Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Aplicação de técnicas de "Deep Reinforcement Learning" na alocação de recursos de um sistema 5G para a comunicação D2D

Gabriel P. de Freitas Cardoso

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Engenharia Elétrica

Orientador
Prof. Dr. Paulo Henrique Portela de Carvalho

Brasília
2021

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Aplicação de técnicas de "Deep Reinforcement Learning" na alocação de recursos de um sistema 5G para a comunicação D2D

Gabriel P. de Freitas Cardoso

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Engenharia Elétrica

Prof. Dr. Paulo Henrique Portela de Carvalho (Orientador)
Universidade de Brasília

Prof. Dr. Paulo Roberto de Lira Gondim
Universidade de Brasília

Prof. Dr. Robson Domingos Vieira
FAPEG/GO

Bacharelado em Engenharia Elétrica

Brasília, 12 de novembro de 2021

Dedicatória

Dedico esse trabalho à minha avó Maria José Cardoso por me ensinar o valor do trabalho intelectual, à minha avó Francisca Maria de Jesus por me ensinar o valor do trabalho prático e ao meu tio Djalma Cardoso, por me mostrar o valor da união de ambos.

Agradecimentos

Agradeço, primeiramente, a Deus por me proporcionar essa oportunidade tão valiosa para o meu aprendizado.

Em seguida, agradeço à minha mãe Jane Margarete Pimenta e ao meu pai Luiz Fernando de Freitas Cardoso por todo apoio e dedicação dispendidos antes e durante a minha formação acadêmica, bem como aos meus irmãos Matheus Pimenta de Freitas Cardoso e Ludmila Pimenta de Freitas Cardoso por sempre me servirem de inspiração.

Agradeço à minha namorada Stefani da Mota Ribeiro por sempre me apoiar nos momentos de dificuldade, renovando minhas forças para continuar a cada dia e celebrando as minhas conquistas como se fossem as suas.

Agradeço, também, ao orientador deste trabalho, o Prof. Paulo Henrique Portela de Carvalho por todo empenho com a pesquisa acadêmica e com a educação e desenvolvimento de seus alunos.

Por fim, agradeço à Prof.a Artemis Marti Ceschin e ao Prof. Luis Fernando Ramos Molinaro, por me abrirem as portas para a pesquisa científica.

Resumo

Considerando a importância de aproveitar ao máximo os recursos disponíveis para as próximas gerações do sistema móvel, este trabalho propõe dois algoritmos inteligentes para realizar a alocação de recursos de um sistema em um ambiente urbano com comunicações prioritárias no *uplink* compartilhando espectro com comunicações *device-to-device* (D2D) em modo *underlay* em cenário *in-coverage*.

Os algoritmos desenvolvidos tiveram estrutura monoagente e foram treinados utilizando técnicas de *Deep Reinforcement Learning* com política determinística, mais especificamente o DDPG e o TD3, devido à capacidade de otimizar problemas de alta complexidade e de definir ações de natureza contínua.

Os modelos desenvolvidos mostraram-se eficazes na proteção das comunicações prioritárias e na maximização das taxas de transmissão das comunicações D2D, realizando o controle de potência e a alocação de espectro de comunicações de um sistema móvel.

Por outro lado, o desempenho dos algoritmos propostos mostrou-se dependente da dimensão do problema, de forma que, à medida que a dimensionalidade aumentou, o desempenho do modelo ficou comprometido. Isso é um reflexo das limitações do processo de treinamento dos algoritmos DRL e da dificuldade de realizar alocação através de um modelo centralizado.

As principais contribuições deste trabalho são propor um modelo que realiza a alocação completa de recursos, controlando a potência e alocando o espectro, por meio de um esquema centralizado treinado com algoritmos de DRL. Além disso, a realização da alocação com a possibilidade de as comunicações D2D usarem mais de um RB aumenta as possibilidades de alocação, o que é discutido neste trabalho.

Palavras-chave: Aprendizado de máquina, Aprendizado por reforço profundo, 5G, D2D, Alocação de recursos

Abstract

Considering the importance of utilize the most of the resources available for the next generation of the mobile system, this work proposes two intelligent algorithms to perform a resource allocation of a system in urban environment with priority communications on uplink sharing spectrum with device-to-device communications in uderlay mode and in-coverage (D2D).

The developed mono-agent algorithms and were trained using Deep Reinforcement Learning techniques with deterministic policy, more specifically the DDPG and the TD3, due to the ability to optimize high complexity problems and the possibility of defining continous actions.

The developed models proved to be effective in protecting priority communications and maximizing the transmission rates of D2D communications, performing power control and spectrum allocation of a mobile system.

On the other hand, the performance of the proposed algorithms depends on the dimensionality of the problem, so that, as the dimensionality increased, the performance of the model was compromised. This is a reflection of the limitations of the DRL algorithm training process and the difficulty of performing allocation through a centralized model.

The main contributions of this work are to propose a model that performs a complete resource allocation, controlling power and allocating spectrum, through a centralized scheme trained using DRL algorithms. Furthermore, the possibility of communications D2D using more than one RB increases the possibilities of allocation, which is discussed in this work.

Keywords: Machine Learning, Deep Reinforcement Learning, 5G, D2D, Resource allocation

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

3GPP *3rd Generation Partnership Project.*

AI *Artificial Inteligence.*

CNN *Convolutional Neural Networks.*

D2D *Device-to-Device.*

DDPG *Deep Deterministic Policy Gradient.*

DL *Deep Learning.*

DNN *Deep Neural Networks.*

DQN *Deep Q-Network.*

DRL *Deep Reinforcement Learning.*

ERB Estação Rádio-Base.

GRU *Gated Recurrent Unit.*

IoT *Internet of Things.*

LSTM *Long Short-Term Memory.*

LTE-A *Long Term Evolution - Advanced.*

ML *Machine Learning.*

OFDM *Orthogonal Frequency Division Multiplexing.*

ProSe *Proximity services.*

QoS *Quality of Service.*

RB *Resource Block.*

RL *Reinforcement Learning.*

RNN *Recurrent Neural Networks.*

SNIR *Signal-to-Noise plus Interference Ratio.*

TD3 *Twin-Delayed DDPG.*

UE *User equipment.*

V2V *Vehicle-to-vehicle.*

1 Introdução

Nas últimas décadas, a grande procura por serviços de telefonia móvel fez com que o Sistema Móvel Pessoal se tornasse o principal sistema de comunicações no mundo. A evolução dos *smartphones* fez com que os aparelhos celulares recebessem múltiplas funções, transformando os celulares em computadores portáteis, cuja função de realizar ligações celulares é apenas mais uma entre várias outras [1].

O aumento do número de usuários e a evolução das ferramentas dos *smartphones* fizeram com que as tecnologias de comunicação também fossem alavancadas. A partir da necessidade de atender a maioria dos usuários de maneira satisfatória, foi preciso desenvolver estratégias inteligentes que permitissem o aumento da cobertura e das taxas de transmissão das comunicações [2].

Esses fatores impulsionaram os avanços ocorridos principalmente na terceira e quarta gerações (3G e 4G) dos sistemas móveis. Para a quinta geração, espera-se que haja grandes desafios, motivados pelos avanços das áreas de Internet das Coisas (*Internet of Things* - IoT), Indústria 4.0 e robótica [3, 4]. Para essas aplicações funcionarem como desejado, é necessário que haja comunicação instantânea com alta confiabilidade, taxas de transmissão elevadas e baixa latência entre os dispositivos do ambiente [4].

Além disso, o aumento de usuários consumindo ou produzindo conteúdos de *streaming*, tendência atual motivada pela popularização dos *e-games* e de *lives* para fins variados, também promove um aumento na demanda de taxas elevadas de transmissão de dados e alta confiabilidade, já que esse tipo de serviço exige ambos os requisitos [5].

Somado a isso, a migração, provocada pela pandemia global de Covid-19, de diversas profissões que antes eram exercidas presencialmente e passaram a ser exercidas virtualmente também é um fator que corrobora para a importância de garantir um sistema confiável, com grande cobertura e condições de atender a todos os usuários de maneira satisfatória.

Nesse contexto, buscando aumentar a cobertura, a confiabilidade e a eficiência espectral do sistema, planeja-se a inserção das chamadas comunicações *Device-to-Device* (D2D) no 5G, tendência que promete o aproveitamento dos recursos do sistema [6].

1.1 Comunicações D2D no 5G

As comunicações D2D permitem que dois dispositivos do sistema móvel se comuniquem sem a necessidade de intermédio da Estação Rádio-Base, transmitindo o sinal diretamente de um dispositivo ao outro. Essas comunicações foram integradas à arquitetura LTE-A do 4G pela *3rd Generation Partnership Project* (3GPP) na Relase 12, sendo a segurança pública a principal aplicação prevista. A Release 13 trouxe mudanças para suportar outras funções do D2D. A Release 14 trouxe recomendações focadas em melhorar a comunicação veicular (V2V). Por fim, a Release 15 trouxe as bases para a aplicação dessas comunicações no 5G, ponto de interesse deste trabalho [6].

