Seleção de RICs com Alocação de Recursos na Arquitetura *Open* RAN usando DRL

Ricardo Bruno Osés de Oliveira, Maykon Renan Pereira da Silva e Flávio Geraldo Coelho Rocha

***Resumo*— Este trabalho propõe uma abordagem com *Deep Reinforcement Learning (DRL)* para otimizar a alocação de recursos para nós inteligentes em redes *Open* RAN. O foco está na seleção eficiente de nós near-RT-RICs considerando tipos distintos de fatias da rede. O agente DDPG decide quais nós utilizar, como alocar recursos computacionais para o processamento das tarefas nos near-RT-RICs e como alocar a taxa de transmissão para comunicação com o non-RT-RIC, respeitando restrições de latência e custo. Os resultados mostram que a abordagem minimiza o custo computacional e de comunicação atendendo aos requisitos dos serviços.**

***Palavras-Chave*— *Open* RAN, RIC, DRL, DDPG, SCA.**

***Abstract*— This work proposes a Deep Reinforcement Learning (DRL) approach to optimize resource allocation for intelligent nodes in Open RAN networks. The focus is on the efficient selection of near-RT RIC nodes, taking into account different types of network slices. The DDPG agent decides which nodes to use, how to allocate computational resources for task processing at the near-RT RICs, and how to allocate transmission rates for communication with the non-RT-RIC, while meeting latency and cost constraints. The results show that the proposed approach minimizes processing and communication resources while meeting service requirements.**

***Keywords*— *Open* RAN, RIC, DRL, DDPG, SCA.**

I. Introdução

As redes móveis de quinta geração (5G) são caracterizadas por casos de uso altamente críticos e heterogêneos que incluem Banda Larga Móvel Aprimorada (*Enhanced Mobile Broad-Band* – eMBB), Comunicações Ultra-confiáveis e de Baixa Latência (*Ultra-Reliable and Low-Latency Communications* – URLLC) e Comunicações em Massa do tipo Máquina (*Massive Machine Type Communications - mMTC*) [1]. Entre esses, o URLLC é particularmente desafiador, devido aos requisitos rigorosos de latência impostos por aplicações como veículos autônomos, realidade aumentada, controle de drones e automação industrial [2], [3], [4]. Para atender simultaneamente a essas demandas, a arquitetura de rede passou a adotar o paradigma de fatiamento de rede (*Network Slicing*), que permite segmentar a infraestrutura física em fatias lógicas com diferentes níveis de Qualidade de Serviço (*Quality of Service* – QoS) [5], [6]. Espera-se que o fatiamento de rede continue sendo fundamental para habilitar serviços avançados no pós-5G, como no 6G, visto que a complexidade dessas redes móveis, com serviços altamente heterogêneos, requisitos ainda mais exigentes, e com opções de escolha de arquiteturas desagregadas e virtualizadas, não poderá ser gerenciada de maneira eficiente apenas por tecnologias das redes legadas.

Nesse contexto, a Inteligência Artificial (IA), especialmente o Aprendizado por Reforço Profundo (*Deep Reinforcement Learning* – DRL), destaca-se como uma ferramenta promissora para aprimorar o processo de tomada de decisão na alocação de recursos para as fatias da rede [7], [8], [9]. Essa capacidade é particularmente relevante na arquitetura Aberta de Rede de Acesso por Rádio (*Open Radio Access Network* – *Open* RAN), padronizada pela O-RAN *Alliance* [10], que promove a desagregação dos componentes da RAN e introduz o Controlador Inteligente da RAN (*RAN Intelligent Controller* – RIC), com duas versões: o controlador em tempo próximo do real (*Near-Real-Time RIC* – near-RT-RIC) e o controlador em tempo não real (*Non-Real-Time RIC* – non-RT-RIC), ambos responsáveis por implementar estratégias de otimização em diferentes escalas temporais. Esses controladores possibilitam a aplicação distribuída de modelos de IA em escalas de tempo que variam de 10 ms a 1 s, promovendo o controle da RAN por meio de interfaces abertas, como a interface E2.

