# CAONS: Controle de Admissão On-line para RAN *Slicing* Baseado na Convergência de Comunicação e Computação

**Henrique V. Lima**[1], **Sand L. Correa**[1], **Kleber V. Cardoso**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Goiânia – GO – Brazil

henrivalle@discente.ufg.br, sand,kleber@ufg.br

***Abstract.*** *The deployment of 5G mobile networks has leveraged Network Slicing (NS), a disruptive technology that can provide dedicated resources in mobile systems, helping to monetize the physical and logical infrastructure. However, NS creates new challenges, such as (1) dynamic and flexible management of Radio Access Network (RAN) resources, (2) smooth integration of services in Multi Access Edge Computing (MEC), and (3) admission of new tenants to the network. In this work, we propose an admission control algorithm aware of RAN resources, which uses the overbooking technique to increase infrastructure utilization, penalizing the operator in case of violations. We evaluate the model and compare it with known solutions, using data from different applications.*

***Resumo.*** *A implantação das redes móveis 5G têm alavancado o Network Slicing (NS), uma tecnologia disruptiva, que pode fornecer recursos dedicados nos sistemas móveis, auxiliando na rentabilização da infraestrutura física e lógica. Contudo, NS cria novos desafios, como: o (1) gerenciamento dinâmico e flexível de recursos da Radio Access Network (RAN), (2) a integração harmoniosa dos serviços em Multi Access Edge Computing (MEC), e (3) admissão de novos inquilinos na rede. Neste trabalho, propomos um algoritmo de controle de admissão para NS ciente de recursos da RAN, que utiliza a técnica de overbooking para aumentar a utilização da infraestrutura, penalizando a operadora em caso de violações. Avaliamos o modelo proposto e comparamos com soluções conhecidas, utilizando dados de diferentes aplicações.*

## 1. Introdução

As redes móveis de quinta geração (5G) trouxeram consigo aplicações como *Enhanced Mobile Broadband* (eMBB), *Ultra Reliable and Low Latency Communication* (URLLC) e *Massive Machine Type Communication* (mMTC), que criam novas oportunidades de monetização na *Radio access network* (RAN) do *Mobile Network Operator* (MNOs), ou simplesmente operadora. No entanto, essas aplicações apresentam novos desafios, em termos de latência, confiabilidade, disponibilidade e segurança, exigindo uma abordagem mais flexível no gerenciamento dos recursos [Li et al. 2018b].

O paradigma de *Network Slicing* (NS) fornece uma base de solução para esse problema. Com o uso de NS o operador pode isolar parte dos recursos da RAN (*Radio Access Network*) na forma de *slices*, alugando-os para um conjunto de inquilinos independentes. Em cada solicitação de *slice*, o operador deve identificar os recursos que podem ser disponibilizados ao inquilino para fornecimento de serviços aos seus clientes. Além disso, o operador deve ter a capacidade de adaptar as solicitações de *slices* de acordo com os requisitos de seus inquilinos em tempo real, evitando despesas adicionais [Zhang 2019].

Cada nova aplicação em 5G apresenta um conjunto de requisitos específicos de QoS (*Quality of Service*). O NS atua dividindo os recursos da rede em partes menores, chamadas *slices*. Um *slice* é composto por um conjunto de VNFs (*Virtual Network Functions*), criadas sob demanda, seguindo os princípios de isolamento, automação, customização, elasticidade, programabilidade e de entrega do serviço *end-to-end*. Dessa maneira, o NS pode modularizar a rede e possibilitar a construção, sobre uma mesma infraestrutura, de diversos *slices* customizados de maneira independente, que atendem as necessidades únicas e particulares de cada aplicação [Rifai and Supriyanto 2017].

Neste artigo, analisamos a arquitetura 5G e propomos uma solução on-line para controle de admissão de inquilinos. Tratamos o problema formulado na Seção 3 utilizando técnicas de *Reinforcement Learning* (RL), mais especificamente, como um problema *Multi-Armed Bandit* (MaB) com restrições de *lock-up* forçado e penalidade na ocorrência de violações. A solução CAONS (Controle de Admissão On-line para RAN *Slicing*) proposta utiliza um conjunto de amostras históricas para equilibrar a relação *exploration* x *exploitation*, e a partir disso, seleciona o conjunto de inquilinos para atendimento. O algoritmo fornece a solução em tempo polinomial, sendo este adequado para problemas on-line. Por fim, comparamos CAONS com heurística disponíveis na literatura. Resultados demonstram que o algoritmo alcança recompensas próximas da solução ótima, respeitando a restrição de capacidade dos recursos de comunicação e computação.

Este trabalho está organizado da seguinte forma. Na Seção 2, são apresentados os conceitos básicos e trabalhos relacionados. Na Seção 3, o modelo de controle de admissão é reformulado. A Seção 4 apresenta o algoritmo CAONS, avaliado numericamente na Seção 5. Por fim, na Seção 6 encontram-se as considerações finais.

## 2. Fundamentação Teórica e Trabalhos Relacionados

O conceito de NS em uma rede está diretamente relacionado à capacidade de realizar diferenciação entre os tipos de serviços. Na RAN, esta informação pode ser obtida pelo mapeamento das métricas de *Quality of Service Class Identifiers* (QCI), permitindo a criação de serviços fim a fim, com diferentes tráfegos e políticas, satisfazendo ao mesmo tempo, uma ampla gama de requisitos heterogêneos. Além disso, espera-se que a utilização de *slices* na RAN abra novos modelos de negócios, como por exemplo, uma OTT (*over-the-top*) locando e utilizando recursos da RAN na forma de vários *slices* lógicos. Dessa maneira, as operadoras poderão expandir os negócios para o crescente mercado de rede flexível, de alto desempenho e sob demanda das redes 5G [Zhang 2019].

Mecanismos de controle de admissão de *slices* já foram abordados na literatura. Em [Elayoubi et al. 2019], o controle de admissão de *slices* é introduzido como um problema NP-difícil. Em [Li et al. 2018a], os autores analisam matematicamente o problema. Em [Bega et al. 2019] é proposta uma solução *offline* baseada em precificação. Ao decidir sobre a admissão de um *slices*, a operadora da rede deve levar em consideração tanto fatores determinísticos, como os requisitos dos *slices* e a capacidade total do sistema, quanto fatores estocásticos, como as requisições futuras e a utilização efetiva de recursos.