Apesar de se mostrar uma alternativa para melhorar o alcance do sistema móvel, as comunicações D2D também produzem outros desafios para o 5G, como o controle da interferência gerada entre os dispositivos e a garantia de que as comunicações primárias, entre um celular e a Estação Rádio-Base, continuarão sendo realizadas de maneira satisfatória.

Para contornar esses dois desafios, é fundamental que se adotem estratégias inteligentes para alocação dos recursos disponíveis, garantindo que os usuários consigam aproveitá-los ao máximo.

1.2 Alocação de recursos em sistemas com comunicações D2D

A alocação de recursos em sistemas de comunicações consiste na união de duas tarefas necessárias: o controle de potência das comunicações e a alocação do espectro disponível para o sistema [7]. A primeira consiste em determinar as potências de transmissão de cada comunicação da célula, enquanto a segunda é a definição das faixas do espectro que cada uma dessas comunicações vai usar [7, 8].

A alocação de recursos se divide em dois tipos principais: alocação com esquema centralizado e alocação com esquema distribuído [9]. Cada um possui suas vantagens e desvantagens: alocações distribuídas são pouco eficientes e de difícil convergência, além do fato de que um conjunto de decisões tomadas independentemente nem sempre cumpre os objetivos gerais do sistema [9]. Por outro lado, alocações centralizadas podem se tornar tarefas muito complexas à medida que a dimensionalidade do sistema aumenta [8]. Este trabalho propõe um modelo de alocação utilizando esquema centralizado.

Além da classificação quanto à estrutura do esquema utilizado, é possível classificar as estratégias de alocação quanto aos objetivos. O controle de potências pode ser dividido em estratégias para minimização das potências de transmissão ou em estratégias para maximização das taxas de transmissão [7]. Já a alocação do espectro pode ser dividida

em estratégias de otimização da eficiência energética, de otimização da eficiência espectral, de maximização das taxas de transmissão e de garantia da Qualidade do Serviço (*Quality of Service* - QoS) [7]. Esse trabalho propõe um modelo de alocação com controle de potência visando maximização das taxas de transmissão e alocação do espectro priorizando a garantia da QoS, mas buscando, de maneira secundária, a otimização da eficiência espectral.

Algumas abordagens foram propostas para a alocação de recursos, usando diferentes métodos para solucionar o problema. Entre essas abordagens, é possível elencar diferentes tipos de algoritmos e suas desvantagens:

- Algoritmos heurísticos, como o *Greedy heuristic* [10], que possuem processos de funcionamento que podem ser infindáveis ou que deixam de alocar algumas comunicações. Além disso, pela natureza da alocação, não é ótima, pois é assumida uma ordem de alocação das comunicações que pode não ser a melhor possível [9].
- Algoritmos baseados em *local search*, como o LORA [11], que apresentam praticamente os mesmos problemas dos algoritmos heurísticos, possuindo mínimos locais distantes da solução ótima. Além disso, os algoritmos de *local search* exigem mais tempo de execução, se comparados aos heurísticos [9].
- Algoritmos baseados em *deferred acceptance*, que gera uma lista de comunicações ordenadas de acordo com a preferência, que, no caso do proposto em [12], era baseado no aumento da proximidade entre dispositivos. Essa abordagem não leva em conta as questões de QoS, o que é uma das prioridades dos sistemas reais [9].
- Algoritmos baseados em grafos, como o proposto por [13], que mostrou bom desempenho, mas também não levou em conta os fatores de QoS [9, 14].

Buscando solucionar as dificuldades abordados acima, as técnicas de Inteligência Artificial se apresentam como alternativas interessantes, já que o desenvolvimento da área possibilitou a criação de algoritmos capazes de executar tarefas de complexidade elevada e com boa generalização [15].

Entre essas técnicas, estão os algoritmos de Aprendizado por Reforço Profundo (*Deep Reinforcement Learning* - DRL), modelos que aprendem a executar ações a partir do recebimento de recompensas. Esses algoritmos vêm se mostrando boas alternativas para solucionar problemas em que não é trivial rotular as entradas que passarão pela rede, o que dificulta a aplicação das técnicas clássicas de aprendizado supervisionado [16].

Nesse sentido, como o modelo proposto objetiva realizar uma alocação que proteja a QoS das comunicações tradicionais e maximize a eficiência espectral das comunicações D2D, o problema se torna multiobjetivo, cuja priorização dos objetivos pode variar de

acordo com a necessidade do sistema. Isso faz com que os algoritmos de DRL sejam recomendáveis para o problema, já que, dependendo de como a função de recompensa seja definida, é possível encontrar uma solução que otimiza os múltiplos objetivos.

Isso foi demonstrado em [17, 18], em que o autor, utilizando algoritmos de DRL, chegou à fronteira de Pareto do problema do caixeiro viajante, considerando a otimização de mais de um objetivo. O artigo demonstrou que, a partir da modificação dos pesos que compõem a função de recompensa, é possível encontrar diferentes soluções pertencentes à fronteira de Pareto. Os resultados alcançados com os algoritmos de DRL superaram as alcançadas utilizando alguns outros algoritmos clássicos de otimização.

1.3 Alocação de Recursos com algoritmos de DRL

Existem, na literatura, alguns trabalhos que propuseram algoritmos de alocação de recursos baseados em algoritmos de DRL. Parte desses trabalhos solucionam apenas uma das duas tarefas da alocação de recursos, se dedicando ou à alocação do espectro [19, 20] ou ao controle de potência [21, 22]. Outros trabalhos já apresentam modelos que executam ambas as tarefas [23].

Alguns desses trabalhos desenvolveram propostas de algoritmos multiagentes como em [19, 22, 23]. Nesses três trabalhos, são desenvolvidos algoritmos capazes de maximizar a taxa de transmissão do sistema, mas a garantia da QoS das comunicações prioritárias não é abordada em nenhum deles.

Além das desvantagens já citadas dos algoritmos multiagentes, no problema de alocação de recursos, é um desafio para esses modelos conseguir proteger um agente em específico, já que o aprendizado dessas técnicas geralmente utiliza técnicas baseadas em competitividade [24].

Além desses modelos multiagente, em [20], é proposto um modelo híbrido, que combina técnicas de aprendizado centralizado e distribuído, que realiza a alocação do espectro para cada comunicação do sistema. Essa proposta consegue proteger a QoS das comunicações prioritárias, enquanto tenta maximizar a taxa de transmissão total do sistema.

Apesar de o algoritmo de [20] mostrar que o seu desempenho é pouco afetado pelo aumento da dimensionalidade do problema, esse modelo realiza apenas a alocação do espectro disponível, não executando o controle de potência das comunicações, que é um grande desafio para técnicas distribuídas.

Por fim, em [21], é proposto um algoritmo monoagente, que realiza o controle de potências das comunicações da célula visando aumentar as taxas de transmissão da célula. O modelo consegue superar alguns alocadores do estado da arte, mas não é analisada a garantia da QoS das comunicações prioritárias e não é feita a alocação do espectro.

Este trabalho propõe um modelo monoagente, que faz a alocação de recursos completa, realizando tanto o controle de potências como a alocação do espectro. Além disso, a proposta deste artigo possibilita que as comunicações D2D utilizem mais de um *Resource Block* para distribuírem a sua potência de transmissão, o que fornece a possibilidade de uma alocação totalmente flexível.

Entretanto, a técnica utilizada para garantir essa flexibilidade também promove um grande aumento de dimensionalidade na saída do algoritmo, o que é um desafio para o processo de aprendizagem de algoritmos de otimização [21].

1.4 Metodologia

A pesquisa se iniciou com uma revisão da literatura buscando compreender as técnicas de *Deep Reinforcement Learning* e os cenários esperados para a aplicação das comunicações D2D em sistemas de comunicação móvel de quinta geração. A partir do estudo da bibliografia, foi desenvolvido um simulador para o sistema de comunicação em que o modelo proposto faria a alocação.

Tanto o simulador quando o algoritmo foram desenvolvidos a partir de arquivos de código utilizando a linguagem de programação Python 3. Não utilizou-se um simulador pronto, de forma que este foi gerado pelo autor, que aplicou os conceitos teóricos da área usando apenas bibliotecas de código aberto para auxílio no tratamento dos dados.

Assim como o simulador, os modelos propostos também foram desenvolvidos pelo autor, mas utilizou-se o Pytorch, biblioteca de código aberto que facilita a utilização de técnicas de *Deep Learning*, para gerar e treinar as redes neurais utilizadas.

