ONETS utiliza próximo de 94% dos recursos computacionais, em média. Essa alta taxa de utilização dos recursos de comunicação demonstra novamente a tendência do algoritmo em violar a quantidade de recursos disponíveis. Apesar de não se tratar de uma aplicação sensível a atrasos, violar continuamente os recursos de comunicação e computacionais em uma aplicação de vídeo por *streamming* pode resultar em atrasos na entrega dos dados ao usuário, comprometendo o QoE da aplicação.

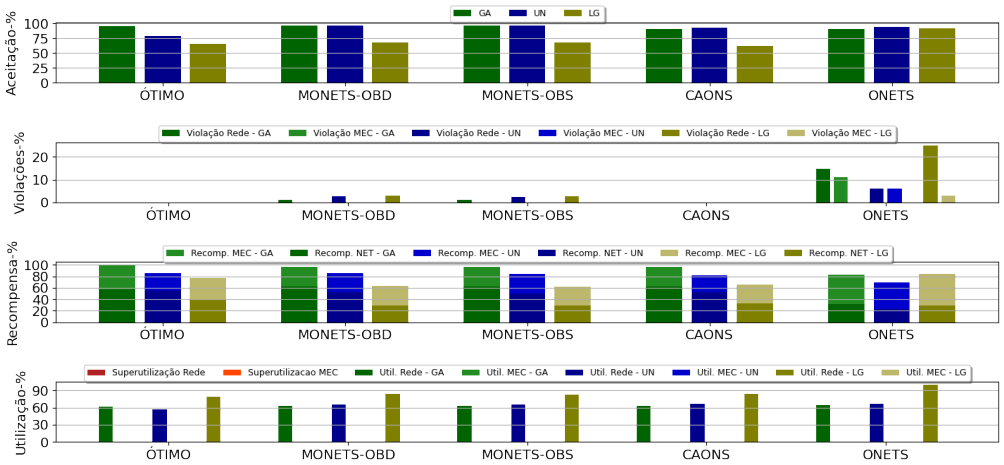
Na Figura 5.5(c), todos os inquilinos oferecem serviços de vídeo, utilizando a demanda mínima. Nos gráficos de aceitação, podemos perceber que ONETS alcança as maiores taxas de aceitação para as três distribuições de demanda analisadas: 91%, 97%, e 95% para as distribuições gaussiana, uniforme, e lognormal, respectivamente. MONETS-OBD, MONETS-OBS, e CAONS obtêm taxas de aceitação próximas ao ÓTIMO.



(a) *Inquilinos de vídeo com demanda variada.*



(b) *Inquilinos de vídeo com demanda máxima.*



(c) *Inquilinos de video com demanda mínima.*

Figura 5.5: Avaliação de desempenho dos algoritmos para as distribuições de demanda com gaussiana, uniforme e lognormal e inquilinos de serviços de vídeo.

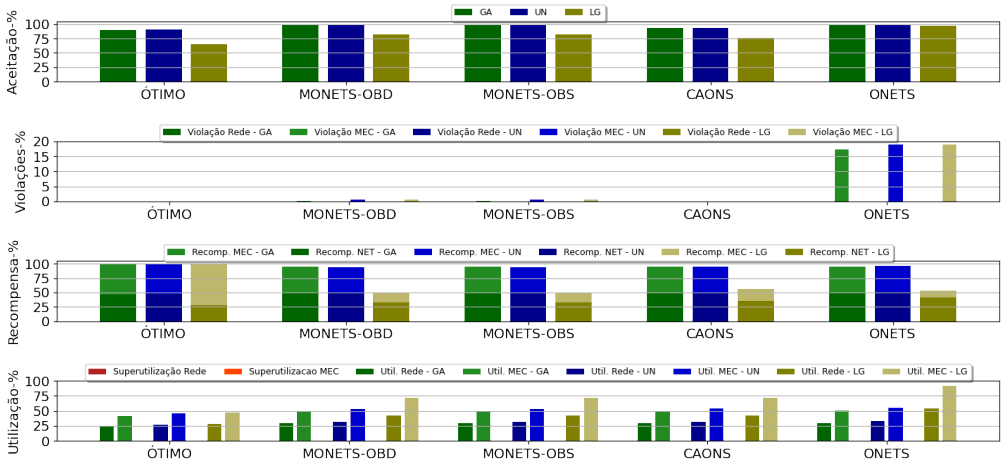
Para o cenário com demandas mínimas, ONETS alcança as maiores taxas de violação de SLA, chegando a 26% de violações dos recursos de comunicação para a distribuição de demanda lognormal. MONETS-OBD e MONETS-OBS também resultam em violações, próximas de 4% para os cenários analisados. Por outro lado, ÓTIMO obtém as maiores recompensas, exceto para o caso da distribuição lognormal, em que ONETS retorna a maior recompensa. Novamente, podemos explicar a maior recompensa obtida por ONETS em função do maior número de violações de SLA ocorridas.

Observando o gráfico de utilização, podemos perceber que ONETS utiliza os recursos computacionais em grande quantidade, atingindo valores próximos a 94% para a distribuição de demanda baseada em lognormal. No entanto, um nível tão alto de utilização dos recursos de comunicação pode levar a violações contínuas, prejudicando a entrega de dados ao usuário e comprometendo o QoE da aplicação de vídeo por *streaming*.

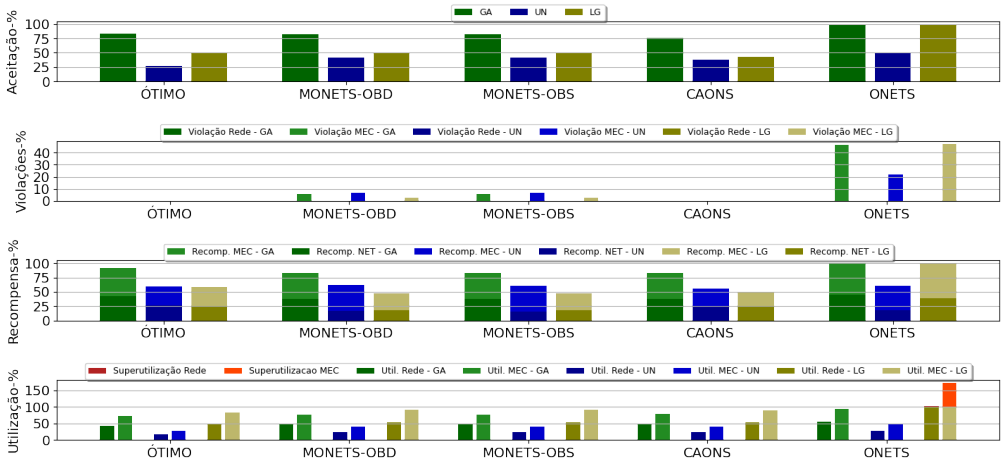
5.6.2 Cenário Mapas

Esta seção apresenta uma análise dos resultados obtidos para os cenários de Mapas 1, Mapas 2 e Mapas 3, no contexto do problema apresentado no Capítulo 4. Em cada um desses cenários, os inquilinos solicitam a criação de *slices* para serviços de mapa e geolocalização, com variações nos *templates* requisitados. No cenário Mapas 1, os inquilinos solicitam *templates* de mapa com demanda variável, que assume valores entre a demanda máxima e a demanda mínima. Já no cenário Mapas 2, todos os inquilinos solicitam *templates* com demanda máxima, o que o torna o cenário com maior demanda de recursos. Por fim, no cenário Mapas 3, todos os inquilinos solicitam *templates* de mapa com demanda mínima, tornando este cenário o de menor demanda em termos de recursos em comparação aos demais cenários de mapas.