Neste contexto, um processo de decisão on-line é a solução mais apropriada para tratar o problema. Dessa maneira, os autores em [Sciancalepore et al. 2021] propõem o ONETS, um mecanismo on-line de controle de admissão de *slices*, que utiliza uma estratégia de *overbooking*. Contudo, ao decidir acerca do o conjunto de *slices* que devem

ser admitidos no sistema, o ONETS considera apenas recursos de rádio. Os autores em [Cominardi et al. 2020] evidenciam quem em 5G e pós-5G, os recursos de computação são fundamentais para as diversas VNFs que compõem os serviços. Desse modo, os autores em [Lima et al. 2022] propõem uma extensão de ONETS, chamada de MONETS, que realiza controle de admissão on-line *slices*, considerando recursos de comunicação e computação.

Nas estratégias apresentadas para controle de admissão de inquilinos, podemos encontrar autores que utilizam técnicas de RL para proposição das soluções. Contudo, nas propostas conhecidas, o aprendizado dos algoritmos é guiado pela recompensa recebida. Em RL, estratégias de busca baseadas em arrependimento ou penalização ajudam a evitar de escolhas sub ótimas, podendo acelerar o processo de treinamento. Desse modo, para preencher essa lacuna, propomos neste artigo o algoritmo CAONS, para controle de admissão de *slices* que leva em consideração os recursos de comunicação e computação de uma RAN. Utilizamos estratégias de *overbooking* e aplicamos penalidades em caso de violação da capacidade máxima do sistema.

## 3. Modelo de Sistema e Formulação do Problema

Nesta seção, formulamos um problema de controle de admissão de inquilinos em um provedor de serviços de telecomunicações (operadora da rede). A operadora, proprietária da infraestrutura da rede, disponibiliza os recursos de comunicação e computação para clientes externos, como por exemplo, clientes da indústria vertical. Consideramos um cenário em que o segmento da RAN contém recursos de rádio, providos por *base stations* (BSs), e de processamento, providos por servidores MEC.

Com o objetivo de maximizar a receita da operadora, assumimos que a infraestrutura possa alugada, por espaços temporais pré definidos. Assumimos que as BSs ofereçam suporte para *network slice*. Dessa maneira, o operador terá de decidir entre admitir ou não inquilinos no sistema, considerando a capacidade total disponível, a recompensa pelo aceite e os padrões históricos de utilização do sistema dos inquilinos. O objetivo principal é decidir qual/quais inquilinos devem ser aceitos na rede de forma a maximizar a recompensa recebida pela operadora.

Consideramos $\mathcal{B}$, $\mathcal{M}$, $\mathcal{I}$, conjuntos de índices associados, respectivamente às BSs, servidores MEC e aos potenciais inquilinos. Cada BS $b \in \{1, 2, \cdots, |\mathcal{B}|\}$ possui uma capacidade total $C^r$ de comunicação, dada em números de PRBs. De maneira semelhante, designamos por $C^c$ a capacidade total de processamento em cada servidor MEC $m \in \{1, 2, \cdots, |\mathcal{M}|\}$, expressa em *CPU ticks*. Seja $S = \{1, 2, \cdots, |S|\}$ o conjunto de modelos (*templates*) de *slices* disponibilizados pelo operador. Cada inquilino $i \in \{1, 2, \cdots, |I|\}$ pode requisitar um *slice* $s \in S$ que melhor represente as características de seu serviço. Cada *template* de *slice* $s = \{R^{(s)}; Q^{(s)}; L^{(s)}\}$ compreende uma quantidade de recursos de radio (dado em número de PRBs), representada por $R^{(s)}$, uma quantidade de recursos computacionais (expressa em *CPU ticks*), denotada por $Q^{(s)}$, e o tempo de duração do *slice* (expresso em segundos), representado por $L^{(s)}$.

Seja $\mathcal{T} = \{1, 2, \cdots, |\mathcal{T}|\}$ o espaço temporal em que o mecanismo de controle de admissão toma decisões sobre quais requisições em curso aceitar. Cada inquilino $i \in \mathcal{I}$ pode solicitar um *slice* $s \in \mathcal{S}$ no tempo $t \in \mathcal{T}$. Denotamos tal requisição por $r_{i,t}^{(s)}$. Modelamos a distribuição das requisições no tempo $i \in \mathcal{I}$ como uma variável aleatória, de

forma que o tempo entre as chegadas, representado por $\Delta_t$, é distribuído exponencialmente com taxa $\phi_i$. Obtemos $\phi_i$ a partir de uma distribuição de Pareto, com média $\rho$ e desvio padrão $\zeta$.

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

A política de seleção adotada decide entre conceder ou não ao inquilino selecionado a quantidade de recursos de rede e computação, necessários ao tipo de *slice* acordado. É possível que um inquilino seja selecionado, mas não tenha manifestado interesse na utilização, nesse caso, não será retornada recompensa pela ação. Definimos as funções de custo $\lambda_{i,t}$ e $\xi_{i,t}$ respectivamente, como o número de PRBs e o número de *CPU ticks* efetivamente utilizados na rodada $t$ na BS $b$ pelo inquilino $i$, de forma que:

$$\lambda_{i,b,t} \leq R^{(s)}_{i,b,t} \leq C^r_b \qquad (1) \qquad\qquad \xi_{i,m,t} \leq Q^{(s)}_{i,m,t} \leq C^c_m \qquad (2)$$

Os orçamentos totais para admissão de *slices* $C^r_b$ e $C^c_m$, representam respectivamente, a capacidade total de transmissão de dados da BS $b$ e a capacidade total de processamento de dados no MEC $m$. Esse conjunto define a região de admissibilidade, já que não podem ser alocados mais *slices* do que as capacidades $C^r$ e $C^c$ suportam.