Os códigos foram executados localmente e os resultados eram armazenados em outros arquivos para análises posteriores.

1.5 Contribuições

No conhecimento do autor, este trabalho é inovador por ser o primeiro a propor um modelo monoagente que realiza o controle de potência e a alocação de espectro em um sistema com comunicações D2D, baseado na revisão bibliográfica feita. Além disso, também é o primeiro algoritmo capaz de fazer a alocação de forma que as comunicações D2D utilizem mais de um *Resource Block* para distribuírem a sua potência de transmissão, aumentando a flexibilidade do sistema.

1.6 Estrutura do trabalho

Esse trabalho está dividido em outros 5 capítulos:

- O Capítulo 2 faz uma breve apresentação sobre os algoritmos de *Deep Learning* e *Reinforcement Learning*, explicando como ambas as abordagens são unidas para gerarem modelos mais robustos, originando a área de *Deep Reinforcement Learning*. O capítulo também apresenta a estrutura e o processo de aprendizagem de alguns dos principais algoritmos da área.
- O Capítulo 3 apresenta o problema de alocação de recursos em sistemas com comunicações D2D, expondo os desafios de se controlar a interferência na célula e maximizar a taxa de transmissão das comunicações que compartilham o mesmo espectro.
- O Capítulo 4 detalha o modelo proposto pelo trabalho, a modelagem do ambiente que possibilite o aprendizado e a estrutura adotada para solucionar o problema. Além disso, detalha o processo de aprendizagem do agente, destacando as técnicas presentes na literatura que ajudaram nesse processo. Por fim, especifica como se dá a utilização do algoritmo após o treinamento e como será feito o teste do desempenho do modelo.
- O Capítulo 5 agrupa os resultados obtidos com o modelo proposto tanto no processo de treinamento quanto em situações de teste. Os resultados de teste estão divididos em 2 tipos: testes gerais e testes específicos. Os testes gerais são aqueles em que o algoritmo é colocado sob diversas situações diferentes e, ao final, são extraídas as grandezas estatísticas resultantes da atuação do modelo. Já os testes específicos são situações controladas em que se deseja olhar detalhadamente a alocação feita pelo algoritmo e a sua evolução a medida que o ambiente se modifica.
- O Capítulo 6 conclui o trabalho, fazendo as últimas considerações sobre os resultados obtidos e sobre as contribuições oferecidas. Por fim, o capítulo também reúne sugestões de trabalhos futuros que possam enriquecer o tema e preencher lacunas encontradas ao longo da pesquisa realizada.

2 Conceitos de Inteligência Artificial

Este capítulo introduz as técnicas de AI que serão utilizadas ao longo deste trabalho, aplicadas em sistemas de comunicações digitais. Essas ferramentas estão divididas em dois campos de inteligência artificial: Aprendizado de Máquina/Aprendizado Profundo e Aprendizado por Reforço Profundo, que é uma subárea da primeira. Esse capítulo está destinado para apresentar a base teórica dessas áreas e as técnicas que serão utilizadas ao longo do trabalho.

2.1 Inteligência Artificial

Conceituando, Inteligência Artificial (*Artificial Inteligence* - AI) é o campo que busca fazer com que máquinas consigam desenvolver algum tipo de inteligência, ao passo que Aprendizado de Máquina (*Machine Learning* - ML) é uma subárea de AI que procura métodos que permitam que uma máquina aprenda, por si só, a executar diferentes tarefas a partir do recebimento de certos dados, sem a necessidade de definir como a execução será realizada. Já o Aprendizado Profundo (*Deep Learning* - DL) é uma subárea de ML que reúne tecnologias que permitem o desenvolvimento de máquinas com capacidade cognitiva complexa, enquanto o Aprendizado por Reforço (*Reinforcement Learning* - RL) é uma subárea de ML que tenta ensinar a máquina a tomar decisões a partir de recompensas, de maneira similar à teoria homônima do campo da psicologia [25].

Dentro de AI, existem duas formas principais de aprendizagem, o aprendizado supervisionado e o não supervisionado. O primeiro utiliza dados rotulados para ensinar um algoritmo a realizar uma tarefa, possibilitando que o algoritmo aprenda o padrão existente entre os dados de entrada e as saídas esperadas. Já o aprendizado não supervisionado é aplicado em problemas em que não há rotulação dos dados, de forma que o algoritmo aprende a realizar as tarefas por meio de alguma estratégia interna, buscando padrões nos dados de entrada que permitam a realização de alguma tarefa.

Nesse cenário, alguns autores classificam o aprendizado por reforço como um terceiro tipo de aprendizagem [15], já que os algoritmos de RL não utilizam exatamente nenhuma das duas estratégias citadas acima. Por outro lado, outros autores classificam o RL como

uma subárea do aprendizado supervisionado, denominando-o, inclusive, de aprendizado supervisionado atrasado [26].

Assim, o Aprendizado por Reforço Profundo (*Deep Reinforcement Learning* - DRL) é a interseção entre DL e RL, unindo conceitos de ambas as teorias, visando-se utilizar dos benefícios de cada uma para construir algoritmos mais robustos. A Figura 2.1 ilustra a associação entre cada uma dessas áreas e subáreas [26].

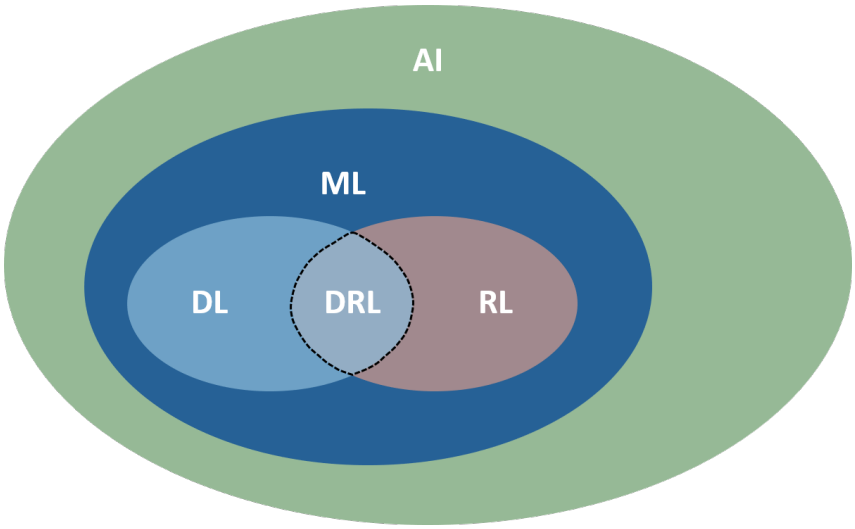


Figura 2.1: Ilustração da associação das áreas de AI, ML, DL, RL e DRL.

A união dessas áreas procura se utilizar da flexibilidade fornecida pelos algoritmos de RL e da capacidade de generalização que os algoritmos de DL apresentam.

A flexibilidade vem do fato de que não é necessário possuir dados rotulados para o treinamento de algoritmos de RL, é preciso, apenas, das recompensas coletadas após a execução do mesmo.

Já a capacidade de generalização se deve à robustez que os algoritmos de DL apresentam, combinando diversas camadas lineares e não lineares para modelar problemas com complexidade elevada [15].

Esse capítulo introduzirá os conceitos básicos da área de DL na seção 2.2 e os de RL na seção 2.3, unindo ambas as teorias na seção 2.4, que abordará algoritmos de DRL.

2.2 Deep Learning

Como visto, o Aprendizado Profundo é uma subárea de ML que, a partir da utilização de redes neurais com várias camadas ocultas e com um número elevado de neurônios, permite o desenvolvimento de algoritmos com maior capacidade de solucionar problemas com complexidade elevada [15].

Existem 3 tipos principais de redes neurais na área de DL, as chamadas Redes Neurais Profundas (*Deep Neural Networks* - DNN), as Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNN) e as Redes Neurais Recorrentes (*Recurrent Neural Networks* - RNN) [15].

2.2.1 Redes Neurais Profundas - DNNs

As DNNs são as redes mais básicas dessa área, com estruturas constituídas apenas por Camadas Totalmente Conectadas (*Fully Connected Layers*), geralmente apresentam mais de uma camada oculta para possibilitar o entendimento de problemas de complexidade alta.

Apesar de sua estrutura robusta, o desempenho dessas redes não é tão bom em problemas cujos vetores de entrada e/ou de saída possuem dimensionalidade alta, o que motivou a utilização de outras redes nesse tipo de problema.

2.2.2 Redes Neurais Convolucionais - CNNs

Nesse cenário, aparecem as CNNs, redes que utilizam filtros convolucionais parametrizados para extrair as principais informações presentes em entradas com dimensionalidade alta, como imagens. Por esse motivo, as CNNs possuem, geralmente, estruturas cujas primeiras camadas são convolucionais e as últimas são Camadas Totalmente Conectadas, reduzindo a dimensão dos dados de entrada nas primeiras camadas para passá-los pelas camadas ocultas.