Trabalhos Relacionados. A alocação de recursos em *Open* RAN com fatiamento de rede tem sido estudada na literatura [11], [12], [13], com ênfase em eficiência e flexibilidade. Pesquisas recentes, como [14], destacam o papel dos RICs na orquestração inteligente de recursos para atender às demandas de redes 6G. Em [15] os autores propõem uma abordagem conjunta para seleção de treinadores dentro de um conjunto de near-RT-RICs distribuídos utilizando aprendizado federado (*Federated Learning*) e resolvendo o problema formulado usando o método de Aproximação Convexa Sucessiva (*Successive Convex Approximation* – SCA). Em [16], os autores propõem o uso de near-RT-RICs para otimizar a alocação de recursos em múltiplos nós de borda. Para isso, formularam um problema de otimização para minimizar o *offloading*, o roteamento no *fronthaul* e os atrasos computacionais na arquitetura *Open* RAN.

Contribuições. Este artigo propõe uma abordagem baseada em DRL, utilizando o algoritmo *Deep Deterministic Policy Gradient* (DDPG), para otimizar a alocação de recursos em redes *Open* RAN. A proposta consiste na seleção de nós near-RT RICs com diferentes capacidades computacionais e associados a diversas fatias de rede, buscando atender de forma eficiente às demandas heterogêneas da rede. Para isso, o agente DDPG toma decisões quanto a: i) seleção de nós participantes; ii) alocação de taxa de transmissão; iii) e distribuição de recursos computacionais. O objetivo é reduzir os custos operacionais, computacionais e de comunicação, e garantir que os requisitos de QoS (latência máxima e vazão mínima) sejam respeitados, considerando também as restrições de comunicação e controle impostas pela arquitetura O-RAN.

Ricardo B. O. de Oliveira, Maykon R. P. da Silva, e Flávio G. C. Rocha, da Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Universidade Federal de Goiás (UFG), Goiânia, Goiás, Brasil, e-mail: ricardo_oliveira@discente.ufg.br, maykonrenan@discente.ufg.br, flavio-gcr@ufg.br. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

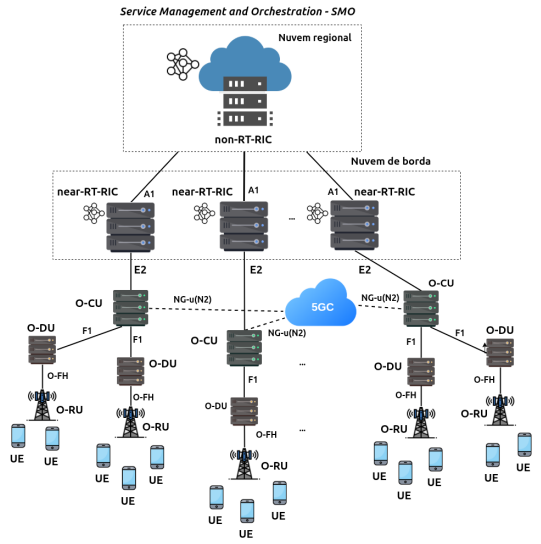

Fig. 1. Modelo do sistema *Open* RAN

O artigo está estruturado da seguinte forma: a Seção II apresenta o modelo do sistema baseado na arquitetura *Open* RAN e a formulação do problema; a Seção III descreve a solução proposta utilizando DRL com DDPG; a Seção IV avalia o desempenho da abordagem em comparação ao método SCA; por fim, a Seção V apresenta as conclusões obtidas e as perspectivas para trabalhos futuros.

II. Modelo do Sistema e Formulação do Problema

Neste trabalho, considera-se uma rede *Open* RAN composta por uma nuvem regional onde está posicionado o orquestrador responsável pela gerência e orquestração de serviços (*Service Management and Orchestration* – SMO) com um controlador inteligente da RAN non-RT-RIC. Esses elementos centrais interagem com um conjunto de $\mathcal{M} = \{1, 2, \ldots, M\}$ controladores near-RT-RICs distribuídos em diversos nós da rede, conforme ilustrado na Figura 1.