Ao aceitar uma requisição de criação de *slice* para um inquilino $i$, na BS $b$, no tempo $st$, a operadora recebe recompensas pela utilização dos recursos de comunicação e de computação. A Equação 3 expressa a recompensa recebida, em termos de recursos de comunicação. De maneira similar, a Equação 4 expressa a recompensa recebida, em termos de recursos de computação.

$$\eta_{i,b,t} = \alpha \frac{R^{(s)}_{i,b,t}}{C^r_b} + (1-\alpha)\frac{R^{(s)}_{i,b,t} - \lambda_{i,b,t}}{R^{(s)}_{i,b,t}}, \quad (3) \qquad \kappa_{i,m,t} = \sigma \frac{Q^{(s)}_{i,m,t}}{C^c_m} + (1-\sigma)\frac{Q^{(s)}_{i,m,t} - \xi_{i,m,t}}{Q^{(s)}_{i,m,t}}, \quad (4)$$

Particularmente, as recompensas $\eta_{i,b,t}$ e $\kappa_{i,m,t}$ levam em consideração a quantidade de recursos solicitada, bem como o ganho de multiplexação, ou seja, a proporção entre o que foi realmente usado e o que está sendo solicitado. Nas Equações 3 e 4, $\alpha$ e $\sigma$ são parâmetros que atribuem pesos à quantidade de recursos e ao ganho de multiplexação.

Neste modelo, utilizamos uma estratégia de *overbooking* que incentiva a recomercialização de forma deliberada de parte dos recursos já alocados anteriormente. No contexto de NS, a utilização de *overbooking* é motivada pela expectativa de que uma parte dos inquilinos não utilizarão totalmente a capacidade disponibilizada na rede.

Conforme demonstrado por [Cominardi et al. 2020], a demanda dos usuários não deve ser modelada com uma distribuição empírica exata, para análise da quantidade de recursos ociosos que podem ser revendidos na rede. A estocasticidade inerente ao comportamento dos inquilinos inviabiliza essa análise. Em adição, uma vez aceita a requisição,

o *slice* deve ser minimamente garantidos ao longo do período de reserva $L_i^{(s)}$. Para reduzir os efeitos causados pela estratégia de *overbooking*, utilizamos uma penalidade, que é multiplicada por uma constante $\varrho$, sempre que o algoritmo aceitar um inquilino $i$ e não for possível manter o atendimento. As penalidades $\vartheta_{i,b,t}$ e $\nu_{i,b,t}$ representam respectivamente as penalidades aplicadas nos recursos de computação e comunicação. A Tabela 1 apresenta as notações utilizadas no modelo.

$$\vartheta_{i,b,t} = \varrho \frac{R_{i,b,t}^{(s)} - \lambda_{i,b,t}}{R_{i,b,t}^{(s)}}, \qquad (5) \qquad\qquad \nu_{i,m,t} = \varrho \frac{Q_{i,m,t}^{(s)} - \xi_{i,m,t}}{Q_{i,m,t}^{(s)}}, \qquad (6)$$

**Tabela 1. Variáveis utilizadas em nosso modelo.**

| | |
|---|---|
| $\mathcal{B}$ | Conjunto de *base stations* |
| $\mathcal{M}$ | Conjunto de servidores MEC |
| $\mathcal{I}$ | Conjunto de inquilinos |
| $\mathcal{T}$ | Conjunto de estampas de tempo onde são feitas decisões |
| $C_b^r$ | Capacidade total disponível para transmitir dados na rede na BS $b$ |
| $C_m^c$ | Capacidade total disponível para processamento de dados no MEC $m$ |
| $\eta_{i,b,t}$ | Valor de recompensa recebida, em termos de recursos de rede, pela aceitação da requisição do inquilino $i$, na BS $b$ no tempo $t$ |
| $\kappa_{i,m,t}$ | Valor de recompensa recebida, em termos de recursos de computação, pela aceitação da requisição do inquilino $i$, no MEC $m$ no tempo $t$ |
| $\vartheta_{i,b,t}$ | Valor da penalidade sofrida, em termos de recursos de rede, pelo não cumprimento da requisição do inquilino $i$, na BS $b$ no tempo $t$ |
| $\nu_{i,m,t}$ | Valor da penalidade sofrida, em termos de recursos de computação, pelo não cumprimento da requisição do inquilino $i$, no MEC $m$ no tempo $t$ |
| $R_{i,b,t}^{(s)}$ | Quantidade de recursos de rede disponibilizada para o inquilino $i$, na BS $b$ no instante de tempo $t$ |
| $\lambda_{i,b,t}$ | Quantidade de recursos de rede efetivamente utilizada pelo inquilino $i$, na BS $b$ no instante de tempo $t$ |
| $Q_{i,m,t}^{(s)}$ | Quantidade de recursos de computação disponibilizada para o inquilino $i$, na BS $b$ no instante de tempo $t$ |
| $\xi_{i,m,t}$ | Quantidade de recursos de computação efetivamente utilizada pelo inquilino $i$, na BS $b$ no instante de tempo $t$ |
| $\alpha$ | Peso atribuído à quantidade de recursos de rede solicitada |
| $\sigma$ | Peso atribuído à quantidade de recursos de computação solicitada |

Para todo inquilino $i \in \mathcal{I}$ e para todo instante de tempo $t \in \mathcal{T}$, definimos as seguintes variáveis de decisão:

- $x_{i,b,j} \in \{0, 1\}$, a qual assume o valor 1, quando o a requisição de criação de *slice* do inquilino $i$ é aceita na BS $b$ em $t$, e o valor 0, caso contrário.
- $x'_{i,m,j} \in \{0, 1\}$, a qual assume o valor 1, quando o a requisição de criação de *slice* do inquilino $i$ é aceita no MEC $m$ em $t$, e o valor 0, caso contrário.

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

$$\textit{maximizar} \quad \sum_{b\in\mathcal{B}}\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{I}}((\eta_{i,b,t}+\kappa_{i,m,t})-(\vartheta_{i,b,t}+\nu_{i,m,t}))x_{i,b,t}x'_{i,m,t} \tag{7}$$

$$\textit{sujeito a:} \quad \sum_{i\in\mathcal{I}}\lambda_{i,b,t}x_{i,b,t}\leq C_b^r,\forall t\in\mathcal{T},\forall b\in\mathcal{B} \tag{8}$$

$$\sum_{i\in\mathcal{I}}\xi_{i,m,t}x'_{i,m,t}\leq C_m^c,\forall t\in\mathcal{T},\forall m\in\mathcal{M} \tag{9}$$

$$x_{i,b,t}\geq x_{i,b,t-1}\mathcal{F}(t-t_{i,b}^{start}\leq L_{i,b}^{(s)}) \tag{10}$$

$$x'_{i,m,t}\geq x'_{i,m,t-1}\mathcal{F}(t-t_{i,m}^{start}\leq L_{i,m}^{(s)}) \tag{11}$$

$$x_{i,b,t},\ x'_{i,m,t}\in\{0,1\},\forall i\in\mathcal{I},\forall t\in\mathcal{T} \tag{12}$$

A função objetivo (7) incentiva a aceitação de requisições de inquilinos, considerando recursos de rede e de processamento. A restrição em (8) garante que a quantidade de PRBs utilizada não exceda a capacidade de cada BS. Em (9) garante-se que a quantidade de *CPU ticks* não exceda a capacidade total disponível. Em (10) e (11) 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 $\mathcal{F}$ é uma função indicadora que fornece valor 1, se a condição entre colchetes for satisfeita, e o valor 0, caso contrário. Por fim, em (12) estabelecemos as variáveis binárias de decisão.