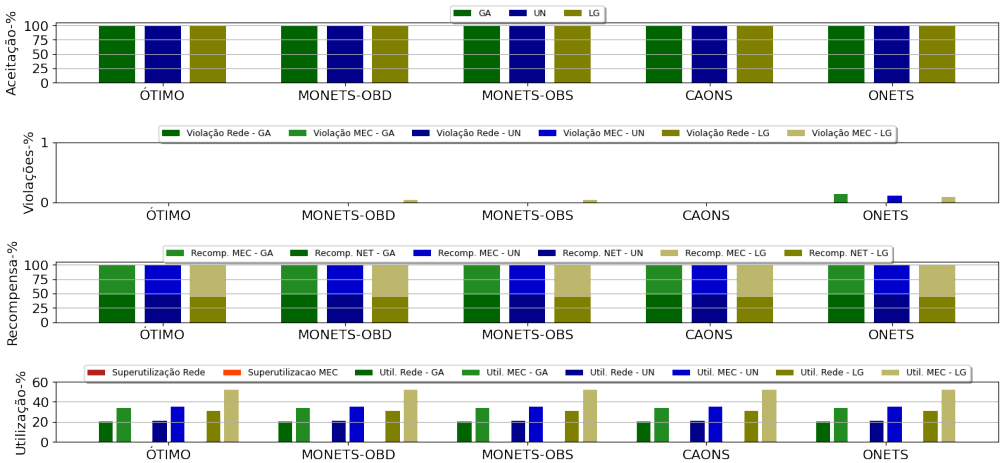
As Figuras 5.6(a), 5.6(b) e 5.6(c) apresentam a taxa média de aceitação, a taxa média de violação, a recompensa média e a taxa média de utilização da infraestrutura para os cenários Mapas 1, Mapas 2 e Mapas 3. Analisando a Figura 5.6(a), podemos perceber que ONETS alcança as maiores taxas de aceitação, para todas as três distribuições de demanda dos inquilinos utilizadas. Por outro lado, MONETS-OBD e MONETS-OBS também apresentam taxas de aceitação maiores que ÓTIMO, contudo, com valores próximos. CAONS é a que mais se aproxima de ÓTIMO, para as três distribuições utilizadas. O gráfico de violações demonstra que MONETS-OBD e MONETS-OBS resultam em cerca de 2% de violações durante a simulação. ÓTIMO e CAONS não incorrem em violações. ONETS apresenta uma taxa média de violação de 16%, 18% e 19% para as distribuições gaussiana, uniforme e lognormal, respectivamente.



(a) *Inquilinos de mapas com demanda variada.*



(b) *Inquilinos de mapas com demanda máxima.*



(c) *Inquilinos de mapas com demanda mínima.*

Figura 5.6: Avaliação de desempenho dos algoritmos para as distribuições de demanda com gaussiana, uniforme e lognormal e inquilinos de serviços de mapas.

A recompensa recebida por ÓTIMO é a maior entre os algoritmos analisados, para as três distribuições utilizadas. Os demais algoritmos se aproximam de ÓTIMO para as distribuições gaussiana e uniforme. Contudo, para a distribuição lognormal, CAONS, que mais se aproxima de ÓTIMO, obtém apenas 54% da recompensa de ÓTIMO, evidenciando que, para uma mesma solução proposta, o comportamento da utilização dos recursos exercida pelo inquilino, impacta diretamente na qualidade da solução proposta. Essa realidade se repete no comportamento dos demais algoritmos. Diferentemente do cenário anterior analisado, de vídeo, aplicações de mapas requerem uma quantidade maior de processamento de dados, apresentando, para ONETS, por exemplo, uma utilização de cerca de 96% dos recursos computacionais disponíveis para uma distribuição de demanda baseada em uma lognormal no cenário analisado. ÓTIMO não ultrapassa a margem de 50% de utilização dos recursos, tanto computacionais, quanto de comunicação. MONETS-OBD, MONETS-OBS e CAONS matém uma tendência semelhante, exceto pelos recursos de computação para uma demanda dada pela distribuição lognormal, que alcança a margem de 75% de utilização da infraestrutura.

Analisando a Figura 5.6(b) podemos perceber que ONETS alcança novamente as maiores taxas de aceitação, dessa vez ao custo de uma violação média de 48%, 22% e 49% para as demandas baseadas em gaussiana, uniforme e lognormal, respectivamente. ÓTIMO apresenta a maior taxa de aceitação, em torno de 77% para uma demanda baseada em uma distribuição gaussiana, seguida da lognormal, aceitando 50% das requisições realizadas e por último, a uniforme, aceitando 27% das requisições de criação de *slices* recebidas. MONETS-OBD, MONETS-OBS e CAONS segue a tendência de aceitação de ÓTIMO, apresentando a menor taxa de aceitação para a distribuição uniforme. Esse resultado demonstra mais uma vez que a qualidade da solução depende, além do algoritmo proposto, mas também do comportamento dos inquilinos no sistema. MONETS-OBD e MONETS-OBS resultam em violações em menos de 2% dos casos analisados. ÓTIMO e CAONS não incorrem em violações.

Ainda na figura 5.6(b) podemos observar que ÓTIMO alcança as maiores recompensas do sistema, seguida de CAONS, MONETS-OBD e MONETS-OBS. ONETS incorre em recompensas próximas ao ÓTIMO, e superior ao ÓTIMO para o caso da demanda baseada em uma distribuição lognormal. Contudo, conforme discutido acima, ONETS alcança essa recompensa ao custo de uma violação dos recursos em 49% das vezes. Além disso, ONETS superestima os recursos computacionais, necessitando de uma capacidade de cerca de 50% a mais do disponível para tornar o conjunto de soluções apresentado válido. Aplicações de mapas e geolocalização, como por exemplo *Vehicle-to-vehicle communication's* (V2V) tem a característica de serem sensíveis ao atraso. Superestimar os recursos computacionais disponíveis no sistema poderia gerar um gargalo de processamento das aplicações, podendo resultar no comprometimento do serviço.

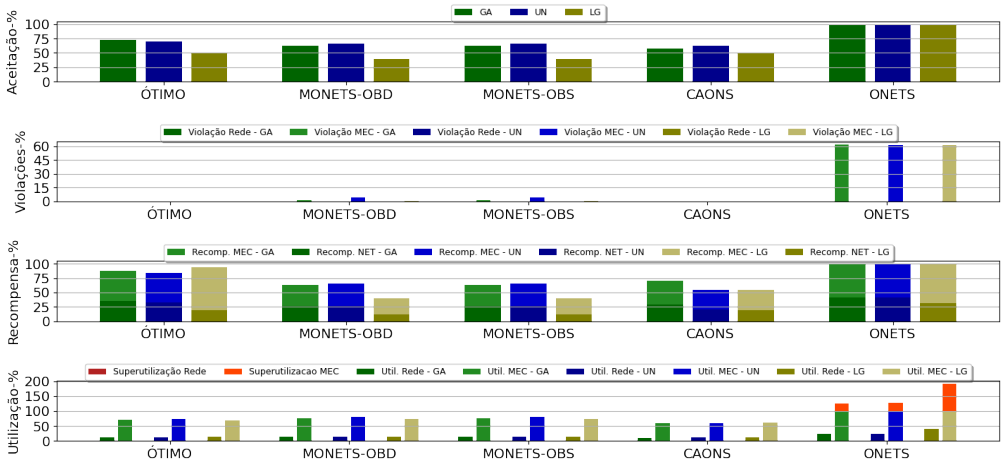
Na Figura 5.6(c), a taxa de aceitação se mantém próxima de 100% para todos os algoritmos e todas as distribuições analisadas. ONETS incorre em violações dos recursos computacionais em um número de vezes, próximo de 0,3%. As recompensas recebidas pelos algoritmos se aproximam muito do ÓTIMO. A tendência de utilização dos recursos de comunicação e computação também se mantém próxima do ÓTIMO. Podemos perceber neste cenário que a quantidade de inquilinos e requisições realizadas neste cenário não é suficiente para promover a competição por recursos, possibilitando que todos os algoritmos acomodem as requisições de maneira eficiente.