Nesse cenário, cada near-RT-RIC atua como unidade local de decisão operando em ciclos entre 10 ms e 1 s, coordenando dinamicamente elementos da rede, como as Unidades Centralizadas (*Centralized Units* – CUs) e Unidades Distribuídas (*Distributed Units* – DUs), por meio da interface E2. Suas funções incluem alocação de recursos de rádio, *handover*, gerenciamento de fatias e controle de políticas de QoS, executadas por módulos *plug-and-play* chamados *xApps*.

O non-RT-RIC desempenha funções de orquestração e alocação de recursos, como largura de banda e capacidade de processamento de forma otimizada, com atuação em escalas de tempo superiores a 1 s. Posicionado no SMO da rede *Open* RAN, o non-RT-RIC é responsável pela otimização global e balanceamento de carga entre os nós da rede com suporte de aplicações baseadas em IA/ML chamadas *rApps*, que envolvem coleta de dados, treinamento e implantação de modelos preditivos. A comunicação entre o non-RT-RIC e os near-RT-RICs é feita pela interface A1.

A. Modelo de Alocação de Recursos

Neste modelo, considera-se que as tarefas processadas nos near-RT RICs correspondem àquelas necessárias para a execução de *xApps* responsáveis por diversas funções na rede, tais como decisões de alocação de recursos de rádio, controle de *handover*, e gerenciamento de fatias para garantia de QoS.

O custo associado à execução dessas tarefas é composto por dois componentes principais:

- **Custo computacional:** relacionado ao processamento de dados e à execução de modelos de ML voltados à otimização da rede, considerando a limitação da capacidade de CPU local;
- **Custo de comunicação:** associado à transmissão dos parâmetros resultantes da execução dos *xApps* para o non-RT-RIC, por meio da interface A1, necessária para manter a coordenação global e garantir conformidade com as políticas definidas na camada de orquestração.

A comunicação entre Near-RT-RIC e Non-RT-RIC é modelada como um canal compartilhado com taxa total B, onde para cada nó é alocada uma fração b_m de B. Formalmente, seja $b_m \in [0, 1]$ a fração de taxa alocada ao near-RT-RIC $m \in \mathcal{M}$, de modo que a taxa efetivamente atribuída é $b_m B$. A alocação deve satisfazer $\sum_{m\in\mathcal{M}} b_m = 1$. Para garantir a viabilidade do sistema, impõe-se que cada near-RT-RIC ativo receba, no mínimo, uma fração b_{min} da taxa total, ou seja, $b_m \geq b_{min}$. O custo total associado ao uso da taxa de comunicação por cada near-RT-RIC é proporcional à quantidade de taxa alocada. Sendo assim, considerando $b_m B$ como a taxa de transmissão alocada ao near-RT-RIC m, o custo de comunicação é dado por:

$$R_m^{co} = p_{tr} \sum_{m=1}^{M} b_m B, \tag{1}$$

onde p_{tr} é o peso do custo de uso da taxa de transmissão. Para cada near-RT-RIC m, denotamos R^{cp} como o custo de recurso computacional necessário para processar as tarefas locais, que depende dos recursos computacionais disponíveis localmente em um *host* e da quantidade de dados a ser processada. Cada near-RT-RIC usa a frequência do ciclo da CPU do *host* para processar o conjunto de dados local D_m. Seja f_m a frequência de operação (em ciclos por segundo) do *host* do m-ésimo near-RT-RIC, p_c é o peso do custo de uso por unidade de tempo e c_m os ciclos de CPU necessários para processar um bit de dados, o custo total dos recursos computacionais é:

$$R_m^{cp} = p_c \sum_{m=1}^{M} \frac{D_m c_m}{f_m}. \tag{2}$$

O custo total do sistema, R^{total} é dado pela soma do custo computacional R_m^{cp} com o custo de comunicação R_m^{co}.

B. Modelo de Latência

A latência total de cada near-RT-RIC em cada nó é o somatório do tempo de processamento local com o tempo de comunicação necessário para transmitir as informações processadas ao non-RT-RIC por meio da interface A1.