## 4. Solução Proposta

No problema exploratório *Multi-Armed Bandit* (MaB) clássico, um jogador tem diversas máquinas caça-níquel à disposição. Cada máquina retorna uma recompensa obtida através de uma distribuição de probabilidade desconhecida. 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.

O controle de admissão de *slices*, pode ser modelado como um problema de decisão do tipo MaB com algumas variações: (i) múltiplos jogadores (inquilinos) podem ser selecionados em cada rodada, (ii) o orçamento (capacidade) é limitado e (iii) existência de períodos de *lock-up*, devido às restrições impostas em (10) e (11) e (iv) inserção de penalidade diante do não cumprimento da restrição de capacidade dentro do período de reserva [Li et al. 2020] [Sciancalepore et al. 2021] [Chen et al. 2022]. No modelo, cada inquilino $i$ é modelado como uma máquina caça-níquel, que se escolhida em $t$ retorna uma recompensa. Desse modo, o objetivo passa a ser maximizar a probabilidade de escolher o melhor braço [Mahajan and Teneketzis 2008]. Como mais de um inquilino pode ter uma requisição pendente no tempo $t$, o algoritmo pode escolher múltiplos inquilinos ao mesmo tempo, desde que respeitando o orçamento disponível.

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 ideal [Mahajan and Teneketzis 2008]. O problema em (7)-(12) não é

polinomial em sua versão como problema de decisão: determinar se existe uma solução viável para a qual o valor da função objetivo (7), é maior que um valor constante arbitrário. Ou seja, não existem reduções viáveis para um problema MaB que busca maximizar a recompensa, em períodos de *lockup* e restrições de capacidade, que possam ser verificadas em tempo polinomial para grandes instâncias [Gao et al. 2019].

Estratégias clássicas como o *Upper Confidence Bound* (UCB) e *Greedy algorithm* são frequentemente utilizadas na literatura para resolver o problema MaB sem garantia da recompensa ideal [Black 2005] [Ontanón 2013] [Guan et al. 2020]. Em um viés semelhante, [Sciancalepore et al. 2021] propõe a utilização do algoritmo ONETS para controle de admissão baseado na utilização instantânea. De modo semelhante, [Lima et al. 2022] propõem a utilização de MONETS para controle de admissão baseado em médias históricas. Contudo, todas as soluções narradas anteriormente não inserem métricas de penalidade ou arrependimento diante de escolhas ruins de inquilinos no sistema. A utilização de penalização pode, em alguns casos, aumentar o nível de precisão do algoritmo, guiando o aprendizado de maneira mais eficiente [Asawa and Teneketzis 1996]. Dessa maneira, propomos neste artigo um algoritmo Controle de Admissão On-line para RAN *Slicing* (CAONS), apresentado na seção (4.1), que resolve o problema em (7)-(12) em tempo polinomial ao custo de uma solução sub-ótima próxima ao valor de referência.

### 4.1. Algoritmo CAONS

Em vez de realizar escolhas arbitrárias sobre qual máquina caça-níquel selecionar em cada rodada, estratégias inteligentes baseiam-se no equilíbrio entre exploração (*exploration*) e aproveitamento (*exploitation*) para tomar decisões. A medida que o tempo avança, o algoritmo reúne conhecimento acerca do ambiente, deixando de ser exploratório, e passando a aproveitar a solução com maior recompensa estimada.

Um modelo clássico para solução de problemas MaB foi proposto em [Garivier and Cappé 2011]. O modelo serve como referência para nossa solução. Durante o processo de execução, o algoritmo coleta a recompensa obtida em cada máquina caça-níquel, 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 o peso G-ALG ($\sqrt{\frac{2\ log\ t}{W_i}}$) que atenua a influência negativa das decisões aleatórias. O peso atua dando preferências à inquilinos selecionados mais vezes historicamente, e que podem retornar maiores recompensas na relação de alocação/utilização de recursos junto à operadora.

Em nosso problema, máquinas caça-níquel podem ser selecionadas, obedecendo as restrições de orçamento ($C^r$ e $C^c$). Se uma máquina for selecionada em uma rodada anterior, e o período de reserva ainda estiver em vigência, deve-se obrigatoriamente selecionar essa máquina para a próxima rodada. Uma versão do algoritmo CAONS para o problema de admissão de *slices*, considerando recurso de rede e de computação, com aplicação de penalidades em caso de violação de capacidade é descrita no Algoritmo 1.

As linhas de 1-4 do algoritmo inicializam as variáveis de recompensa e média história. Nessa etapa, inquilinos apresentam seus interesses de utilização de *slices*. A média histórica acumulada é calculada na linha 8, de acordo com G-ALG. A cada rodada, o algoritmo seleciona inicialmente os inquilinos que estão com período de reserva vigente, respeitando a restrição de *lock-up* (linha 10). Se a quantidade de recursos alocada for

menor que os orçamentos disponíveis, o algoritmo escolhe, dentre os demais inquilinos, aqueles que contém os maiores valores para $\hat{\theta}_i(t)$ respeitando a restrição de orçamento (linha 11). Após a seleção, caso os recursos disponíveis sejam suficientes para atender o inquilino, ele será inserido no conjunto de inquilinos atendidos e os orçamentos $C^r$ e $C^c$ atualizados (linhas 12 - 13). Nas linhas 14 - 15 tratamos os casos de inquilinos com período de reserva vigente em momentos de indisponibilidade de recursos no sistema. Por fim, atualizamos a recompensa e as médias histórias nas linhas 18 - 20. O algoritmo CAONS seleciona $K$ inquilinos, respeitando a restrição de orçamento, de forma que a média da distribuição empírica seja maximizada. Essa estratégia reduz a complexidade da solução em $\mathcal{O}(K)$, servindo como um indicador entre o nível de qualidade da solução e o tempo de execução total esperado.