2.2.3 Redes Neurais Recorrentes - RNNs

Por outro lado, as RNNs não foram criadas para solucionar a limitação relacionada a dimensionalidade que as DNNs apresentam, mas sim para melhorar o desempenho das redes neurais em problemas em que existe relação temporal entre os dados de entrada. Para isso, as RNNs são redes que possuem memória capaz de armazenar informações relacionadas às entradas recebidas no passado e utilizá-las para realizar tarefas no presente, o que favorece a sua utilização em problemas envolvendo séries temporais.

Entre as RNNs, destacam-se a *Gated Recurrent Unit* (GRU) e a *Long Short-Term Memory* (LSTM), redes com estruturas que permitem a armazenagem das informações na memória de maneira inteligente [15].

Essas são duas das principais estruturas da área de *Deep Learning*. Tais redes são, geralmente, treinadas por meio de aprendizado supervisionado. Entretanto, em alguns problemas não é possível utilizar esse tipo de aprendizado, seja pela falta de um conjunto

de dados rotulados, seja pela natureza do problema. É nesse contexto que se insere o aprendizado por reforço.

2.3 Reinforcement Learning

O Aprendizado por Reforço (*Reinforcement Learning* - RL) é uma área da Inteligência Artificial que se propõe a ensinar uma máquina a tomar decisões a partir das recompensas recebidas após a execução das ações realizadas [27].

Os algoritmos de RL não aprendem utilizando os mesmos processos usados com as técnicas de ML clássicas [26]. A motivação para essa mudança de estratégia é o fato de que alguns problemas possuem dados dificilmente rotuláveis, seja por ter um universo enorme de possíveis entradas e saídas, seja pela dificuldade do próprio projetista do algoritmo conhecer qual é a melhor decisão a ser tomada em cada situação possível. Por conta dessas diferenças presentes no processo de aprendizagem, existem outros conceitos que são fundamentais para a área de RL.

2.3.1 Conceitos básicos

Essa área vem se desenvolvendo há décadas, englobando algoritmos capazes de atuar em diferentes problemas, cada algoritmo contendo diferentes tecnologias associadas, de forma que não é possível resumi-las em apenas uma seção. Entretanto, existem conceitos chave que serão utilizados em praticamente todos os algoritmos desse campo e, portanto, são de fundamental importância. São eles: ambiente, estado, agente e recompensa [28].

O ambiente é a modelagem do problema que será utilizada para permitir que um algoritmo aprenda a tomar decisões baseadas no estado em que o ambiente se encontra [29]. O estado, denotado como s, é a situação do ambiente em um determinado instante em que se faz uma observação. O agente é um algoritmo inteligente capaz de compreender o estado em que o ambiente se encontra e, a partir disso, decidir qual ação tomar seguindo uma política μ desenvolvida. Por fim, a recompensa, denotada como r, é a variável responsável por avaliar o quão positiva ou negativa foi uma determinada ação executada [28].

A política μ pode ser de natureza estocástica ou determinística. No primeiro caso, a política é uma distribuição de probabilidades que, dado um estado, definirá a probabilidade de o agente escolher cada uma das ações possíveis. No segundo caso, a política define qual ação será executada de maneira determinística, não havendo nenhuma incerteza.

Apesar de a recompensa ser a variável que vai promover o aprendizado do agente, é preciso desenvolver um algoritmo capaz de tomar decisões que maximizem não a recompensa instantânea, mas sim a soma das recompensas recebidas em uma determinada

trajetória τ [29] vivenciada pelo agente. Esse somatório recebe o nome de retorno e pode ser matematicamente expresso como:

$$R(\tau) = \sum_{t=0}^{\tau} \gamma^t r_t \tag{2.1}$$

em que r_t é a recompensa obtida no evento t e γ é o fator de desconto, com valor entre 0 e 1, utilizado para priorizar as recompensas mais próximas do presente, em detrimento das recompensas que serão recebidas em um futuro mais distante.

Como mostrado na Equação 2.1, o retorno é calculado a partir de uma determinada trajetória τ, que é a sequência de estados e ações vivenciadas pelo agente, isto é, o conjunto finito de estados explorados pelo agente e as ações definidas em cada um desses estados. Conhecendo essa trajetória, é possível calcular a recompensa para cada etapa e, consequentemente, o retorno obtido ao longo da trajetória τ.

Alguns problemas possuem ambientes cuja evolução pode ser modelada completamente, isto é, ambientes em que uma função é capaz de predizer quais serão as trajetórias vivenciadas e quais serão as recompensas recebidas. Nesse tipo de problema, os algoritmos tentam aprender qual é esta função, visando utilizarem-na para planejarem quais ações serão tomadas, visando o maior retorno possível. Algoritmos que usam essa estratégia são chamados de *model-based*.

Entretanto, na maioria dos problemas, não é possível modelar o ambiente completamente por meio de uma função, seja por conta da complexidade, seja pela presença de fatores estocásticos. Nesses casos, o agente não pode se utilizar da mesma estratégia para definir a sua política de decisão. Nessas situações, os algoritmos precisam aprender por meio da experiência, executando e avaliando as ações tomadas, para melhorarem a sua política ao longo do tempo, buscando encontrar não uma função que modele o ambiente, mas sim uma maneira de decidir ações a partir das experiências vividas. Algoritmos que usam essa estratégia são chamados de *model-free* e serão o foco deste trabalho.

Para algoritmos dessa natureza, é necessário utilizar uma estratégia que forneça condições de ensinar uma rede neural a decidir qual ação levará o agente para trajetórias em que o retorno é máximo, levando em consideração as diferentes trajetórias possíveis. É nesse sentido que se inserem as funções valor e ação-valor.

2.3.2 Funções valor e ação-valor

Nesse cenário em que não é possível conhecer a trajetória que será vivenciada, seja por conta da aleatoriedade do ambiente ou por conta da imprevisibilidade das ações tomadas pelo agente, é necessário desenvolver algoritmos capazes de estimar a esperança de retorno a partir de um determinado s levando em conta todas as trajetórias possíveis. Para isso,

criou-se uma variável denominada função valor, cuja definição matemática é a média dos retornos para todas as trajetórias possíveis partindo do estado s [28]:

$$V_\mu(s) = \underset{\tau\sim\mu}{\mathrm{E}}[R_t|s_t = s] \tag{2.2}$$

em que $V_\mu(s)$ é a função valor do estado s, E é o símbolo que denota o cálculo da esperança, R_t é o retorno obtido até o evento t , s_t é o estado vivenciado no evento t e μ é a política utilizada pelo agente para tomada de decisões.

A partir da Equação 2.2, percebe-se que a função valor não leva em conta a ação tomada no instante t para o cálculo da esperança, apenas o estado presente. Assim, surge a grandeza denominada função ação-valor, que calcula a mesma esperança, mas levando em consideração o estado presente s e ação a executada nesse estado. Essa grandeza pode ser definida como [28]:

$$Q_\mu(s, a) = \underset{\tau\sim\mu}{\mathrm{E}}[R_t|s_t = s, a_t = a] \tag{2.3}$$

em que $Q_\mu(s, a)$ é a função ação-valor no estado s ao se executar a ação a, E é o símbolo que denota o cálculo da esperança, R_t é o retorno obtido até o evento t , s_t é o estado vivenciado no evento t, a_t é a ação realizada no evento t e μ é a política utilizada pelo agente para tomada de decisões

Essas duas grandezas se relacionam a partir da seguinte equação:

$$V_\mu(s) = \underset{a\sim\mu}{\mathrm{E}}[Q_\mu(s, a)] \tag{2.4}$$

Apesar de ser muito similar à função valor, a função ação-valor, conhecida também como função-Q, fornece uma possibilidade que a outra não oferece: o fato de levar em consideração a ação tomada faz com que, estando em um estado s, o agente consiga conhecer a esperança de retorno para cada ação possível.

A partir dessa variável, é possível desenvolver um algoritmo inteligente que, estando em um estado qualquer s, consiga escolher qual ação maximiza o retorno apenas conhecendo a função-Q de todas as ações possíveis. Esse é o princípio de funcionamento do *Q-learning*, um dos algoritmos mais populares de RL, que serviu de base para o surgimento de diversos outros [28].

2.3.3 Q-Learning

Esse algoritmo consiste em preencher uma tabela que agrupa os valores da função-Q para todas as combinações possíveis de estados e ações, possibilitando que o agente tome decisões embasadas nesses valores.