5.6.3 Cenário Jogos

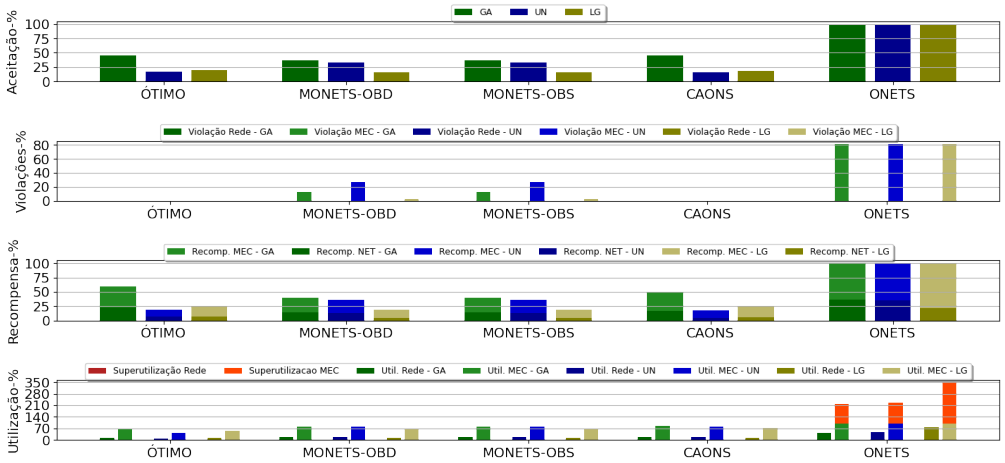
Na Figura 5.7, são apresentados os resultados dos cenários Jogos 1, Jogos 2 e Jogos 3, nos quais todos os inquilinos disponibilizam serviços de jogos, mas com diferentes requisitos de recursos em cada cenário. No cenário Jogos 1, os inquilinos solicitam *templates* de jogos com demanda variável, enquanto que, no cenário Jogos 2, todos os inquilinos solicitam *templates* de jogos com demanda máxima e, no cenário Jogos 3, todos os inquilinos solicitam *templates* de jogos com demanda mínima. As Figuras 5.7(a), 5.7(b) e 5.7(c) apresentam a taxa de aceitação, taxa de violação, recompensa e taxa de utilização média dos recursos de comunicação e computação.

A Figura 5.7(a) apresenta resultados os para os diferentes algoritmos avaliados. De forma geral, ONETS se destaca por alocar 100% das solicitações de *slices templates* no sistema, mas esse aparente bom resultado vem acompanhada de alto custo alto. O algoritmo incorre em violação dos recursos computacionais em cerca de 60% dos casos. Além disso, ONETS superestima os recursos computacionais necessários, requerendo um serviço de computação 2X maior do que o disponível para uma demanda de inquilinos baseada em uma lognormal. O ÓTIMO alcança, no melhor caso, uma taxa de aceitação de 75% das requisições recebidas, para uma distribuição de demanda gaussiana e não incorre em violações da capacidade do sistema. MONETS-OBD e MONETS-OBS, por sua vez, obtêm taxas de aceitação semelhantes ao ÓTIMO, incorrendo em violações em menos de 4% dos casos para uma distribuição uniforme. Por fim, CAONS atinge taxas de aceitação próximas ao ÓTIMO e não resulta em violações da capacidade do sistema.

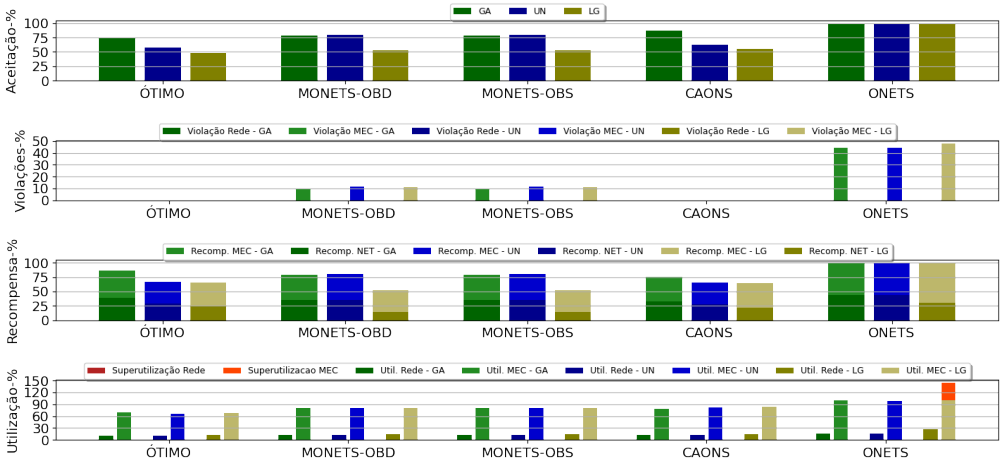
Ao analisar a Figura 5.7(b), podemos observar que ONETS novamente atinge uma alocação de 100% das solicitações de *slices templates* recebidas pelo sistema. No entanto, essa alocação é alcançada à custa de violações de recursos computacionais em cerca de 80% dos casos. Além disso, ONETS superestima os recursos computacionais, exigindo um serviço de computação 250% maior do que o disponível para uma demanda de inquilinos que apresentam o comportamento dado por uma distribuição lognormal. O ÓTIMO, por outro lado, consegue acomodar apenas 48% das requisições recebidas no



(a) *Inquilinos de jogos com demanda variada.*



(b) *Inquilinos de jogos com demanda máxima.*



(c) *Inquilinos de jogos com demanda mínima.*

Figura 5.7: Avaliação de desempenho dos algoritmos para as distribuições de demanda com gaussiana, uniforme e lognormal e inquilinos de serviços de jogos.

melhor cenário, com uma distribuição de demanda gaussiana. No entanto, esse método não incorre em violações das capacidades do sistema. MONETS-OBD e MONETS-OBS apresentam taxas de aceitação semelhantes ao ÓTIMO, mas com uma média de 22% de violações em casos de distribuição uniforme. CAONS atinge taxas de aceitação próximas ao ÓTIMO e não resulta em violações de capacidade. Embora os jogos não sejam aplicações sensíveis, eles exigem recursos computacionais significativos que nem sempre estão disponíveis no dispositivo do usuário. Por isso, muitas vezes, eles precisam ser processados remotamente em serviços de borda ou em nuvem remota. Se ocorrerem atrasos no processamento dessas aplicações, o QoE do usuário pode ser comprometido, afetando negativamente a aplicação.

Ao analisarmos os gráficos da Figura 5.7(c), podemos observar que ONETS alcança alocações de 100% das demandas recebidas, mas ao custo de uma violação média de 48% dos casos durante a simulação. Isso significa que em determinado instante, os inquilinos aceitos no sistema, requisitaram, ao mesmo tempo, uma quantidade de recursos computacionais maior que a quantidade disponível. Além disso, apesar de oferecer as maiores recompensas, ONETS superestima os recursos computacionais necessários em 50% a mais do que os disponíveis, no caso de demandas com comportamento de lognormal. O ÓTIMO, por sua vez, é capaz de acomodar 75%, 53% e 49% das solicitações recebidas para as distribuições gaussiana, uniforme e lognormal, respectivamente. MONETS-OBD e MONETS-OBS apresentam taxas de aceitação superiores às de ÓTIMO, principalmente para a distribuição uniforme, mas incorrem em violações dos recursos computacionais em cerca de 11% das vezes durante a simulação. Por outro lado, CAONS se aproxima da taxa de aceitação de ÓTIMO sem incorrer em violações. As recompensas obtidas por MONETS-OBD e MONETS-OBS são próximas às de ÓTIMO, exceto para a distribuição de demanda uniforme. Como já mencionado anteriormente, essas dois algoritmos apresentam taxas de aceitação maiores, resultado em recompensas maiores, porém ao custo da ocorrência de violações dos recursos disponíveis. Por fim, a recompensa obtida por CAONS se aproxima de ÓTIMO sem a ocorrência de violações médias no sistema, e o comportamento da utilização do sistema se mantém semelhante entre ÓTIMO, MONETS-OBD, MONETS-OBS e CAONS.