**Algoritmo 1** CAONS - Algoritmo de seleção de inquilinos, com garantias de orçamento.

**Entradas:** $K$, $\mathcal{B}$, $\mathcal{T}$, $\mathcal{I}$, $C^r$, $C^c$.
**Inicialização:** $B = 0$, $Z = 0$, $L \leftarrow \emptyset$, $W_i = 0$, $\bar{\theta}_i(0) = 0$, $n = 0$
1: **for all** $i \in \mathcal{I}$ **do**
2: $v_i \leftarrow \eta_{i,b,0} + \kappa_{i,m,0}$
3: ATUALIZE $\bar{\theta}_i(0)$
4: $W_i = W_i + 1$
5: **for all** $t \in \mathcal{T}$ **do**
6: **for all** $b \in \mathcal{B}$ **do**
7: **for all** $i \in \mathcal{I}$ **do**
8: $\hat{\theta}_i(t) = \bar{\theta}_i(t) + \sqrt{\frac{2\ log\ t}{W_i}}$
9: **while** $n \leq K$ **do**
10: **if** $L(\mathrm{t}) \neq \emptyset$ **then** $\hat{i} \leftarrow L(t)$
11: **else** $\hat{i}$ : *arg max* $\hat{\theta}_i(t)$
12: **if** $B + \hat{\lambda}_{i,b} + 3\varphi_i \leq C^r$ **and** $Z + \hat{\xi}_{i,m} + 3\varphi_i \leq C^c$ **then**
13: $R \leftarrow R \cup \hat{i},\ B = B + \hat{\lambda}_i + 3\varphi_i,\ Z = Z + \hat{\xi}_i + +3\varphi_i,\ n = n + 1$
14: **if** $B + \hat{\lambda}_{i,b} + 3\varphi_i > C^r$ **and** $Z + \hat{\xi}_{i,m} + 3\varphi_i > C^c$ **and** $L(t) \neq \emptyset$ **then**
15: ATUALIZE $\vartheta_{i,b}(t)$ e $\nu_{i,m}(t)$
16: **for all** $b \in B$ **do**
17: **for all** $i \in R$ **do**
18: $v_i \leftarrow (\eta_{i,b,t} + \kappa_{i,m,t}) - (\vartheta_{i,b}(t)\ +\ \nu_{i,m}(t))$
19: ATUALIZE $\bar{\theta}_i$(t)
20: $W_i = W_i + 1$
21: ATUALIZE $L(t)$; $B = 0$; $Z = 0, n = 0$

## 5. Avaliação de Desempenho

Nesta seção, avaliamos os resultados obtidos com o modelo de controle de admissão proposto na Seção 3. Todos os cenários de dados do problema foram gerados utilizando linguagem de programação Python a partir dos parâmetros fornecidos na Tabela 2.

Foram gerados seis cenários de experimentação para as estratégias analisadas. Em todos os cenários, as requisições de *slices* são geradas a partir de uma distribuição exponencial, com $\rho$ - frequência e $\zeta$ - ocorrências fixos. Para gerar o parâmetro $\lambda_{i,t}$ são utilizadas três diferentes distribuições, simulando os diferentes comportamentos dos inquilinos no sistema. Gaussiana, com média $\mu_i$ e desvio padrão $\omega$, uniforme, com intervalos [0,1 e 1,0] de $R^{(s)}_{i,b,t}$ e log-normal, com média $\mu_i$ e desvio padrão $\omega$. Por fim, para criar diferentes cenários de tráfego, $\mu_i$ varia de 10% a 90% do parâmetro $R^{(s)}_{i,b,t}$, definido no *template* de *slice* associado ao inquilino $i$. A Tabela 2 resume os parâmetros utilizados na avaliação.

**Tabela 2. Parâmetros da simulação.**

| Parâmetro | Valor | Parâmetro | Valor | Parâmetro | Valor |
|---|---|---|---|---|---|
| $\|\mathcal{I}\|$ | 10 | $\mu_i$ | [0,1 - 0,9] | $\omega$ | 1 |
| $\|\mathcal{T}\|$ | 10000 | $\rho$ | 100 | $\varrho$ | 10 |
| $\|\mathcal{B}\|$ | 5 | $\zeta$ | 0,2 | $K$ | 10 |
| $C^r$ | 216 | $\alpha$ | 0,5 | $L$ | [100,500, |
| $C^c$ | 26.052 | $\sigma$ | 0,5 | | 1000] |

A cada rodada $t$, a política de seleção implementada, CAONS, analisa o melhor conjunto de inquilinos. O algoritmo MONETS, proposto em [Lima et al. 2022], que provisiona recursos pela quantidade instantânea utilizada no *template*, i.e., $\lambda_{i,b,t}$ e $\xi_{i,b,t}$ é implementado para efeitos de comparação. Utiliza-se também a formulação ótima do problema (Ótimo), descrita na Seção 2, resolvida com o uso da biblioteca *docplex* do Python. Por fim, considera-se o algoritmo ONETS proposto em [Sciancalepore et al. 2021]. Os resultados representam as médias das recompensas, do número de violações de SLA e dos percentuais de utilização e aceitação obtidas durante 30 execuções.

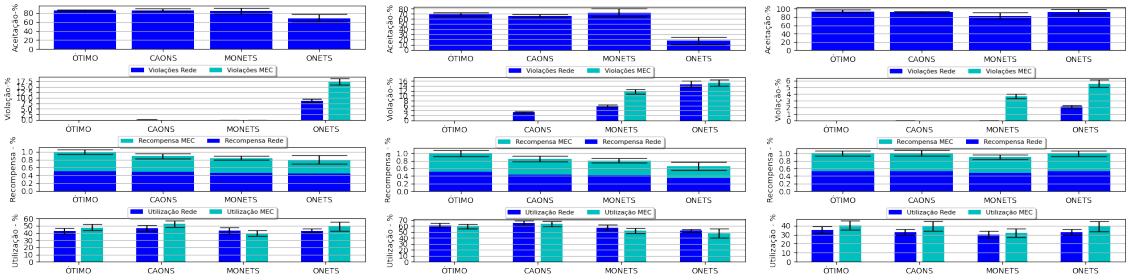
### 5.1. Configuração de Avaliação

No modelo, PRBs indicam a quantidade de recursos de rede disponível em uma BS. A taxa efetiva de transferência, em PRBs, será de aproximadamente 648 Mbps [Hwang and Park 2017]. Para quantificar os recursos de computação oferecidos por MEC, é utilizado o conceito de CPU *ticks* [Weisbecker 2013]. O modelo de carga de processamento segue Malandrino et al. [Malandrino et al. 2020], que utiliza os perfis de tráfego para caracterizar os serviços providos por inquilinos, i.e., (1) vídeo por *streaming* ($CPU_t = 0,25 * tr_{Mbit} + 6,76$), (2) jogos online ($CPU_t = 161,38 * tr_{Mbit} + 1675,03$) e (3) geolocalização e mapas ($CPU_t = 67,44 * tr_{Mbit} - 7,53$).