Inicialmente, o algoritmo não possui conhecimento do ambiente para preencher a tabela de maneira confiável, já que a sua política ainda não foi bem desenvolvida. Entretanto, o agente consegue melhorar cada uma das estimativas da tabela utilizando a equação de Bellman, uma das equações mais importantes dessa área [28].

$$Q(s, a) = E_{s' \sim P}[r + \gamma E_{a' \sim \mu} Q(s', a')] \tag{2.5}$$

em que s' é o estado futuro, a' é a ação realizada no estado s' e $E_{s' \sim P}$ é a esperança com relação as probabilidades P de o agente sair do estado s e assumir um estado s' qualquer e $E_{a' \sim \mu}$ é a esperança de a ação realizada em s' ser a' dada a política de definição das ações μ.

A Equação 2.5 formula que o valor da função-Q no par estado-ação atual é igual à recompensa recebida no momento atual somada ao valor da função-Q do par estado-ação futuro (s' e a') descontado pelo fator γ. A primeira esperança da equação se deve ao fato de que, em alguns ambientes, a transição de estados não é determinística, de forma que o estado futuro s' é definido por uma probabilidade P. Já a segunda esperança se deve ao fato de existir mais de uma ação futura a' possível, de maneira que o valor do estado-ação atual deve levar em consideração todas as possíveis combinações de s' e de a'.

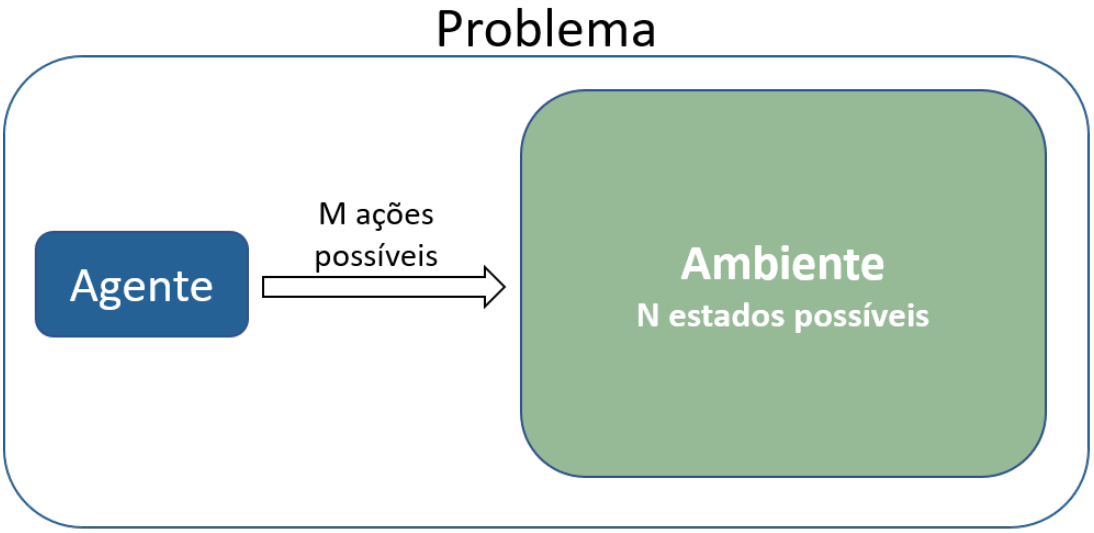
Usando essa equação, é possível melhorar as estimativas da tabela Q utilizando a técnica *Temporal-Difference Learning*, que consiste em um processo iterativo, que atualiza os valores da *Q-table* a partir das recompensas recebidas, de forma que a tabela esteja preenchida de maneira confiável após diversas experiências vivenciadas e, assim, o algoritmo consiga tomar suas decisões baseadas nela. Uma ilustração básica desse algoritmo pode ser vista na Figura 2.2.

Para problemas em que o ambiente é simples, ou seja, em que existem poucos estados possíveis para o agente explorar, é viável utilizar a *Q-table*, mas, quando a complexidade do ambiente é maior, isso se torna extremamente ineficiente e, em alguns casos, impraticável computacionalmente, já que o número de combinações de estados e ações aumenta rapidamente [28].

Nesses casos, é preciso recorrer a outras técnicas que otimizem esse processo. Uma forma de se fazer isso é utilizar técnicas estatísticas para estimar o valor da função-Q de estados e ações que não foram vivenciados a partir do cálculo dessa função em combinações (s, a) exploradas.

Entretanto, para problemas cuja complexidade é ainda maior, essas técnicas também se tornam pouco eficientes e surge a necessidade de encontrar uma maneira mais robusta de estimar a função-Q.

Assim, em 2013, o [30] propôs a utilização de uma rede neural para estimar a função-Q, a fim de usufruir da capacidade de generalização das redes profundas. O artigo desenvolveu



Q-table

	Estado 1	Estado 2	...	Estado N
Ação 1	$Q(s_1, a_1)$	$Q(s_2, a_1)$	...	$Q(s_N, a_1)$
Ação 2	$Q(s_1, a_2)$	$Q(s_2, a_2)$	...	$Q(s_N, a_2)$
...	...	...	...	...
Ação M	$Q(s_1, a_M)$	$Q(s_2, a_M)$	...	$Q(s_N, a_M)$

Figura 2.2: Ilustração de uma *Q-table* que reúne os valores da função-Q em cada par estado-ação.

um algoritmo capaz de atuar em ambientes de jogos clássicos de videogames antigos, cujo espaço de estados era enorme, se comparado com outros problemas clássicos de RL. Essa aplicação serviu como base para diversos outros artigos e algoritmos que foram desenvolvidos unindo conceitos de DL e RL [31, 32, 33, 34], originando a área denominada *Deep Reinforcement Learning*.

2.4 Deep Reinforcement Learning

Por estarem justamente na interseção entre os algoritmos de DL e RL, os algoritmos de DRL utilizam ambas as teorias para possibilitar que as máquinas aprendam: as redes neurais de DL fazem o papel do agente, enquanto a teoria de RL é utilizada para ensinar

essa rede neural. Dessa ideia básica, surgiram diversos algoritmos, cada um com suas particularidades, vantagens e desvantagens.

No geral, os algoritmos dessa área não são rígidos quanto ao tipo de rede neural que será utilizado, variando com a necessidade do problema. Assim, o tipo de rede neural que será utilizado é escolhido visando possibilitar o maior entendimento dos estados do ambiente e a melhor tomada de decisão das ações. Normalmente, se utilizam DNNs em problemas mais simples, CNNs em problemas com dimensionalidade alta e RNNs em problemas com relação temporal entre as entradas [15]. Também é possível combinar redes para usufruir das vantagens de cada uma delas, mas isso eleva a complexidade do algoritmo consideravelmente, o que se traduz em um maior tempo de treinamento.

Dessa forma, os algoritmos de DRL são caracterizados não pelas redes neurais utilizadas, mas sim pelo processo de treinamento. A seção abaixo introduzirá alguns dos principais algoritmos da área, expondo as suas diferenças.

2.4.1 Algoritmos clássicos de DRL

Um dos algoritmos pioneiros da área foi o *Deep Q-Network* (DQN), algoritmo de política determinística que utiliza uma rede neural para estimar a função-Q das diferentes ações possíveis e decide qual delas realizar a partir dessa estimativa. O seu processo de aprendizagem se baseia na Equação de Bellman (Equação 2.5), utilizando técnicas de *Temporal-Difference Learning* e *bootstrapping* para ensinar à rede neural a fazer estimativas cada vez mais próximas às estimativas calculadas com a equação de Bellman. É um algoritmo simples, cuja principal limitação é não ser aplicável em ambientes cujo espaço de ações é contínuo.

Nesses tipos de ambientes, o *Deep Deterministic Policy Gradient* (DDPG) se apresenta como alternativa, um algoritmo de política determinística com estrutura *Actor-Critic*, isto é, que apresenta separação entre a rede neural do agente e a do crítico. Essa característica permite que a saída da rede neural do agente seja de natureza contínua, solucionando a limitação que o DQN apresentava nesse quesito.

Paralelamente a esses algoritmos que usam política determinística, existem aqueles que utilizam política estocástica, como é o caso do *Advantage Actor-Critic* (A2C), do *Asynchronous Advantage Actor Critic* (A3C) e do *Soft Actor-Critic* (SAC).

Alguns desses algoritmos incorporaram melhorias em sua estrutura que originaram novos algoritmos, como é o caso do *Double DQN* (Double DQN), que utiliza duas redes neurais para estimar a função-Q, e do *Twin-Delayed DDPG* (TD3), que utiliza duas redes para o crítico, atrasa a atualização da política e suaviza a sincronização entre as redes envolvidas no processo de treinamento [24].

Nesse trabalho, optou-se por utilizar algoritmos que utilizam política determinística, mais especificamente o DDPG e o TD3, estruturas que são aplicáveis em ambientes cujas ações são de natureza contínua, sem a necessidade de discretizá-las. Por conta da similaridade existente entre o DDPG e o TD3, é possível ilustrar o funcionamento de ambos, mesmo que de maneira simplificada, por uma única ilustração, que está mostrada na Figura 2.3.