5.6.4 Cenário Variado

A Figura 5.8 ilustra os resultados obtidos nos cenários Variado 1, Variado 2 e Variado 3, nos quais inquilinos que requisitam criação de *slices* oferecem serviços variados e com diferentes demandas de recursos. Em cada cenário, os *templates* requisitados são diferentes. No cenário Variado 1, inquilinos de vídeo, jogos e mapas requisitam *templates* com demanda variada, o que o torna o cenário com maior heterogeneidade. Já no cená-

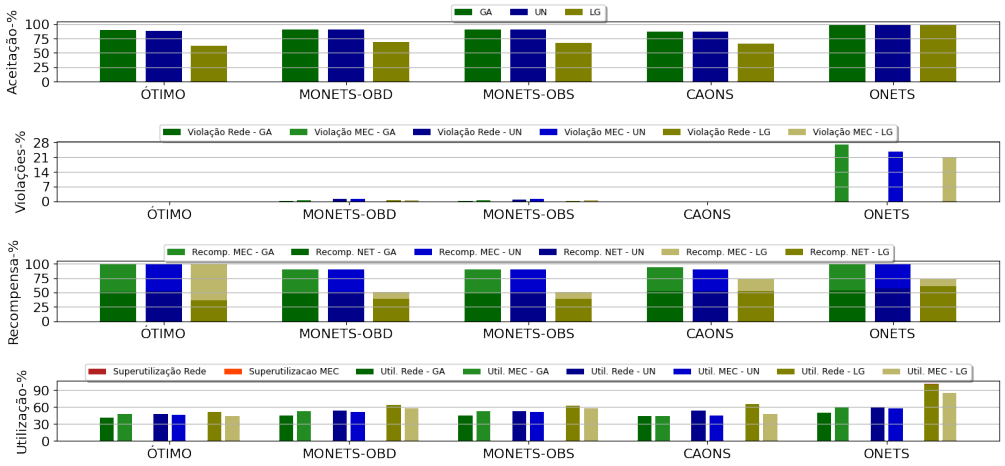
rio Variado 2, os inquilinos desses serviços requisitam *templates* com demanda máxima, enquanto no Variado 3, os inquilinos demandam o mínimo possível para a aplicação. As Figuras 5.8(a), 5.8(b) e 5.8(c) apresentam a taxa média de aceitação, a taxa média de violação, a recompensa média e a utilização média da infraestrutura.

Analisando os gráficos de recompensa, podemos perceber que ONETS alcança as maiores recompensas do sistema, para as três distribuições analisadas. Além disso, CAONS é o que mais se aproxima dos resultados obtidos no algoritmo ÓTIMO, seguidas de MONETS-OBD e MONETS-OBS. No gráfico de utilização podemos identificar o motivo de ONETS alcançar os melhores resultados da análise. ONETS não leva em consideração os recursos computacionais na análise, e por isso superestima a capacidade dos recursos disponíveis. Desse modo, para tornar os resultados de ONETS verdadeiros para o cenário avaliado, seria necessário um aumento de cerca de 20% dos recursos computacionais disponíveis para um cenário variado, com demanda máxima e distribuição uniforme. Esse fato pode inviabilizar a análise fornecida por ONETS para cenários de aplicações reais. Vale ressaltar que os recursos computacionais são onerosos e por muitas vezes escassos na borda, e por isso, tão importantes de serem considerados na análise. ÓTIMO mantém uma utilização média de 61% dos recursos computacionais, ao passo que MONETS-OBD, MONETS-OBS e CAONS utilizam mais recursos para os cenários analisados.

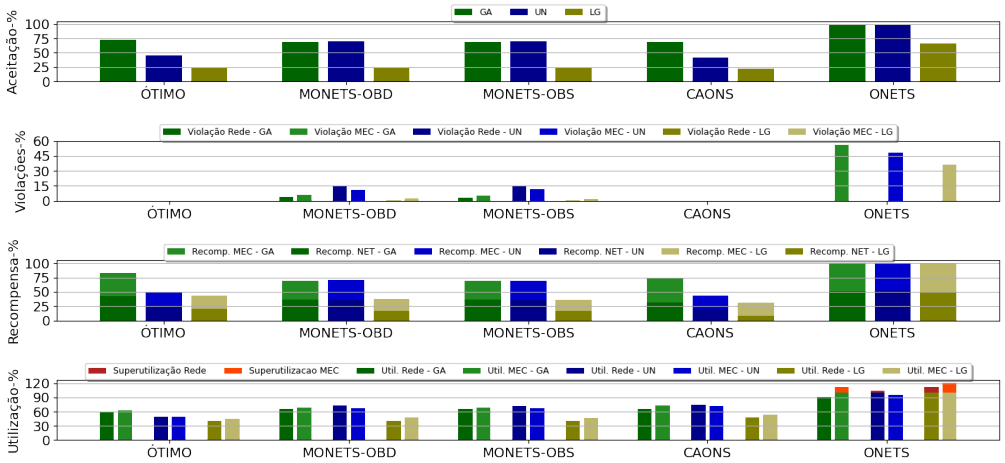
5.6.5 Variação de inquilinos e parâmetros do sistema

Para as próximas análises, retiramos o modelo ÓTIMO do conjunto. Adotamos essa estratégia devido ao custo computacional. Conforme demonstrado no Capítulo 4, o problema é NP-difícil, exigindo tempo exponencial para ser resolvido de maneira ótima. Trabalhamos na Figura 5.9 com conjuntos de dados maiores, incrementando o número de inquilinos. Dessa maneira, a execução do modelo ÓTIMO, para os casos analisados, torna-se impraticável.

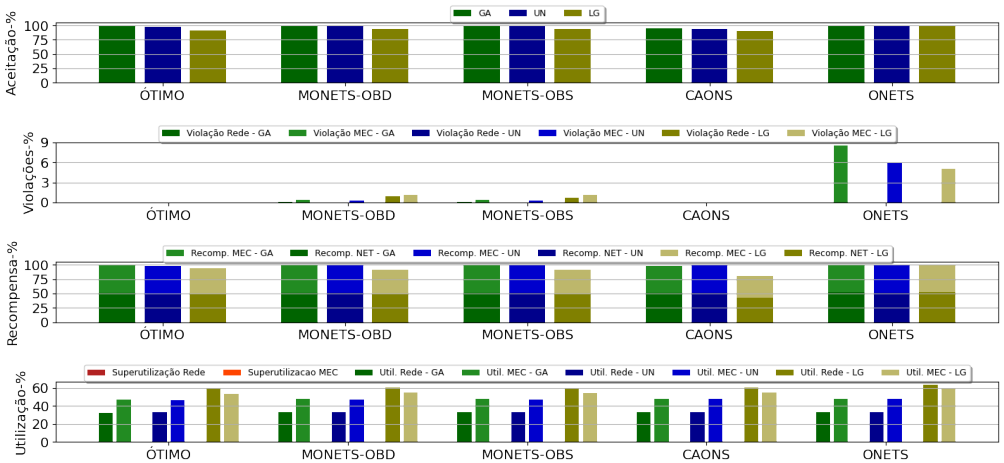
Na Figura 5.9, apresentamos a variação da taxa média de recompensa, taxa média de aceitação, taxa média de utilização e a taxa média da violação de SLA perante o incremento do número de inquilinos no sistema. Na Figura, são apresentadas as variações dos resultados de ONETS, MONETS-OBD, que aqui nos referimos como simplesmente MONETS, e CAONS. Variamos o conjunto de inquilinos $\mathcal{I}$ de $\in [10-70]$ inquilinos. Cada inquilino realiza diversas solicitações de criação de *slices* no sistema ao longo do período de simulação. As simulações foram realizadas para um conjunto variável de usuários com demanda máxima e distribuição de $\lambda_{i,b,t}$ e $\xi_{i,m,t}$ dada por uma gaussiana. Na Figura 5.9(a) apresentamos a variação da recompensa diante do incremento do número de usuários. Além disso, para cada incremento do número de inquilinos, novos cenários são gerados,