De posse dessas relações, é possível calcular a quantidade de CPU *ticks* necessária para cada tipo de tráfego. Consideramos dois *templates* de fatias para os mesmos serviços: demanda máxima e demanda mínima. As aplicações de vídeo variam entre [42 - 113] $CPU_t$, jogos entre [7.000 - 17.651] $CPU_t$ e mapas entre [2.757 - 8.287] $CPU_t$.

### 5.2. Avaliação da solução CAONS

Inicialmente, estudamos os resultados dos inquilinos que requisitam *slices* com serviços de jogos, mapas e vídeo. Na Figura 1 são ilustradas taxa de aceitação, violação, utilização e a recompensa média do sistema. A demanda dos usuários é gerada por uma distribuição gaussiana. Na Figura 1(a), a estratégia CAONS, que implementam penalização, alcança taxas de aceitação mais elevadas que MONETS e ONETS. CAONS e MONETS retornam em torno de 88% e 81% da recompensa recebida por Ótimo para cenários com demanda variada, respectivamente. Na Figura 1(b), ONETS retorna uma recompensas próximas de 80% da recompensa recebida por Ótimo, ao custo de uma violação de SLA próxima de 9% e 18% dos casos para os recursos de comunicação e computação, respectivamente. Por fim, na Figura 1(c), observa-se que nos gráficos de utilização de recursos, devido a variedade do tipo de inquilinos, a demanda manteve-se média, em torno de 40% - 60% para recursos de comunicação e computação.



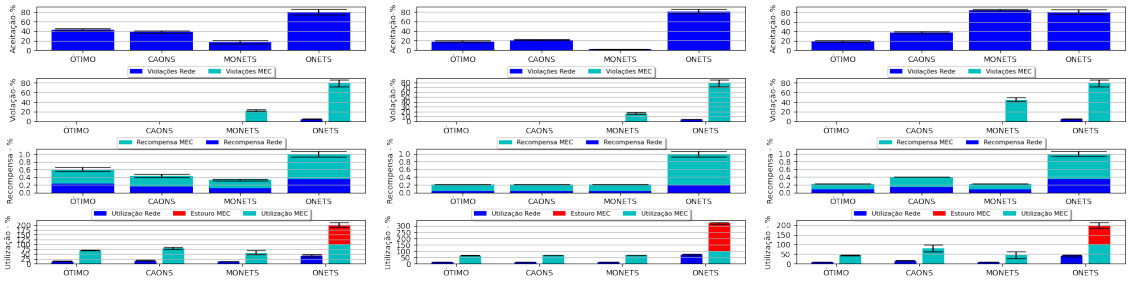

(a) Usuários Variados com demanda variada

(b) Usuários Variados com demanda máxima

(c) Usuários Variados com demanda mínima

**Figura 1. Resultados para requisições de criação de *slices* envolvendo aplicações de jogos, mapas e vídeo com demanda gaussiana.**

A Figura 2 considera que todos os inquilinos oferecem serviços de jogos on-line. Nas Figuras 2(a), 2(b) e 2(c), as recompensas médias obtidas com CAONS se aproximam do Ótimo, alcançando 99%, 94% e 80% respectivamente. Nos três casos analisados, Ótimo e CAONS não superestimam os recursos computacionais. MONETS e ONETS apresentam uma taxa de violação de SLA de 21% e 80% respectivamente. Nos três casos analisados, ONETS alcança as maiores recompensas e as maiores taxas de aceitação,s no entanto, como demonstrado nos gráficos de utilização, esses valores são alcançados porque o algoritmo não considera os recursos de comunicação, violando-os em até 320%. Apesar de não se tratarem de aplicações sensíveis, incorrer em atrasos no processamento de dados nessas aplicações poderia impactar os níveis de *Quality of Experience* (QoE) dos usuários finais, comprometendo o serviço ofertado. Por fim, os resultados se justificam pelas características das distribuição, por exemplo, a log-normal tem um comportamento de cauda longa, com forte pico de valores em curto espaço. Esse comportamento tende a deixar mais recursos livre para revenda, aumentando a utilização do sistema.




(a) Distribuição de demanda gaussiana

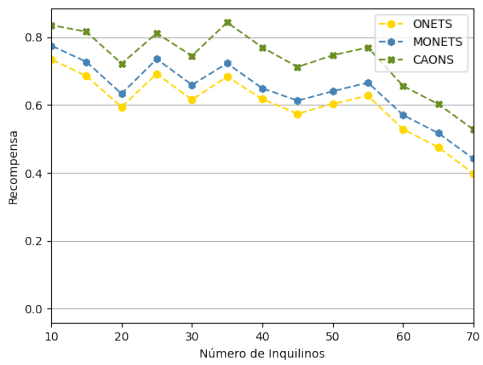
(b) Distribuição de demanda log-normal

(c) Distribuição de demanda uniforme

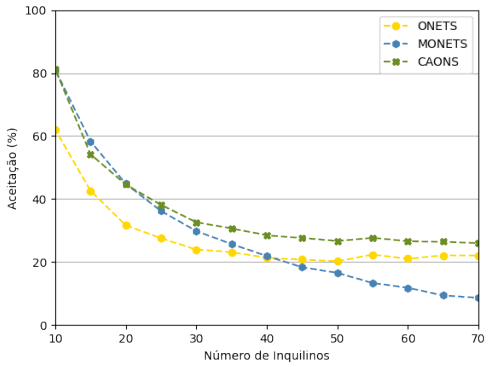
**Figura 2. Resultados para requisições de criação de *slices* envolvendo aplicações de jogos com demanda máxima.**

Para as próximas análises, retiramos o modelo ótimo do conjunto. Adotamos essa estratégia devido ao custo computacional. Conforme demonstrado na Seção 4, o problema é NP-difícil, exigindo tempo exponencial para ser resolvido. Trabalhamos na Figura 3 com conjuntos de dados maiores, incrementando o número de inquilinos. Dessa maneira, a execução do modelo ótimo, para os casos analisados, torna-se impraticável.