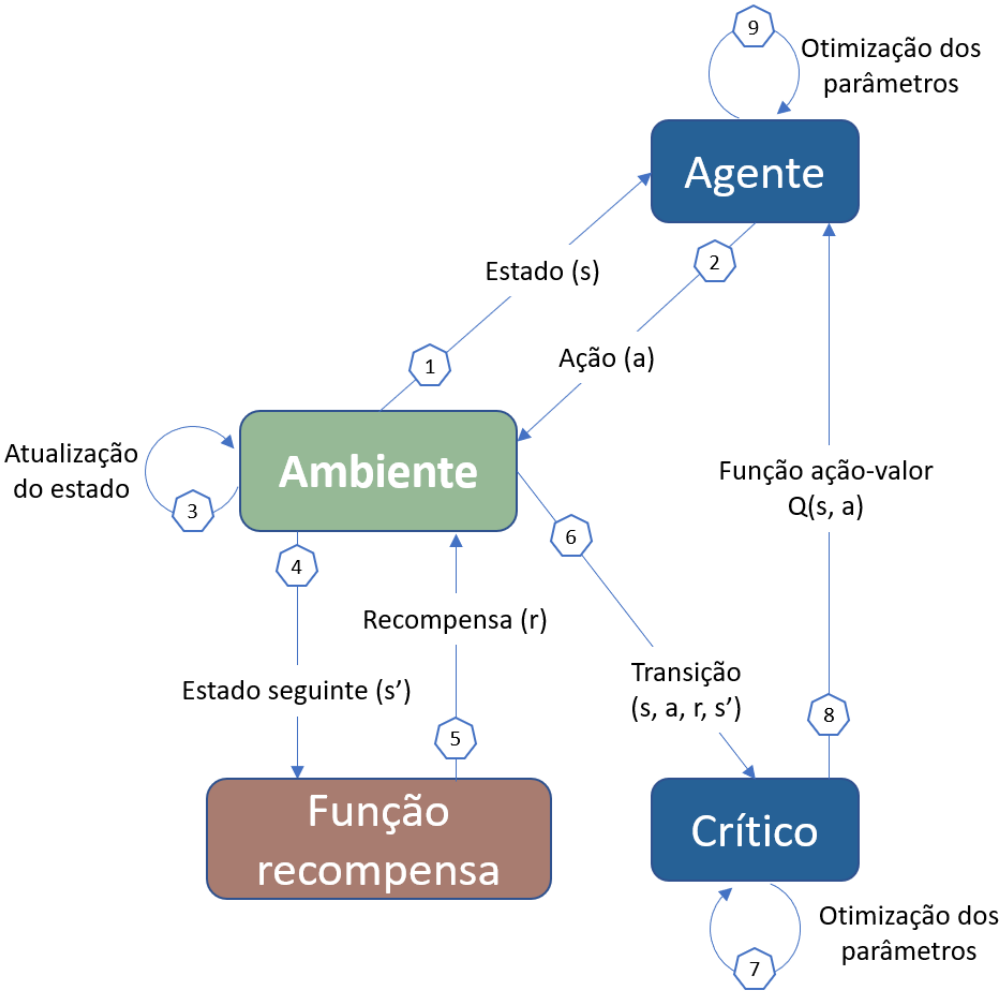


Figura 2.3: Representação do funcionamento do DDPG e do TD3.

As etapas presentes na Figura 2.3 estão detalhadas abaixo a partir dos seus índices:

1. Observação que o agente faz do estado s do ambiente;
2. Dado o estado s do ambiente, decisão de qual ação a será realizada;
3. Execução da ação a e atualização do ambiente para o estado s';

4. Cálculo da recompensa r utilizando a função de recompensa usando s' como entrada;
5. Coleta da recompensa r;
6. Envio da matriz de transição (s, a, r, s') para o crítico, para estimação do $Q(s, a)$ e $Q(s', \mu(s'))$, em que μ é a política do agente, definida na subseção 2.3.1;
7. Atualização dos parâmetros da rede neural do crítico a partir da equação de Bellman (Equação 2.5);
8. Envio da função-Q que servirá como avaliação da ação a indicada pelo agente;
9. Atualização dos parâmetros da rede neural do agente utilizando o valor de $Q(s, a)$.

O processo de aprendizagem dos algoritmos apresentado acima é suficiente para treinar a rede neural do agente mas, na prática, ambos os algoritmos são treinados, geralmente, utilizando outras técnicas que melhoram a convergência e a eficiência do processo.

2.4.2 Técnicas de aprimoramento do aprendizado

Além dos algoritmos, outras técnicas foram desenvolvidas para melhorar o aprendizado dos algoritmos dessa área, parte delas buscam tornar os algoritmos mais eficientes, enquanto outras tentam melhorar a convergência do processo de aprendizagem. Algumas dessas técnicas se tornaram mais populares e são utilizadas na maior parte dos algoritmos. Entre elas, destacam-se o *Experience Replay* [35, 30] e a utilização das chamadas *Target Networks* [30], ambas utilizadas nos algoritmos desse trabalho.

Além do aprimoramento da convergência, com a utilização desses algoritmos em ambientes cada vez mais complexos, também foi necessário ajustar alguns pontos do treinamento para que o algoritmo conseguisse compreender e explorar as possibilidades que o ambiente fornece, fugindo dos máximos locais. Nesse contexto situa-se o dilema Exploração-Explotação, que define a dificuldade que os algoritmos têm em decidir entre utilizar a política que já foi desenvolvida em busca de consolidar o aprendizado, processo denominado explotação, ou realizar ações diferentes que podem levá-lo para situações potencialmente vantajosas, que seria a exploração do ambiente.

Ajustar as estratégias de exploração é fundamental para um bom aprendizado do agente e, por isso, diversas técnicas já foram desenvolvidas para esse fim, entre as mais clássicas estão a utilização do ϵ-*greedy* para problemas com ações discretas [28] e do *Ornstein–Uhlenbeck noise* para problemas com espaço de ações contínuo [36].

O ϵ-*greedy* consiste em definir uma probabilidade ϵ de que a ação realizada não seja aquela definida pelo agente, e sim uma outra ação decidida aleatoriamente entre as ações possíveis[28]. Essa probabilidade ϵ é ajustada ao longo do processo de aprendizagem,

garantindo que, nos primeiros episódios, o processo de exploração seja preponderante, enquanto nos últimos episódios, o processo explotação seja priorizado [28].

Já a adição de ruído *Ornstein–Uhlenbeck* em ações de natureza contínua é uma estratégia em que o agente define a ação que será realizada e, ao final do processo de decisão, adiciona-se uma variável aleatória aos valores definidos pelo agente, conferindo um fator estocástico à ação realizada, o que promove a exploração do espaço contínuo de ações [36].

Nesse trabalho, utilizou-se a estratégia de adição de ruido nos parâmetros da rede neural do agente, proposto em [37]. Essa técnica consiste em somar uma variável com distribuição gaussiana aos pesos da rede neural, inserindo aleatoriedade no processo de escolha de ações e promovendo a exploração do ambiente.

Diferente do que é feito ao se adicionar um ruído *Ornstein–Uhlenbeck* na saída da rede neural, isto é, adicionar aleatoriedade após a definição da ação, o mecanismo de exploração utilizado adiciona aleatoriedade durante o processo de decisão da ação, já que os parâmetros da rede são modificados por uma variável estocástica, o que faz com que o processo de tomada de decisão do agente seja parcialmente aleatório, garantindo a exploração do ambiente. A Figura 2.4 apresenta a diferença entre a inserção de ruído na ação ou nos parâmetros da rede.

Esse ruído é adaptativo, de forma que a distribuição dessa variável é modificada ao longo do processo de treinamento obedecendo a um critério pré-determinado. O critério utilizado nesse trabalho foi a diferença entre as ações determinadas com e sem a adição do ruído, de maneira que, se essa diferença for maior do que um determinado limiar (δ), o desvio padrão do ruído (σ) é acrescido, caso contrário, o desvio padrão é decrescido [37]. O pseudocódigo dessa técnica está exposto no Algoritmo 1.

A partir da exposição dessas técnicas de aprimoramento do aprendizado, é possível detalhar o processo de aprendizagem de cada um desses algoritmos utilizando algumas delas. Os pseudocódigos do processo de treinamento do DDPG e do TD3 estão expostos no Algoritmo 2 e no Algoritmo 3, respectivamente.

2.4.3 Outros algoritmos

Além dos algoritmos já citados, existem outras estruturas de DRL mais recentes que possuem outras características que visam melhorar o desempenho do agente e conferir mais confiabilidade à convergência do processo de aprendizagem. Entre esses algoritmos, destaca-se o *Distributed Distributional Deep Deterministic Policy Gradient* (D4PG) e o *Soft Actor-Critic* (SAC).