(a) *Inquilinos variados com demanda variada.*



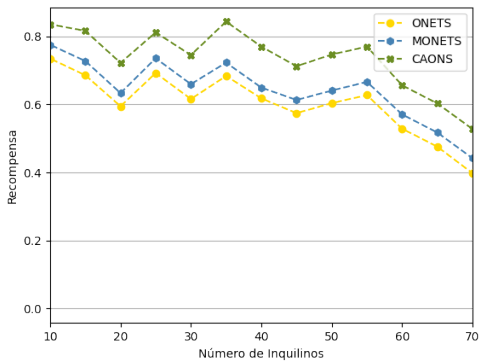
(b) *Inquilinos variados com demanda máxima.*



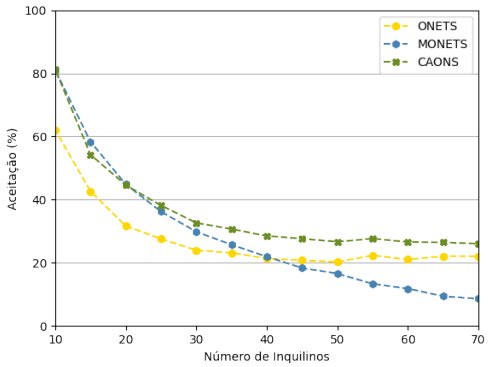
(c) *Inquilinos variados com demanda mínima.*

Figura 5.8: Avaliação de desempenho dos algoritmos para as distribuições de demanda com gaussiana, uniforme e lognormal e inquilinos de serviços variados.

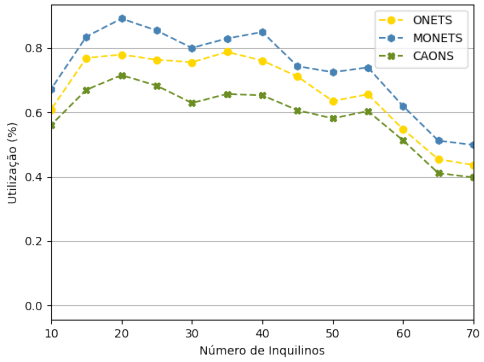
com novas taxas de chegadas de solicitação de criação de slices, com novas demandas de utilização. Esse fato pode explicar o comportamento de crescimento e decrescimento do gráfico de maneira contínua, pois o conjunto de demandas utilizados nas execuções perfazem demandas diferentes, resultando em alocações diferentes ao longo da simulação. A estratégia CAONS obteve as maiores recompensas, mesmo para cenários de maior densidade de inquilinos competindo por recursos. As Figuras 5.9(b) e 5.9(c) demonstram que a aceitação e a utilização de inquilinos no sistema tende a diminuir a medida que o número de inquilinos disputando por recursos aumenta. Essa variação pode ser explicado pela alta demanda por recursos gerada por múltiplos inquilinos variados. Por fim, na Figura 5.9(d) demonstra que o número de violações da estratégia CAONS mantém-se muito próxima a zero, mesmo para cenários de maior densidade de inquilinos.



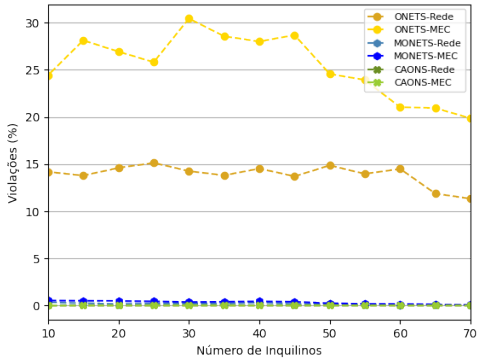
(a) *Recompensa recebida*



(b) *Taxa de aceitação*



(c) *Taxa de utilização*



(d) *Taxa de Violação*

Figura 5.9: Custo da variação do número de inquilinos diante de requisições de criação de *slices* envolvendo aplicações variadas com demanda máxima.

Na Figura 5.10 a recompensa da rede é analisada como função de α e σ. A Figura 5.10(a) mostra que a recompensa recebida na rede mantém-se por toda simulação maior que MONETS e ONETS, varia conforme os valores de α e σ elevam. Os valores mais próximos ao modelo de referência para as heurísticas propostas são obtidas quando os

parâmetros α e σ da função recompensa são iguais a 0.5 e 0.5, respectivamente. Essa variação pode ser explicada pelo modo com que a recompensa foi estabelecida em 4-3 e 4-4. Valores maiores α e σ buscam atender clientes que fazem as maiores reservas na rede, ao passo que valores menores, privilegiam clientes que menos utilizam da quantidade de recursos reservada. Quando os valores de α e σ ultrapassam 0.5, a recompensa é afetada, representando que escolher sempre o cliente que faz as maiores reservas, como um algoritmo guloso faria, por exemplo, não é a melhor opção para o caso. A Figura 5.10(b) demonstra que a aceitação de inquilinos no sistema da estratégia CAONS mantém-se maior que ONETS e MONETS. As Figura 5.10(c) demonstra que a utilização do sistema é mais baixa para CAONS, mesmo alcançando as maiores recompensas. Esse fato pode ser explicado pela escolha mais inteligente que o algoritmo realiza, selecionando inquilinos com utilização mais comportada, e que sofrem menos penalizações. Por fim, a Figura, 5.10(d) demonstra que CAONS mantém o número de violações próximo a zero, mesmo diante da variação do peso da recompensa. Os parâmetro α e σ nesse modelo atuam como um ajuste da solução no cenário analisado, podendo fornecer ao tomador de decisões, dados mais precisos, considerando configurações de rede que capturem situações mais próximas do mundo real, e levando em consideração o tipo de utilização esperada para o sistema.

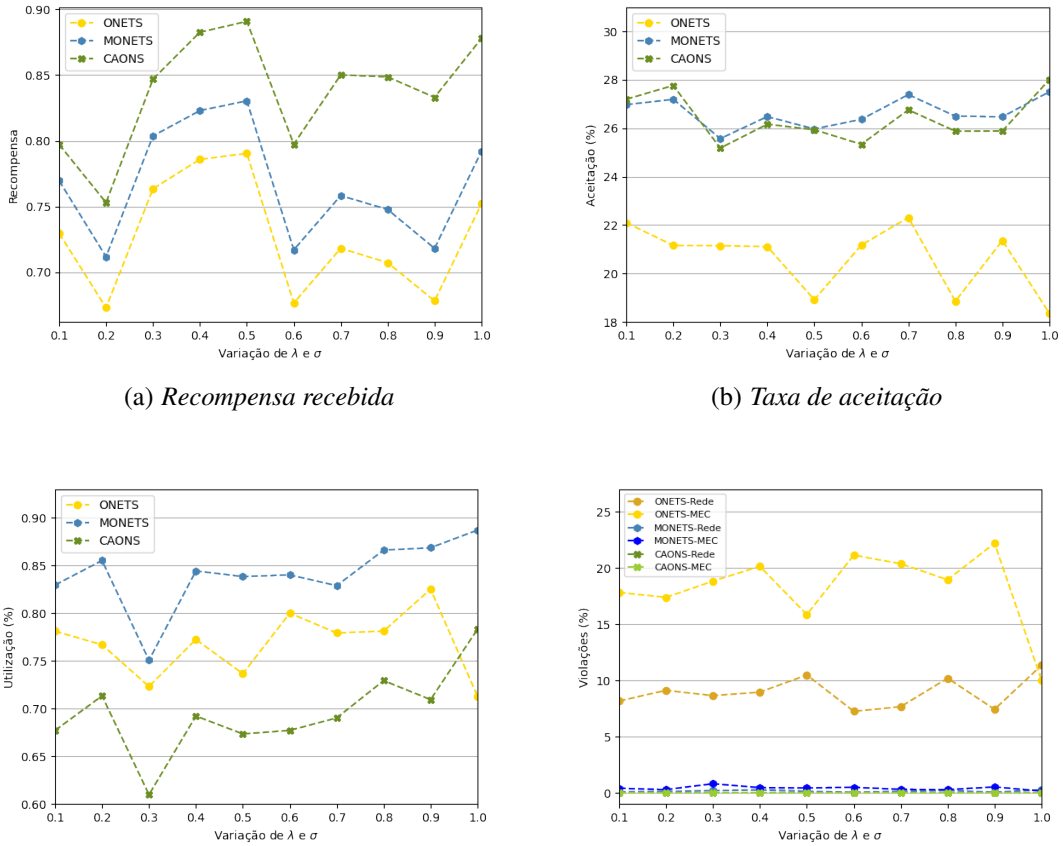
5.6.6 Custo computacional

A Tabela 5.6 apresenta os tempos de execução dos algoritmos propostos nesta tese para diferentes números de inquilinos. Em cada algoritmo, o tempo computacional é medido por 30 execuções.