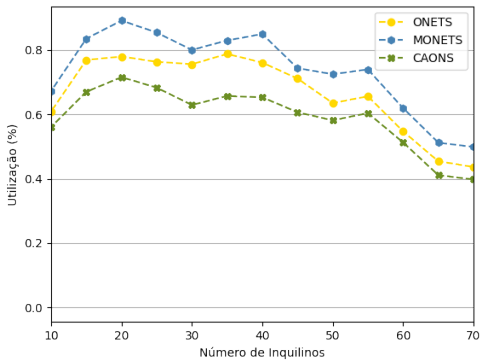
Na Figura 3, a variação das métricas de recompensa, aceitação, utilização e violação de SLA diante do incremento do número de inquilinos no sistema. Variamos $\mathcal{I} \in [10 - 70]$ inquilinos que realizam solicitações de criação de *slices* no sistema. As simulações foram realizadas para um conjunto variável de usuários com demanda máxima e distribuição de $\lambda_{i,b,t}$ e $\xi_{i,m,t}$ dada por uma gaussiana. Na Figura 3(a) apresentamos a variação da recompensa diante do incremento do número de usuários. A estratégia CAONS obteve as maiores recompensas, mesmo para cenários de maior densidade de inquilinos competindo por recursos. As Figuras 3(b) e 3(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 3(d) demonstra que o número de violações da estratégia CAONS mantém-se muito próxima a zero, mesmo para cenários de maior densidade de inquilinos.



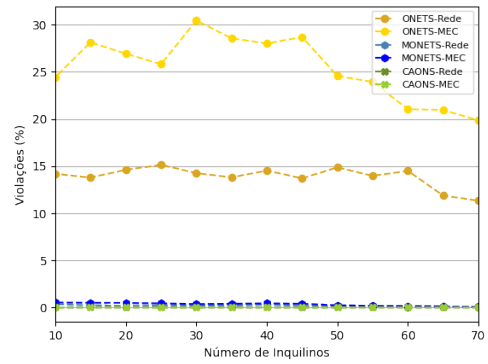

(a) Recompensa recebida




(b) Taxa de aceitação




(c) Taxa de utilização

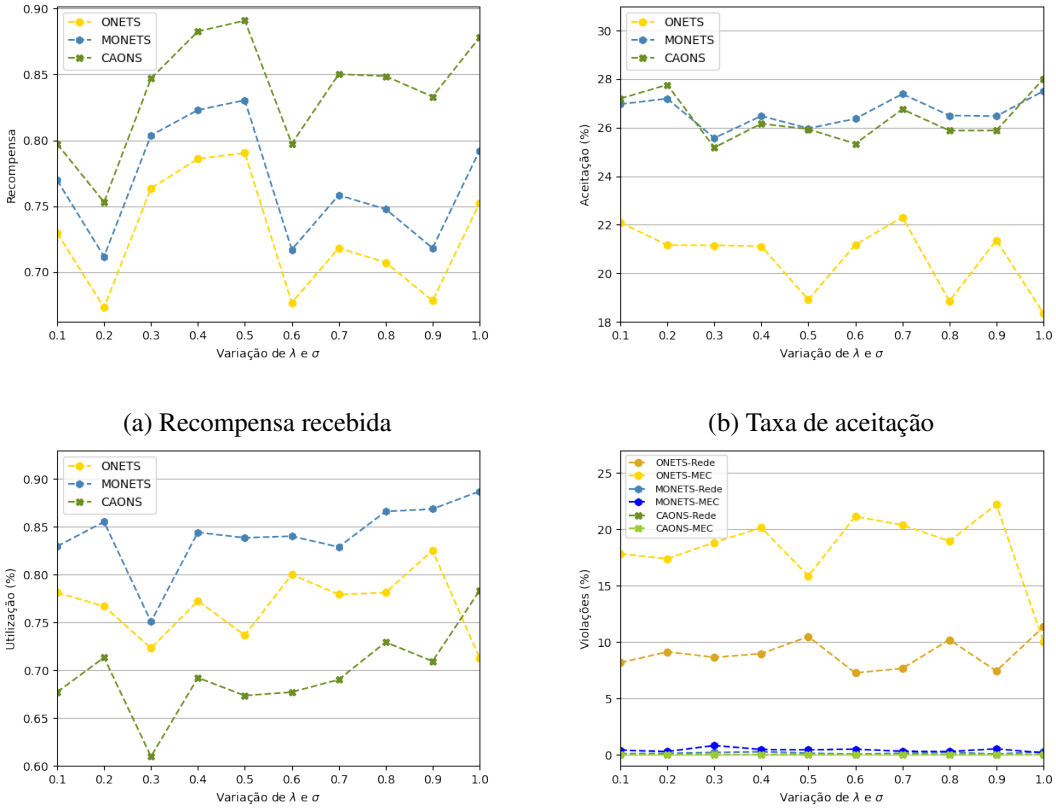



(d) Taxa de Violação

**Figura 3. Custo da variação do número de inquilinos diante de requisições de criação de *slices* envolvendo aplicações variadas com demanda máxima.**

Na Figura 4 a recompensa da rede é analisada como função de $\alpha$ e $\sigma$. A Figura 4(a) mostra que a recompensa recebida na rede mantém-se por toda simulação maior que MONETS e ONETS, varia conforme os valores de $\alpha$ e $\sigma$ elevam. Os melhores resultados são obtidos para $\alpha$ e $\sigma$ iguais a 0.5 e 0.5, respectivamente. Essa variação pode ser explicada pelo modo com que a recompensa foi estabelecida em 3 e 4. Valores maiores $\alpha$ e $\sigma$ 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 $\alpha$ e $\sigma$ 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 4(b) demonstra que a aceitação de inquilinos no sistema da estratégia CAONS mantém-se maior que ONETSe MONETS. As Figura 4(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, 4(d) demonstra que CAONS mantém o número de violações próximo a zero, mesmo diante da variação do peso da recompensa. Os parâmetro $\alpha$ e $\sigma$ 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.