O tempo para que o near-RT-RIC m processe sua carga de dados local D_m é:

$$T_m^{cp} = \frac{D_m c_m}{f_m}. \tag{3}$$

Após o término da computação local, o near-RT-RIC m transmite dados de tamanho d_m, dado em bits, para o non-RT-RIC utilizando uma fração b_m da taxa de transmissão total B, dado em bps. O tempo necessário para a transmissão é:

$$T_m^{co} = \frac{d_m}{b_m B}. \tag{4}$$

A latência total do m-ésimo near-RT-RIC é:

$$T_m = T_m^{cp} + T_m^{co} = \frac{D_m c_m}{f_m} + \frac{d_m}{b_m B}. \tag{5}$$

A latência total de transmissão de todos os nós near-RT RIC para o non-RT-RIC é:

$$T^{total} = T_{\max} = \max T_m; \forall m \in \mathcal{M}. \tag{6}$$

C. Formulação do Problema

O objetivo é minimizar o custo total relacionado ao uso de recursos computacionais e de comunicação, atendendo aos requisitos de latência definidos pela *Open* RAN. Essa otimização tem como restrições a capacidade computacional, os recursos de transmissão e os requisitos de latência e de banda mínima específicos de cada tipo de fatia. A formulação do problema é expressa da seguinte forma:

$$\min_{\mathbf{b},\mathbf{R}} \quad (1-\rho)(R^{cp} + R^{co}) + \rho T^{total}, \tag{7}$$

sujeito à:

$$\sum_{m=1}^{M} b_m B \leq B, \tag{8}$$

$$\sum_{m=1}^{M} b_m \leq 1, \tag{9}$$

$$b_{min} \leq b_m \leq 1, \quad \forall m \in \mathcal{M}, \tag{10}$$

$$T^{max} \leq T_{deadline}. \tag{11}$$

O parâmetro ρ ajusta a prioridade entre minimizar o custo total de operação e o atendimento aos prazos das tarefas já que esses dois componentes são conflitantes. A função objetivo (7) busca o equilíbrio entre o custo operacional (computação e comunicação) e a latência máxima dos near-RT-RICs, ponderado por ρ. A Restrição (8) limita a taxa de transmissão total alocada para as tarefas em cada nó. A Restrição (9) define b_m como a soma das frações da taxa de transmissão que deve ser menor ou igual a 1. A restrição (10) define os limites de alocação de taxa de transmissão por near-RT RIC e (11) restringe a latência ao valor máximo permitido.

O problema formulado em (7) é um problema de otimização não convexo porque a função objetivo e a restrição (8) são funções não convexas. O problema é NP-difícil, não sendo possível obter soluções exatas em tempo polinomial, por isso ele foi resolvido por meio de um modelo de DRL.

III. Alocação de Recursos com Base em Aprendizado por Reforço

Na abordagem proposta utilizando DRL, dado o conjunto $\mathcal{M} = \{1, 2, \ldots, M\}$ de nós near-RT-RICs, como ilustrado na Figura 1, busca-se determinar uma política eficiente que selecione dinamicamente quais nós devem participar da execução das tarefas e como alocar a taxa de transmissão $\mathbf{b} = [b_1, b_2, ..., b_M]$ entre os nós selecionados. O objetivo é minimizar o custo total do sistema, composto pelos custos de computação e comunicação, além das penalidades associadas à violação de requisitos de latência específicos de cada *slice*.

Definição do Ambiente de Aprendizado por Reforço. O problema de seleção de nós e alocação de recursos no ambiente *Open* RAN é modelado como um Processo de Decisão de Markov (*Markov Decision Process* – MDP), definido pelo conjunto $(\mathcal{S}, \mathcal{A}, \mathcal{R}, P)$, onde: $\mathcal{S}$ é o espaço de estados observáveis do sistema; $\mathcal{A}$ é o espaço de ações, correspondente às decisões de seleção e alocação de recursos; $\mathcal{R}$ é a função de recompensa que avalia o impacto da ação tomada; e P representa a função de transição entre estados.