Tabela 5.6: Carga Computacional

Solução		5 inquilinos	10 inquilinos	15 inquilinos	20 inquilinos
ÓTIMO		6324s	37742s	-	-
eUCB		51658s	108701s	-	-
M-ONETS-OBD		366s	752s	827s	1127s
M-ONETS-OBS		331s	699s	802s	1101s
CAONS		321s	701s	781s	1098s
ONETS		301s	647s	738s	989s
ϵ-Greedy		215s	431s	519s	665s

De acordo com os dados apresentados na tabela 5.6, os algoritmos ÓTIMO e eUCB não são escaláveis, ou seja, o tempo computacional desses algoritmos cresce de forma exponencial, à medida que o número de inquilinos aumenta. Isso torna esses algoritmos inviáveis para análise de cenários com muitos inquilinos, devido à grande dimensão do espaço de busca resultante da consideração dos recursos de comunicação e



(a) *Recompensa recebida*

(b) *Taxa de aceitação*

(c) *Taxa de utilização*

(d) *Taxa de Violação*

Figura 5.10: Custo da variação de α e σ diante de requisições de criação de *slices* envolvendo aplicações variadas com demanda máxima.

computação. Dessa maneira, por questões de viabilidade de tempo de execução, nas simulações com mais de 10 inquilinos, desconsideramos os algoritmos ÓTIMO e eUCB. Observamos que MONETS-OBD, MONETS-OBS, CAONS, ϵ-Greedy e ONETS apresentam soluções sub-ótimas, que reduzem significativamente o tempo de execução quando comparados ao modelo ótimo e são escaláveis, sendo apropriados ao problema abordado nesta tese, mesmo para números maiores de inquilinos no sistema. A Tabela 5.6 fornece evidências de que MONETS-OBD, MONETS-OBS e CAONS apresentam desempenho computacional superior em relação aos outros algoritmos, mesmo quando o número de inquilinos aumenta. Por exemplo, para um conjunto composto por 20 inquilinos, o tempo de execução de uma instância t, ou seja, da necessidade de resposta do controle de admissão, para os três algoritmos propostos (MONETS-OBD, MONETS-OBS e CAONS) é inferior a 1 segundo, ao passo que o tempo de execução de eUCB é superior a 400 segundos. Portanto, para lidar com problemas de decisão de controle de admissão de inquilinos de maneira instantânea, que envolvam um número elevado de inquilinos, os três algoritmos propostos são mais indicadas do que os algoritmos ÓTIMO e eUCB.

5.7 Conclusão

Neste capítulo, apresentamos a avaliação de desempenho dos algoritmos estudados no Capítulo 3 e também dos propostos no Capítulo 4. Descrevemos a infraestrutura, os serviços e os cenários considerados nesta tese. Utilizamos um modelo matemático, desenvolvido com programação linear, para controlar a admissão de inquilinos na infraestrutura da operadora. Além disso, comparamos os algoritmos propostos nesta tese com os algoritmos clássicos de solução do arcabouço MaB. Os resultados mostram que os algoritmos propostos, MONETS-OBD, MONETS-OBS e CAONS, alcançam ganhos significativos de multiplexação, superando soluções não orientadas (ϵ-greedy), soluções que abordam apenas recursos de comunicação (ONETS) e também uma solução orientada (eUCB), para os cenários analisados. A heurística MONETS-OBD alcança taxas de aceitação, recompensa e utilização próximas de 99%, em alguns casos, do cenário ideal, ao custo de ocorrências de violações da capacidade do sistema próximas de 10% no pior caso. MONETS-OBS por outro lado, utiliza um fator de reserva, que reserva a média história de utilização do sistema, acrescido de uma capacidade adicional, reduzindo o número de violações ocorridas no sistema. Por fim, destacamos que não considerar recursos de borda no conjunto de análise pode incorrer em superestimação dos recursos computacionais, tornando a solução proposta inviável em alguns cenários.

O fato de que a escala de consumo de processamento de dados na borda é variável de acordo com cada aplicação, e portanto, inesperada, servindo como evidência clara de que projetar e implantar uma rede, considerando apenas recursos de transmissão de dados como único orientador no processo de decisão, provavelmente resultará em problemas de desempenho e baixa eficiência.

CAPÍTULO 6

Considerações Finais e Trabalhos Futuros

As redes de quinta geração (5G) são um marco na evolução da transmissão e processamento de informações. Uma das principais inovações dessa tecnologia é o conceito de NS, que possibilita a divisão e isolamento de recursos de comunicação e computação para cada serviço, na forma de *slices*. Novos casos de uso, como carros autônomos, *eHealth*, sistemas de gerenciamento de energia e provedores de serviços *Over-The-Top*, têm impulsionado o crescimento do NS, devido às suas demandas críticas e únicas de processamento e transmissão de dados.

Nesta tese, propusemos e analisamos três algoritmos de aprendizado por reforço, baseados no arcabouço de soluções do tipo MaB. Os algoritmos MONETS-OBD, MONETS-OBS e CAONS compõem uma solução de controle de admissão de *slices* instantânea na rede da operadora, que leva em conta os recursos de comunicação e computação de borda. No problema abordado, o objetivo é decidir, com base em métricas de recompensa e penalização, quais *slices* devem ser admitidos na infraestrutura da operadora, buscando maximizar os ganhos de multiplexação da rede. Nossa solução se concentra em problemas da classe MaB, com restrição de lock-up forçado. Projetamos soluções de baixa complexidade, a custo de uma solução sub-ótima, que alcança uma acomodação de *slices* na rede da operadora de maneira eficiente.

Nossos resultados indicam que considerar os recursos computacionais no conjunto de decisão é igualmente importante à consideração dos recursos de comunicação no processo de admissão de inquilinos na rede. Caso contrário, os recursos computacionais, escassos na borda, poderiam ser superestimados, levando a interpretações equivocadas de admissão de inquilinos e alocação dos recursos. Os algoritmos propostos, MONETS-OBD, MONETS-OBS e CAONS, alcançam ganhos significativos de multiplexação, superando soluções não orientadas (ϵ-greedy), soluções que abordam apenas recursos de comunicação (ONETS) e também uma solução orientada (eUCB), para os cenários analisados. A heurística MONETS-OBD alcança taxas de aceitação, recompensa e utilização próximas de 99%, em alguns casos, do cenário ideal, ao custo de ocorrências de violações da capacidade do sistema próximas de 10% no pior caso. MONETS-OBS por outro lado, utiliza um fator de reserva, que reserva a média história de utilização do sistema,

acrescido de uma capacidade adicional, reduzindo o número de violações ocorridas no sistema. Por fim, CAONS utiliza técnicas de penalização, para guiar o algoritmo acerca de de inquilinos que podem resultar em violações pós aceite de *slices*. CAONS alcança taxas de aceitação, utilização e recompensa próximas ao modelo de referência, a um custo de ocorrências de violações em menos de 1% dos casos. Por fim, demonstramos que a complexidade computacional dos algoritmos propostos é semelhante à de uma solução não orientada (ϵ-greedy), sendo adequados para problemas de decisão instantânea.

6.1 Produções Acadêmicas e Bibliográficas

Esta tese gerou dois resultados preliminares, que foram publicados ou aceitos para publicação em congressos nacionais:

- *"Controle de Admissão para Network Slicing Ciente de Recursos de Rede e de Processamento"*, aceito e apresentado no SBRT 2022 (Simpósio Brasileiro de Telecomunicações e Processamento de Sinais 2022).
- *"CAONS: Controle de Admissão On-line para RAN Slicing Baseado na Convergência de Comunicação e Computação"*, aceito no SBRC 2023 (Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos 2023).