(a) Recompensa recebida

(b) Taxa de aceitação

(c) Taxa de utilização

(d) Taxa de Violação

**Figura 4. Custo da variação de $\alpha$ e $\sigma$ diante de requisições de criação de *slices* envolvendo aplicações variadas com demanda máxima.**

## 6. Conclusão

As redes 5G trazem consigo inovações na forma de transmitir e processar informações. Um dos conceitos chaves desse tipo de rede é o *Network Slicing* (NS). Por meio NS, é possível dividir e isolar, de forma fim a fim, recursos dedicados da RAN, na forma de *slices*. Com isto, NS é capaz de criar novas oportunidades de monetização, aceitando inquilinos no sistema que para utilização de *slices* sob demanda. Aproveitando

essa gama de oportunidades, este artigo propõe e analisa o CAONS, um algoritmo on-line que utiliza técnicas de *overbooking* para controle da admissão de inquilinos de maneira eficiente, considerando recursos de comunicação e computação, enquanto aplica penalidades no caso de ocorrências de violações de capacidade. O objetivo é decidir, a partir da recompensa histórica, acerca do conjunto de inquilinos admitidos no sistema, objetivando maximizar o lucro. A solução é baseada em problemas da classe MaB, com restrição de *lock-up* e aprendizado guiado por arrependimentos. Propomos uma solução de baixo custo computacional, que converge em uma solução sub-ótima em tempo polinomial. Os resultados demonstram que a utilização de métricas de penalização em caso de violação dos orçamentos disponíveis aceleram o processo de aprendizagem do algoritmo, alcançando maiores recompensas e reduzindo consideravelmente o número de violações durante a simulação.

Por fim, apesar de termos aqui assumido que, por exemplo, a demanda de cada inquilino seja dada por uma distribuição definida, na prática ela pode variar bastante e, no modelo, poderia ser também representada por dados próprios da operadora. Além disso, tratamos de um problema cujo contexto operacional é constituído por inquilinos que estão em constante movimentação, realizando requisições e finalizando períodos de reserva. A abordagem desse estudo, no entanto, se aplica além dessa ótica, pois o problema é sobretudo de natureza estratégica, e o que temos nos referido como um inquilino, no modelo, pode representar um cenário de inquilinos, agrupados por critérios categóricos do tipo de serviço ofertado, e os respectivos valores de recompensa podem representar medidas estatísticas apropriadas, incluindo algum cenário de interesse particular para análise de decisões.

## Agradecimentos

Este trabalho foi parcial financiado pelo CNPq/CAPES. A RNP apoiou parcialmente, com recursos do MCTIC, nº 01245.010604/2020-14 no âmbito do projeto de Sistemas de Comunicações Móveis 6G do CRR da Inatel e MCTIC/CGI.br/FAPESP por meio do Projeto Slicing Future Internet Infrastructures (SFI2) sob concessão 2018/23097-3.

## Referências

Asawa, M. and Teneketzis, D. (1996). Multi-armed bandits with switching penalties. *IEEE transactions on automatic control*, 41(3):328–348.

Bega, D. et al. (2019). A machine learning approach to 5G infrastructure market optimization. *IEEE Transactions on Mobile Computing*, 19(3):498–512.

Black, P. E. (2005). Greedy algorithm. *Dictionary of Algorithms and Data Structures*, 2:62.

Chen, G., Liew, S. C., and Shao, Y. (2022). Uncertainty-of-information scheduling: A restless multi-armed bandit framework. *IEEE Transactions on Information Theory*.

Cominardi, L., Deiss, T., Filippou, M., Sciancalepore, V., Giust, F., and Sabella, D. (2020). Mec support for network slicing: Status and limitations from a standardization viewpoint. *IEEE Communications Standards Magazine*, 4(2):22–30.

Elayoubi, S. E. et al. (2019). 5G RAN slicing for verticals: Enablers and challenges. *IEEE Communications Magazine*, 57:28–34.

Gao, Z., Han, Y., Ren, Z., and Zhou, Z. (2019). Batched multi-armed bandits problem. *Advances in Neural Information Processing Systems*, 32.

Garivier, A. and Cappé, O. (2011). The kl-ucb algorithm for bounded stochastic bandits and beyond. In Kakade, S. M. and von Luxburg, U., editors, *Proceedings of the 24th Annual Conference on Learning Theory*, volume 19 of *Proceedings of Machine Learning Research*, pages 359–376, Budapest, Hungary. PMLR.

Guan, Z., Ji, K., Bucci Jr, D. J., Hu, T. Y., Palombo, J., Liston, M., and Liang, Y. (2020). Robust stochastic bandit algorithms under probabilistic unbounded adversarial attack. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4036–4043.

Hwang, S. and Park, S. (2017). On the effects of resource usage ratio on data rate in LTE systems. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 78–80.

Li, F., Yu, D., Yang, H., Yu, J., Karl, H., and Cheng, X. (2020). Multi-armed-bandit-based spectrum scheduling algorithms in wireless networks: A survey. *IEEE Wireless Communications*, 27(1):24–30.

Li, R. et al. (2018a). Deep reinforcement learning for resource management in network slicing. *IEEE Access*, 6:74429–74441.

Li, Z., Uusitalo, M. A., Shariatmadari, H., and Singh, B. (2018b). 5G urllc: Design challenges and system concepts. In *2018 15th international symposium on wireless communication systems (ISWCS)*, pages 1–6. IEEE.

Lima, H. V., Bruno, G. Z., Grings, F. H., Both, C. B., Alberti, A. M., Cardoso, K. V., and Correa, S. L. (2022). Controle de admissão para network slicing ciente de recursos de rede e de processamento.

Mahajan, A. and Teneketzis, D. (2008). Multi-armed bandit problems. In *Foundations and applications of sensor management*, pages 121–151. Springer.

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

Ontanón, S. (2013). 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*, volume 9.

Rifai, B. and Supriyanto, E. (2017). Management system failover dengan routing dinamis open shortest path first dan border gateway protocol. *Jurnal Ilmu Pengetahuan dan Teknologi Komputer*, 3:39–46.

Sciancalepore, V., Zanzi, L., Costa-Perez, X., and Capone, A. (2021). Onets: online network slice broker from theory to practice. *IEEE Transactions on Wireless Communications*, 21(1):121–134.

Weisbecker, F. (2013). Status of Linux dynticks. In *9th annual workshop on Operating Systems Platforms for Embedded Real-Time applications*.

Zhang, S. (2019). An overview of network slicing for 5G. *IEEE Wireless Communications*, 26(3):111–117.