O D4PG é uma extensão do DDPG, com a diferença de que o crítico avalia as decisões do agente por meio de uma ótica distribucional, levando em consideração não apenas esperança do retorno, ou função ação-valor $Q(s, a)$, mas sim a distribuição dos possíveis

Algorithm 1 Parameter Noise

Input: parâmetros iniciais da rede neural do agente θ

Output: parâmetros otimizados da rede neural do agente θ

Inicialização da rede neural do agente do algoritmo μ_θ
Inicialização da rede neural de exploração ($\mu_{\tilde{\theta}}$)
Inicialização da rede neural de adaptação do ruído ($\mu_{\overline{\theta}}$)
Sincronização dos pesos das redes: $\tilde{\theta} \leftarrow \theta$ e $\overline{\theta} \leftarrow \theta$
Inicialização do desvio padrão do ruído de exploração σ
Inicialização dos parâmetros de adaptabilidade da estratégia α, $delta$, T_{adapt} e $d(\cdot, \cdot)$
for $episodio = 1, 2, \ldots n\_episodios$ **do**
    Pertubação $\tilde{\theta} \leftarrow \theta + \mathcal{N}(0, \sigma)$ e obtenção de $\mu_{\tilde{\theta}}$
    for $evento = 1, 2, \ldots, n\_eventos$ **do**
        Escolha da ação a ser executada usando $\mu_{\tilde{\theta}}$
        Execução da ação, coleta da transição e execução do processo de aprendizagem
        if $evento$ mod $T_{adapt} = 0$ **then**
            Perturbação $\overline{\theta} \leftarrow \theta + \mathcal{N}(0, \sigma)$ e obtenção de $\mu_{\overline{\theta}}$
            Cálculo da distância $dist = E[d(\mu_\theta(s), \mu_{\overline{\theta}}(s))]$
            Adaptação de σ através de

$$\sigma \leftarrow \begin{cases} \alpha\sigma, & dist \leq \delta \\ \frac{1}{\alpha}\sigma, & cc \end{cases}$$

        end if
    end for
end for

Algorithm 2 DDPG

Input: parâmetros iniciais da rede neural do agente θ

Output: parâmetros otimizados da rede neural do agente θ

1: Inicialização das redes neurais do agente $\mu(s)$ e do crítico $Q(s, a)$ com pesos θ_μ e θ_Q, respectivamente.
2: Inicialização das redes neurais *target* μ' e Q' com pesos θ'_μ e θ'_Q.
3: Sincronização dos pesos das redes *target*: $\theta'_\mu \leftarrow \theta_\mu$ e $\theta'_Q \leftarrow \theta_Q$.
4: Inicialização da memória de *replay* R para utilização do *Experience Replay*.
5: **for** $episodio = 1, 2, \ldots n\_episodios$ **do**
6: Inicialização da estratégia de exploração η utilizada.
7: Observação do estado inicial s_0.
8: **for** $evento = 1, 2, \ldots, n\_eventos$ **do**
9: Seleção da ação $\mu(s_{evento})$ utilizando a estratégia de exploração η.
10: Execução da ação a_{evento} e recebimento da recompensa r_{evento}.
11: Mudança do ambiente para o estado $s_{evento+1}$ provocada pela ação a_{evento}.
12: Armazenagem da transição (s_{evento}, a_{evento}, r_{evento}, $s_{evento+1}$) na memória R.
13: Amostragem aleatória de um *minibatch* com N transições (s_i,a_i,r_i,s_{i+1}) de R.
14: Cálculo da Equação de Bellman: $h_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}))$.
15: Atualização dos parâmetros do crítico θ_Q minimizando o erro:

$$Loss_Q = \frac{1}{N} \sum_i [h_i - Q(s_i, \mu(s_i))]^2$$

16: Atualização dos parâmetros do agente θ_μ minimizando o erro:

$$Loss_\mu = \frac{1}{N} \sum_i -Q'_i$$

17: Atualização suavizada dos parâmetros das redes *target*:

$$\theta'_\mu \leftarrow \tau\theta_\mu + (1 - \tau)\theta'_\mu$$

$$\theta'_Q \leftarrow \tau\theta_Q + (1 - \tau)\theta'_Q$$

18: **end for**
19: **end for**

Algorithm 3 TD3

Input: parâmetros iniciais da rede neural do agente θ

Output: parâmetros otimizados da rede neural do agente θ

Inicialização das redes neurais do agente $\mu(s)$ e dos críticos $Q_1(s,a)$ e $Q_2(s,a)$ com pesos θ_μ, θ_{Q_1} e θ_{Q_2}, respectivamente.
Inicialização das redes neurais *target* μ', Q'_1 e Q'_2 com pesos θ'_μ, θ_{Q1}' e θ_{Q2}'.
Sincronização dos pesos das redes *target*: $\theta'_\mu \leftarrow \theta_\mu$, $\theta'_{Q_1} \leftarrow \theta_{Q_1}$ e $\theta'_{Q_2} \leftarrow \theta_{Q_2}$.
Inicialização da memória de *replay* R para utilização do *Experience Replay*.
for $episodio = 1, 2, \ldots n\_episodios$ **do**
    Inicialização da estratégia de exploração η utilizada.
    Observação do estado inicial s_0.
    for $evento = 1, 2, \ldots, n\_eventos$ **do**
        Seleção da ação $\mu(s_{evento})$ utilizando a estratégia de exploração η.
        Execução da ação a_{evento} e recebimento da recompensa r_{evento}.
        Mudança do ambiente para o estado $s_{evento+1}$ provocada pela ação a_{evento}.
        Armazenagem da transição (s_{evento}, a_{evento}, r_{evento}, $s_{evento+1}$) na memória R.
        Amostragem aleatória de um *minibatch* com N transições (s_i,a_i,r_i,s_{i+1}) de R.
        Definição da ação *target* através de: $a' \leftarrow \mu'(s_i) + clip(\mathcal{N}(0, \sigma_\mu), -c, c)$.
        Cálculo da Equação de Bellman: $h_i = r_i + \gamma \min_{i=1,2} Q'_i(s_{i+1}, a')$.
        Atualização dos parâmetros do crítico θ_Q minimizando o erro:

$$Loss_Q = \frac{1}{N} \sum_i [h_i - Q(s_i, \mu(s_i))]^2$$

        if $evento$ mod T_{update} **then**
            Atualização dos parâmetros do agente θ_μ minimizando o erro:

$$Loss_\mu = \frac{1}{N} \sum_i -Q'_i$$

            Atualização suavizada dos parâmetros das redes *target*:

$$\theta'_\mu \leftarrow \tau\theta_\mu + (1-\tau)\theta'_\mu$$

$$\theta'_Q \leftarrow \tau\theta_Q + (1-\tau)\theta'_Q$$

        end if
    end for
end for

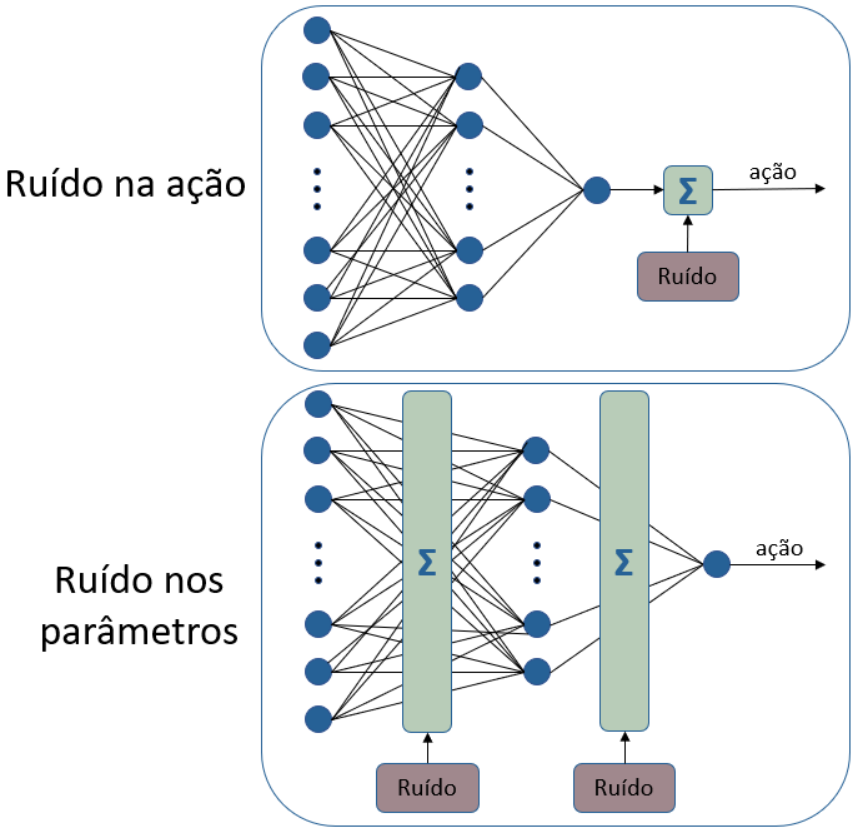


Figura 2.4: Ilustração da diferença entre as técnicas de exploração que inserem ruído na ação e a técnica utilizada, que insere ruído nos parâmetros da rede neural do agente.

retornos, representada por $Z(s, a)$. Essa adaptação permite que o agente aprenda a realizar ações mais confiáveis, levando em conta características como a variância e o desvio padrão dos possíveis retornos que uma ação gerará e não somente a esperança deles.