6.2 Trabalhos Futuros

Por fim, este trabalho deixa perspectivas de continuidade para a avaliação de desempenho realizada nesta tese. Em particular, como sugestão para trabalhos futuros, mencionamos as seguintes possibilidades:

- Avaliar novas técnicas de *Reinforcement Learning*, e.g., *Deep Q-Learning* para resolver o problema de controle de admissão de *slices*. Essa implementação poderá reforçar os resultados apresentados ou apresentar novas óticas não identificadas com o uso das técnicas utilizadas;
- Estender o problema de controle de admissão de *slices* para um modelo fim-a-fim, que considere os segmentos da RAN, TN e CN.
- Avaliar o desempenho do modelo proposto para controle de admissão de *slices* com diferentes categorias de serviço.

Além disso, destaca-se que conhecimentos prévios acerca do modo de funcionamento das ferramentas MATLAB e CPLEX, representam uma facilitação no processo de geração de novos resultados, assim como conhecimentos de técnicas de modelagens de programação. Dessa maneira, novos pesquisadores poderão superar mais facilmente as

principais barreiras iniciais que foram encontradas neste trabalho, quebrando rapidamente dificuldades da construção de conhecimentos acerca desse conjunto de ferramentas e das técnicas de programação matemática utilizadas.

Referências Bibliográficas

[3GPP 2019]3GPP. *3GPP Release 15*. 2019. [Online]. Disponível em: https://www.3gpp.org/specifications-technologies/releases/release-15. Acessado em: 04 de fevereiro de 2023.

[3GPP 2020]3GPP. *3GPP Release 16*. 2020. [Online]. Disponível em: https://www.3gpp.org/specifications-technologies/releases/release-16. Acessado em: 04 de fevereiro de 2023.

[3GPP 2021]3GPP. *3GPP Release 17*. 2021. [Online]. Disponível em: https://www.3gpp.org/specifications-technologies/releases/release-17. Acessado em: 04 de fevereiro de 2023.

[3GPP 2009]3GPP, A. A. N. 3gpp long term evolution: System overview, product development, and test challenges. *Literature Number*, 2009.

[Afolabi et al. 2020]AFOLABI, I. et al. Dynamic resource provisioning of a scalable e2e network slicing orchestration system. *IEEE Transactions on Mobile Computing*, v. 19, n. 11, p. 2594–2608, 2020.

[Afolabi et al. 2018]AFOLABI, I. et al. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, IEEE, v. 20, n. 3, p. 2429–2453, 2018.

[Ahmad et al. 2018]AHMAD, I. et al. Overview of 5g security challenges and solutions. *IEEE Communications Standards Magazine*, IEEE, v. 2, n. 1, p. 36–43, 2018.

[Albreem, Juntti e Shahabuddin 2019]ALBREEM, M. A.; JUNTTI, M.; SHAHABUDDIN, S. Massive mimo detection techniques: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 4, p. 3109–3132, 2019.

[Alliance 2016]ALLIANCE, N. Description of network slicing concept. *NGMN 5G P*, v. 1, n. 1, p. 1–11, 2016.

[Alliance 2016]ALLIANCE, N. Perspectives on vertical industries and implications for 5g. *White Paper, Jun*, 2016.

[Alliance 2018]ALLIANCE, N. Ngmn 5g pre-commercial networks trials. 2018.

[Bairagi et al. 2020]BAIRAGI, A. K. et al. Coexistence mechanism between embb and urllc in 5g wireless networks. *IEEE Transactions on Communications*, IEEE, v. 69, n. 3, p. 1736–1749, 2020.

[Bakri, Brik e Ksentini 2021]BAKRI, S.; BRIK, B.; KSENTINI, A. On using reinforcement learning for network slice admission control in 5g: Offline vs. online. *International Journal of Communication Systems*, Wiley Online Library, v. 34, n. 7, p. e4757, 2021.

[Black 2005]BLACK, P. E. Greedy algorithm. *Dictionary of Algorithms and Data Structures*, US National Institute of Standards and Technology, v. 2, p. 62, 2005.

[Caballero et al. 2018]CABALLERO, P. et al. Network slicing for guaranteed rate services: Admission control and resource allocation games. *IEEE Transactions on Wireless Communications*, v. 17, n. 10, p. 6419–6432, 2018.

[Cao, Kodialam e Lakshman 2014]CAO, Z.; KODIALAM, M.; LAKSHMAN, T. V. Traffic steering in software defined networks: Planning and online routing. In: *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2014. (DCC '14), p. 65–70. ISBN 9781450329927. Disponível em: <https://doi.org/10.1145/2627566.2627574>.

[Chih-Lin, Kuklinskí e Chen 2020]CHIH-LIN, I.; KUKLINSKÍ, S.; CHEN, T. A perspective of o-ran integration with mec, son, and network slicing in the 5g era. *IEEE Network*, IEEE, v. 34, n. 6, p. 3–4, 2020.

[Chochliouros et al. 2020]CHOCHLIOUROS, I. P. et al. Dynamic network slicing: Challenges and opportunities. In: SPRINGER. *Artificial Intelligence Applications and Innovations. AIAI 2020 IFIP WG 12.5 International Workshops: MHDW 2020 and 5G-PINE 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings 16*. [S.l.], 2020. p. 47–60.

[CISCO 2022]CISCO. Cisco annual internet report (2022-2025). . 2022.

[Cominardi et al. 2020]COMINARDI, L. et al. Mec support for network slicing: Status and limitations from a standardization viewpoint. *IEEE Communications Standards Magazine*, v. 4, n. 2, p. 22–30, 2020.

[CPLEX 2018]CPLEX, M. U. Ibm ilog cplex optimization studio. *Version*, v. 12, p. 1987–2018, 2018.

[Cruz, Achir e Viana 2022]CRUZ, P.; ACHIR, N.; VIANA, A. C. On the edge of the deployment: A survey on multi-access edge computing. Association for Computing Machinery, New York, NY, USA, v. 55, n. 5, 2022. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3529758>.

[Demmer et al. 2018]DEMMER, D. et al. Analytical study of 5g nr embb co-existence. In: IEEE. *2018 25th International Conference on Telecommunications (ICT)*. [S.l.], 2018. p. 186–190.

[Devlic et al. 2017]DEVLIC, A. et al. Nesmo: Network slicing management and orchestration framework. In: *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*. [S.l.: s.n.], 2017. p. 1202–1208.

[ETSI 2019]ETSI. Multi-access edge computing (mec); radio network information api. In: ETSI. *Group Spec. (GS) 012 v2.1.1*. [S.l.], 2019.

[Feng et al. 2020]FENG, J. et al. Dynamic network slicing and resource allocation in mobile edge computing systems. *IEEE Transactions on Vehicular Technology*, IEEE, v. 69, n. 7, p. 7863–7878, 2020.

[Foukas et al. 2017]FOUKAS, X. et al. Network slicing in 5g: Survey and challenges. *IEEE Communications Magazine*, IEEE, v. 55, n. 5, p. 94–100, 2017.

[Garivier e Cappé 2011]GARIVIER, A.; CAPPé, O. The kl-ucb algorithm for bounded stochastic bandits and beyond. In: KAKADE, S. M.; LUXBURG, U. von (Ed.). *Proceedings of the 24th Annual Conference on Learning Theory*. Budapest, Hungary: PMLR, 2011. (Proceedings of Machine Learning Research, v. 19), p. 359–376. Disponível em: <http://proceedings.mlr.press/v19/garivier11a.html>.

[Guan et al. 2020]GUAN, Z. et al. Robust stochastic bandit algorithms under probabilistic unbounded adversarial attack. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2020. v. 34, n. 04, p. 4036–4043.

[Gunther 2007]GUNTHER, N. J. *Understanding load averages and stretch factors*. [S.l.], 2007.

[Han et al. 2015]HAN, B. et al. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, v. 53, n. 2, p. 90–97, 2015.