Espaço de Estados ($\mathcal{S}$). O vetor de estado s_t, no instante t, é composto por características da rede e de cada near-RT-RIC: a frequência da CPU f_m, o volume de dados locais D_m, a prioridade da fatia (uRLLC, eMBB e mMTC), o uso da taxa de transmissão em relação ao total disponível e a diferença entre a latência estimada T_m e o *deadline* operacional do near-RT-RIC. Essa estrutura de estado permite ao agente uma visão abrangente do sistema, incluindo aspectos computacionais, de comunicação, a criticidade do serviço e as restrições temporais, permitindo decisões adaptativas.

Espaço de Ações ($\mathcal{A}$). A ação contínua a_t do agente é composta por dois vetores:

- O vetor de seleção de nós ativos $\mathbf{a}^{\text{sel}} \in [0,1]^M$ indica a probabilidade de ativação de cada near-RT-RIC na rodada corrente. Se $a_m^{\text{sel}} = 1$, o nó está ativo e contribui para o custo computacional, custo de comunicação e latência total. Se $a_m^{\text{sel}} = 0$, o nó está inativo e não gera custos nem impacto de latência na rodada considerada.
- O vetor de alocação de taxa $\mathbf{b} \in [b_{\min}, 1]^M$ representa a fração da taxa total B alocada aos nós selecionados.

Essa estrutura de ação permite ao agente decidir, de forma conjunta, quais nós devem executar suas tarefas e quanto de capacidade de transmissão cada um deve receber.

Política de Controle. A política de controle define a estratégia de decisão a ser utilizada na seleção dinâmica dos near-RT-RICs e qual fração da taxa de transmissão total B será alocada para cada um deles, com base no estado atual da rede. Este processo é representado pela função determinística:

$$\mu(s_t) : \mathcal{S} \rightarrow \mathcal{A}, \tag{12}$$

que associa um estado observado s_t a uma ação a_t contínua.

Transição de Estados. A transição do estado do sistema ocorre em função da ação executada pelo agente, considerando os efeitos do processamento local e da comunicação dos resultados com o non-RT-RIC. A latência total da rodada é a latência máxima dos near-RT-RICs ativos:

$$T^{\max} = \max_m \left(\frac{D_m c_m}{f_m} + \frac{d_m}{b_m B} \right) a_m^{sel}, \tag{13}$$

onde o primeiro termo representa o tempo de computação local e o segundo termo o tempo de comunicação necessário para transmitir parâmetros ao non-RT-RIC.

Função de Recompensa ($\mathcal{R}$). A função de recompensa reflete os objetivos principais do sistema: minimizar os custos com os recursos e penalizar violações de requisitos de latência, ao mesmo tempo em que se prioriza fatias mais críticas, como a URLLC. A recompensa em cada rodada é definida como:

$$r_t = -\left((1-\rho)(R_{\text{cp}} + R_{\text{co}}) + \rho T_{\max}\right) + \sum_{m=1}^{M} a^{sel}\omega_m, \quad (14)$$

onde ω_m é o peso da prioridade do tipo de *slice* do nó m.

Algoritmo DDPG. O algoritmo DDPG foi utilizado para treinar a política de controle proposta. Esse algoritmo é particularmente adequado para tarefas com espaços de ações contínuas [17], como a alocação de taxa de transmissão $b_m \in [b_{\min}, B]$ para cada nó m, onde o vetor de ação contínua $\mathbf{b}_t = [b_1, b_2, \ldots, b_M]$ precisa ser aprendido.

O DDPG é um algoritmo do tipo ator-crítico que combina os métodos de valor, como o *Q-Learning*, com gradientes de políticas determinísticas. A arquitetura do DDPG é composta por dois conjuntos de redes neurais: um par de redes para o ator (*policy network*) e um par de redes para o crítico (*Q-network*). A rede principal do ator, denotada por $\mu(s_t|\theta^{\mu})$, é responsável por gerar ações com base no estado atual. Já sua rede alvo $\mu'(s_t|\theta^{\mu'})$ fornece estabilidade durante o treinamento ao suavizar as atualizações.

De forma semelhante, a rede principal do crítico $Q(s_t, a_t|\theta^{Q})$ estima a qualidade de uma ação em determinado estado, enquanto a rede alvo $Q'(s_t, a_t|\theta^{Q'})$ é usada no cálculo da função de perda para atualização estável. O estado no tempo t é representado por s_t, enquanto a_t corresponde à ação escolhida pela política. Os parâmetros θ^{μ} e θ^{Q} correspondem às redes principais, e $\theta^{\mu'}$ e $\theta^{Q'}$ às redes alvo.

Durante a exploração, a política determinística gera uma ação com ruído adicionado:

$$\mathbf{b}_t = \mu_\theta(s_t) + \mathcal{N}_t. \quad (15)$$

O termo $\mathcal{N}_t$ representa o ruído exploratório, como ruído gaussiano, que estimula a exploração do espaço de ações durante o treinamento. A rede crítica estima a expectativa de retorno acumulado com desconto:

$$Q_\phi(s_t, a_t) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right], \quad (16)$$

em que γ é o fator de desconto que pondera recompensas futuras. A função de perda que atualiza o crítico é dada por:

$$L(\phi) = E\left[\left(Q_\phi(s_t, a_t) - y_t\right)^2\right], \quad (17)$$

com o alvo definido como:

$$y_t = r_t + \gamma Q_{\phi'}(s_{t+1}, \mu_{\theta'}(s_{t+1})). \quad (18)$$

A atualização da política é feita utilizando o gradiente da função objetivo, segundo a seguinte expressão:

$$\nabla_\theta J = E\left[\nabla_a Q_\phi(s, a)\big|_{a=\mu_\theta(s)} \nabla_\theta \mu_\theta(s)\right]. \quad (19)$$

Por fim, para garantir estabilidade durante o aprendizado, os parâmetros das redes alvo são atualizados progressivamente a partir dos parâmetros das redes principais:

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}, \quad \theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \quad (20)$$

com τ é um valor pequeno (geralmente $\tau \ll 1$) que controla a taxa de atualização suavemente. O agente DDPG é executado no Non-RT-RIC, que aloca recursos e recebe métricas dos Near-RT-RICs para *feedback* contínuo.

IV. Resultados

As simulações deste trabalho foram conduzidas usando *Python* 3.10 com *PyTorch* 2.1.0, executadas em um *Intel Core i7-1255U* × 12 (2,30 GHz), 16 GB de RAM e sem GPU dedicada. O agente DDPG utilizou duas camadas totalmente conectadas com 256 neurônios e ativação *ReLU*. Os hiperparâmetros foram: taxa de aprendizado ($lr = 10^{-3}$), fator de desconto ($\gamma = 0{,}99$), atualização suave das redes-alvo ($\tau = 0{,}005$), mini-lote de 256, buffer de *replay* com 10^5 amostras, ruído de ação modelado como $\mathcal{N}(0, 0{,}1)$, atraso da política configurado para 2 atualizações por atualização do crítico, e espaço de ação normalizado habilitado. O treinamento foi executado ao longo de 10^4 etapas, considerando três tipos de fatias (URLLC, eMBB e mMTC). Os demais parâmetros utilizados nas simulações encontram-se resumidos na Tabela I.

TABELA I
PARÂMETROS DE SIMULAÇÃO

Parâmetro	Descrição	Valor
M	Número máximo de RICs locais	50
B	Taxa de transmissão total	10 Mbps
c_m	Taxa de processamento	15 ciclos/bit
f_m	Frequência máxima da CPU	$\sim U(1,1{,}6)$ GHz
p_{tr}	Custo unitário de transmissão	1
p_c	Custo unitário de computação	1
D_m	Tamanho do conjunto de dados	$\sim U(5, 10)$ MB
d_m	Tamanho do vetor de atualização	$\sim U(10, 150)$ Kb
b_{min}	Taxa mínima de transmissão	0,1 Mbps
ρ	Parâmetro de balanceamento	0,9
$T_{deadline}$	*Deadline* do near-RT-RIC	1 s
ω_m	Prioridade por *slice*	(3, 1,5, 0,5)

Para comparação, utilizou-se como referência o método SCA de [15]. As métricas analisadas são: o custo total de operação, o número médio de nós near-RT-RICs selecionados e a distribuição dos recursos de taxa entre esses nós obedecendo as restrições de latência da arquitetura *Open* RAN, bem como das fatias de rede.