[Han e Schotten 2020]HAN, B.; SCHOTTEN, H. D. Machine learning for network slicing resource management: A comprehensive survey. *arXiv preprint arXiv:2001.07974*, 2020.

[Hattachi e Erfanian 2015]HATTACHI, R. E.; ERFANIAN, J. Ngmn 5g white paper ngmn alliance. *NGNM*, 2015.

[He e Song 2015]HE, J.; SONG, W. Appran: Application-oriented radio access network sharing in mobile networks. In: *2015 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2015. p. 3788–3794.

[Henry, Alsohaily e Sousa 2020]HENRY, S.; ALSOHAILY, A.; SOUSA, E. S. 5g is real: Evaluating the compliance of the 3gpp 5g new radio system with the itu imt-2020 requirements. *IEEE Access*, IEEE, v. 8, p. 42828–42840, 2020.

[Husni e Bramantyo 2018]HUSNI, E.; BRAMANTYO, A. Design and implementation of mpls sdn controller application based on opendaylight. In: *2018 International Symposium on Networks, Computers and Communications (ISNCC)*. [S.l.: s.n.], 2018. p. 1–5.

[Hwang e Park 2017]HWANG, S.; PARK, S. On the effects of resource usage ratio on data rate in lte systems. In: IEEE. *2017 19th International Conference on Advanced Communication Technology (ICACT)*. [S.l.], 2017. p. 78–80.

[Hwang e Park 2017]HWANG, S.; PARK, S. On the effects of resource usage ratio on data rate in lte systems. In: *2017 19th International Conference on Advanced Communication Technology (ICACT)*. [S.l.: s.n.], 2017. p. 78–80.

[Jiang, Condoluci e Mahmoodi 2016]JIANG, M.; CONDOLUCI, M.; MAHMOODI, T. Network slicing management & prioritization in 5g mobile systems. p. 1–6, 2016.

[Katsalis, Nikaein e Huang 2018]KATSALIS, K.; NIKAEIN, N.; HUANG, A. Jox: An event-driven orchestrator for 5g network slicing. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2018. p. 1–9.

[Kekki et al. 2018]KEKKI, S. et al. Mec in 5g networks. *ETSI white paper*, v. 28, p. 1–28, 2018.

[Komiyama, Sato e Nakagawa 2013]KOMIYAMA, J.; SATO, I.; NAKAGAWA, H. Multi-armed bandit problem with lock-up periods. In: PMLR. *Asian Conference on Machine Learning*. [S.l.], 2013. p. 100–115.

[Kukliński e Tomaszewski 2019]KUKLIńSKI, S.; TOMASZEWSKI, L. Key performance indicators for 5g network slicing. In: *2019 IEEE Conference on Network Softwarization (NetSoft)*. [S.l.: s.n.], 2019. p. 464–471.

[Li et al. 2017]LI, X. et al. Network slicing for 5g: Challenges and opportunities. *IEEE Internet Computing*, IEEE, v. 21, n. 5, p. 20–27, 2017.

[Mahajan e Teneketzis 2008]MAHAJAN, A.; TENEKETZIS, D. Multi-armed bandit problems. In: *Foundations and applications of sensor management*. [S.l.]: Springer, 2008. p. 121–151.

[Malandrino et al. 2020]MALANDRINO, F. et al. From megabits to cpu ticks: Enriching a demand trace in the age of mec. *IEEE Transactions on Big Data*, v. 6, n. 1, p. 43–50, 2020.

[MEC 2019]MEC, I. E. Multi-access edge computing (mec): Framework and reference architecture. 2019.

[Medeiros et al. 2017]MEDEIROS, A. M. M. et al. The fifth generation of mobile communication and its applications on the internet of things (iot). In: IEEE. *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. [S.l.], 2017. p. 1–7.

[Mekikis et al. 2019]MEKIKIS, P.-V. et al. Nfv-enabled experimental platform for 5g tactile internet support in industrial environments. *IEEE Transactions on Industrial Informatics*, IEEE, v. 16, n. 3, p. 1895–1903, 2019.

[Naqvi et al. 2020]NAQVI, S. I. et al. Integrated lte and millimeter-wave 5g mimo antenna system for 4g/5g wireless terminals. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 14, p. 3926, 2020.

[Ontanón 2013]ONTANÓN, S. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. [S.l.: s.n.], 2013. v. 9, n. 1.

[Popovski et al. 2018]POPOVSKI, P. et al. 5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view. *Ieee Access*, IEEE, v. 6, p. 55765–55779, 2018.

[Ricart-Sanchez et al. 2019]RICART-SANCHEZ, R. et al. P4-netfpga-based network slicing solution for 5g mec architectures. In: IEEE. *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. [S.l.], 2019. p. 1–2.

[Rifai e Supriyanto 2017]RIFAI, B.; SUPRIYANTO, E. Management system failover dengan routing dinamis open shortest path first dan border gateway protocol. *JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer)*, v. 3, n. 1, p. 39–46, Aug. 2017. Disponível em: <http://ejournal.nusamandiri.ac.id/index.php/jitk/article/view/335>.

[Samdanis, Costa-Perez e Sciancalepore 2016]SAMDANIS, K.; COSTA-PEREZ, X.; SCIANCALEPORE, V. From network sharing to multi-tenancy: The 5g network slice broker. *IEEE Communications Magazine*, v. 54, n. 7, p. 32–39, 2016.

[Sciancalepore et al. 2022]SCIANCALEPORE, V. et al. Onets: Online network slice broker from theory to practice. *IEEE Transactions on Wireless Communications*, v. 21, n. 1, p. 121–134, 2022.

[Sim e Reid 1999]SIM, J.; REID, N. Statistical inference by confidence intervals: issues of interpretation and utilization. *Physical Therapy*, Oxford University Press, v. 79, n. 2, p. 186–195, 1999.

[Singh e Kumbhani 2020]SINGH, R. S. K.; KUMBHANI, B. The evolution of radio access network towards open-ran: Challenges and opportunities. In: IEEE. *2020 IEEE wireless communications and networking conference workshops (WCNCW)*. [S.l.], 2020. p. 1–6.

[Taleb et al. 2017]TALEB, T. et al. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, IEEE, v. 19, n. 3, p. 1657–1681, 2017.

[Tran-Thanh et al. 2012]TRAN-THANH, L. et al. Knapsack based optimal policies for budget–limited multi–armed bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 26, n. 1, Jul. 2012. Disponível em: <https://ojs.aaai.org/index.php/AAAI/article/view/8279>.

[Wang, Rosa e Pedersen 2010]WANG, H.; ROSA, C.; PEDERSEN, K. Performance of uplink carrier aggregation in lte-advanced systems. In: *2010 IEEE 72nd Vehicular Technology Conference - Fall*. [S.l.: s.n.], 2010. p. 1–5.

[Weisbecker 2013]WEISBECKER, F. Status of linux dynticks. In: *9th annual workshop on Operating Systems Platforms for Embedded Real-Time applications-OSPERT13*. [S.l.: s.n.], 2013.

[Wijethilaka e Liyanage 2021]WIJETHILAKA, S.; LIYANAGE, M. Survey on network slicing for internet of things realization in 5g networks. *IEEE Communications Surveys & Tutorials*, IEEE, v. 23, n. 2, p. 957–994, 2021.

[Wu et al. 2015]WU, J. et al. Cloud radio access network (c-ran): a primer. *IEEE network*, IEEE, v. 29, n. 1, p. 35–41, 2015.

[Zaidi et al. 2018]ZAIDI, Z. et al. Will sdn be part of 5g? *IEEE Communications Surveys & Tutorials*, IEEE, v. 20, n. 4, p. 3220–3258, 2018.

[Zhang et al. 2017]ZHANG, H. et al. Network slicing based 5g and future mobile networks: mobility, resource management, and challenges. *IEEE communications magazine*, IEEE, v. 55, n. 8, p. 138–145, 2017.

[Zhang 2019]ZHANG, S. An overview of network slicing for 5g. *IEEE Wireless Communications*, IEEE, v. 26, n. 3, p. 111–117, 2019.