A Figura 2 compara o desempenho dos algoritmos DDPG e SCA em relação ao número de near-RT-RICs selecionados ao longo do tempo. O DDPG exibe um comportamento dinâmico, ajustando-se continuamente às condições da rede e alternando a priorização entre diferentes near-RT-RICs convergindo para uma política mais eficiente ao longo do tempo, resultado direto do aprendizado contínuo e da capacidade de otimização do algoritmo. Em contraste, o SCA mantém um padrão sem tanta variação, o que reflete sua natureza heurística e menos adaptativa. Enquanto o DDPG se beneficia de sua abordagem baseada em DRL para otimizar a seleção ao longo do tempo, o SCA fica limitado por regras pré-definidas.

Em relação ao tempo necessário para a tomada de decisão, a Figura 2 compara os tempos de execução do DDPG e SCA.

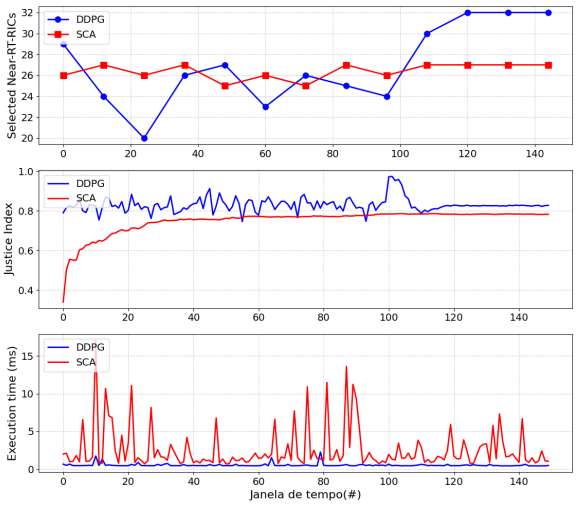

Fig. 2. Comparação dos resultados obtidos com DDPG e SCA

O DDPG demonstrou desempenho melhor do que o SCA, pois o tempo para tomada de decisão é menor e mais estável, sendo mais apropriado para aplicações sensíveis à latência.

Neste trabalho, o índice de justiça de Jain [18] foi utilizado para avaliar o quão equitativamente os near-RT-RICs foram selecionados. A Figura 2 mostra que O DDPG apresenta valores mais altos para o índice de justiça, indicando sua capacidade de adaptar a escolha de nós de maneira mais justa e variada. Já o SCA mantém uma curva mais estável, mas com justiça inferior, sugerindo preferência por determinados nós.

A Tabela II apresenta a comparação entre os principais indicadores de desempenho, incluindo número de near-RT-RICs selecionados, utilização de taxa de transmissão, custo computacional e custo de transmissão.

TABELA II

RESULTADOS FINAIS DA ALOCAÇÃO DE RECURSOS

Métrica	**DDPG**	**SCA**
RICs Selecionados	32	26
Taxa Total Utilizada	83,7%	100%
Média de Taxa por RIC	0,27 Mbps	0,4 Mbps
Custo Computacional	2,65	3,10
Custo de Transmissão	8,37 Mbps	10 Mbps

Analisando os resultados apresentados na Tabela II observa-se que o algoritmo DDPG demonstrou melhor desempenho em comparação ao SCA, selecionando mais near-RT-RICs e utilizando os recursos de forma mais eficiente. Ele consumiu 83,7% da taxa de transmissão disponível, enquanto o SCA usou 100%, indicando que o DDPG reserva capacidade de transmissão para variações de tráfego. A média de taxa alocada por nó também foi menor com o DDPG (0,27 Mbps contra 0,4 Mbps). Além disso, o DDPG teve menor custo computacional e de transmissão, mesmo com mais nós selecionados.

V. CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho propôs e avaliou um algoritmo de alocação de recursos baseado em DRL com DDPG para solucionar um problema de otimização em redes *Open* RAN. O objetivo é maximizar a quantidade de near-RT-RICs selecionados sem exceder as restrições de taxa de transmissão e latência. O problema é NP-difícil e o DDPG foi utilizado devido à sua capacidade de aprender políticas adaptativas em ambientes dinâmicos e com múltiplas restrições, otimizando a alocação de recursos em tempo real. Os resultados demonstraram que o DDPG superou o método baseado em SCA, proporcionando maior número de near-RT-RICs selecionados, menor utilização da capacidade de transmissão e menores custos computacionais e de transmissão. Esses resultados evidenciam o potencial do DRL para otimizar a gestão de recursos em ambientes complexos e dinâmicos. Como continuidade deste trabalho, pretende-se propor a integração do Aprendizado Federado (*Federated Learning* - FL) no processo de alocação de recursos, permitindo o treinamento colaborativo de modelos entre múltiplos near-RT-RICs de maneira descentralizada e preservando a privacidade dos dados locais.

REFERÊNCIAS

[1] F. Song, J. Li, C. Ma, Y. Zhang, L. Shi, and D. N. K. Jayakody, "Dynamic virtual resource allocation for 5g and beyond network slicing," *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 215–226, 2020.

[2] N. Alliance, "5g white paper," *Next generation mobile networks, white paper*, vol. 1, no. 2015, 2015.

[3] N. Alliance, "Verticals urllc use cases and requirements," *NGMN Alliance*, 2019.

[4] A. Zahemszky, "5g e2e technology to support verticals urllc requirements," *NGMN Alliance, White Paper*, 2020.

[5] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[6] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, *et al.*, "Network slicing to enable scalability and flexibility in 5g mobile networks," *IEEE Communications magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[7] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Dynamic sdn-based radio access network slicing with deep reinforcement learning for urllc and embb services," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2174–2187, 2022.

[8] J. A. Hurtado Sánchez, K. Casilimas, and O. M. Caicedo Rendon, "Deep reinforcement learning for resource management on network slicing: A survey," *Sensors*, vol. 22, no. 8, p. 3031, 2022.

[9] Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, "Energy-efficient deep reinforcement learning assisted resource allocation for 5g-ran slicing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 856–871, 2021.

[10] O-RAN Alliance, "O-RAN.WG1.TS.OAD-R004-v13.00," 2025. Accessado: 05. março, 2025.

[11] M. K. Motalleb, V. Shah-Mansouri, and S. N. Naghadeh, "Joint power allocation and network slicing in an open ran system," *ArXiv*, vol. abs/1911.01904, 2019.

[12] M. K. Motalleb, V. Shah-Mansouri, S. Parsaeefard, and O. L. A. López, "Resource allocation in an open ran system using network slicing," *IEEE Transactions on Network and Service Management*, vol. 20, pp. 471–485, 2023.

[13] A. Filali, B. Nour, S. Cherkaoui, and A. Kobbane, "Communication and computation o-ran resource slicing for urllc services using deep reinforcement learning," *IEEE Communications Standards Magazine*, vol. 7, no. 1, pp. 66–73, 2023.

[14] Q. Wang, Y. Liu, Y. Wang, X. Xiong, J. Zong, J. Wang, and P. Chen, "Resource allocation based on radio intelligence controller for open ran toward 6g," *IEEE Access*, vol. 11, pp. 97909–97919, 2023.

[15] A. K. Singh and K. K. Nguyen, "Joint selection of local trainers and resource allocation for federated learning in open ran intelligent controllers," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1874–1879, IEEE, 2022.

[16] A. Ndikumana, K. K. Nguyen, and M. Cheriet, "Federated learning assisted deep q-learning for joint task offloading and fronthaul segment routing in open ran," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3261–3273, 2023.

[17] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, pp. 387–395, Pmlr, 2014.

[18] R. K. Jain, D.-M. W. Chiu, W. R. Hawe, *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, no. 1, 1984.