Por outro lado, o SAC é um algoritmo muito similar ao TD3, mas possui política estocástica. Essa característica permite ao SAC levar em consideração a entropia relacionada às ações possíveis, o que permite que o agente aprenda quais são as incertezas relacionadas a cada uma das ações possíveis, o que torna mais robusto o processo de tomada de decisão.

Tanto o SAC quanto o D4PG não foram implementados nesse trabalho por insuficiência de tempo, mas as suas utilizações constam como sugestões para trabalhos futuros, dispostas na Seção 6.1.

2.5 Conclusão

A área de DRL vêm crescendo nos últimos anos e isso vem motivando a criação de novos algoritmos, cada um com suas especifidades, vantagens e desvantagens. Neste trabalho, optou-se por utilizar algoritmos com política determinística que conseguissem definir ações de natureza contínua, já que o problema apresenta essa necessidade.

Assim, utilizar-se-ão dois modelos, um baseado no DDPG e outro no TD3 para que seja possível comparar dois algoritmos da área executando a alocação de recursos do sistema, considerando a inserção de comunicações D2D, cujos detalhes e desafios são discutidos no Capítulo 3.

3 Alocação de recursos em sistemas com comunicações D2D

Este capítulo introduz a comunicação *Device-to-Device* (D2D), quais são as aplicações esperadas para essas comunicações no 5G, além de apresentar as dificuldades relacionadas à sua inserção no sistema móvel.

3.1 Definição da comunicação D2D

Diferente das comunicações tradicionais do sistema móvel, a comunicação D2D permite que dois dispositivos se comuniquem sem intermédio da Estação Rádio-Base (ERB). O objetivo dessas comunicações é diminuir os efeitos do *path-loss* nas comunicações em que transmissor e receptor estejam próximos, possibilitando que os dispositivos se comuniquem utilizando potências mais baixas. A Figura 3.1 ilustra uma célula do sistema móvel com comunicações tradicionais e comunicações D2D.

É esperado que este tipo de comunicação seja capaz de promover um aumento da eficiência espectral do sistema móvel através do compartilhamento do espectro, do aumento de cobertura e da diminuição de perdas no canal de comunicação. Visando aproveitar esses potencias, as *releases* do 3GPP classificaram essas comunicações em termos de suas aplicações, assistências, cenários e casos de uso esperados para a o D2D no 5G [6].

3.1.1 Classificação

É possível dividir as comunicações D2D em *in-band* e *out-band*, classificação que leva em consideração o reúso do espectro [6].

Comunicações D2D *in-band* são aquelas alocadas no espectro licenciado e, portanto, reúsam o espectro destinado para as comunicações tradicionais do sistema móvel. Comunicações *in-band* podem ser classificadas como *underlay*, se operarem em todo o espectro licenciado, ou como *overlay*, se operarem em pequenas frações do espectro que os usuários do sistema móvel não utilizam [6].

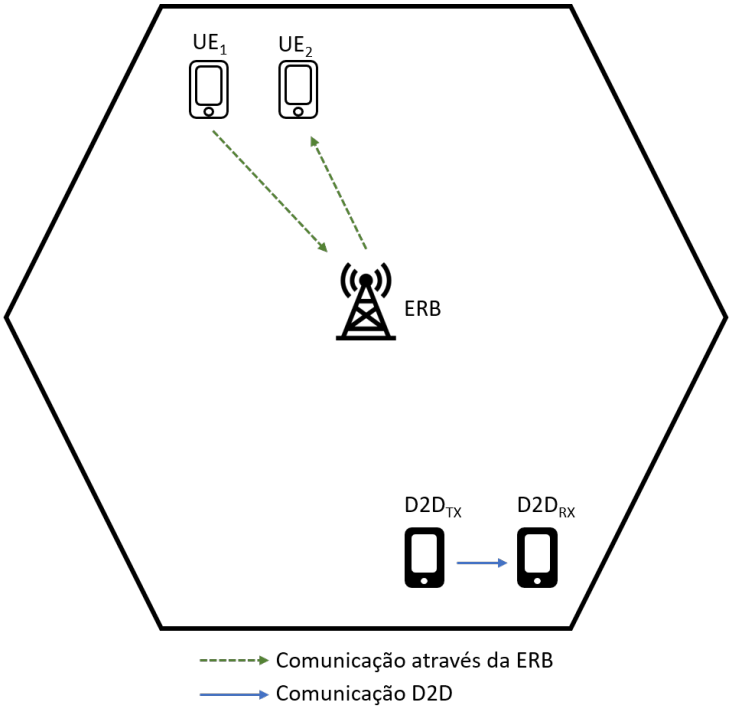


Figura 3.1: Ilustração de comunicações entre dispositivos realizadas com o intermédio da ERB (linha tracejada em verde) e sem o intermédio da ERB (linha contínua em azul).

Já as comunicações D2D *out-band* são aquelas que operam dentro do espectro não licenciado e, por isso, apresentam dificuldade de controle e gerenciamento da interferência [6].

O foco desse trabalho serão as comunicações *in-band*, mais especificamente, do tipo *underlay* [6].

3.1.2 Assistências

As comunicações D2D podem ser consideradas *network-assisted* se a ERB é a responsável por identificar dispositivos aptos para a comunicação D2D e permitir o estabelecimento da mesma. Por outro lado, elas podem ser consideradas *network-independent* se os dispositivos possuem autonomia para encontrar outros dispositivos para o estabelecimento de uma comunicação D2D sem a necessidade da ERB [6].

3.1.3 Aplicações

As comunicações D2D foram motivadas por duas principais aplicações, os serviços de proximidade (*proximity services* - ProSe) e as comunicações em grupo [6].

O ProSe permite que dispositivos fisicamente próximos se descubram e se comuniquem, diminuindo o *path-loss* e a interferência que comunicações clássicas causariam.

Já as comunicações em grupo são aquelas em que há a difusão de dados de um dispositivo para muitos, que tradicionalmente é feita a partir de diferentes comunicações de *downlink* que recebem o mesmo sinal dentro de uma célula. Isso pode ser otimizado a partir da utilização de comunicações D2D no sistema de comunicações.

Esse trabalho focará nas comunicações D2D aplicadas ao ProSe.

3.1.4 Cenários

Os cenários previstos de aplicação das comunicações D2D no 5G pela 3GPP são três [6]: *in-coverage*, que ocorre quando ambos os dispositivos da comunicação D2D estão dentro do raio de cobertura da ERB; *out-of-coverage*, que ocorre quando ambos os dispositivos da comunicação D2D estão fora do raio de cobertura da ERB; *partial-coverage*, que ocorre quando um dos dispositivos da comunicação D2D está dentro do raio de cobertura da ERB, enquanto o outro está fora.

A alocação de comunicações D2D *in-coverage* serão o foco desse trabalho.

3.1.5 Casos de uso

O 3GPP também previu diversos casos de uso das comunicações D2D. Entre os principais estão [6]:

- Comunicação Cooperativa Multiusuário (*Multiuser cooperative communication* - MUCC): situação em que um dispositivo com intensidade de sinal fraca utiliza outro dispositivo com intensidade de sinal forte para melhorar a sua comunicação;
- *D2D offloading*: situação em que uma comunicação D2D é utilizada para ajudar uma comunicação que exige altas taxas de transmissão, descongestionando-a;
- Comunicação veicular e marítima: situação em que duas embarcações ou dois veículos estão próximos e precisam se comunicar entre si;
- Segurança na transmissão de dados: situação em que se deseja utilizar uma comunicação D2D pra armazenar os dados transmitidos em um dispositivo seguro, evitando o armazenamento em nuvem, que pode ser mais vulnerável a invasões.

A partir do detalhamento previsto nas *releases* do 3GPP, é preciso definir a comunicação D2D tecnicamente, explorando os efeitos que ela produz no sistema de comunicação.

3.2 Objetivos e desafios

Como representação do problema, será utilizada uma modelagem para o sistema de comunicações móveis formada por uma única célula com M Equipamentos do Usuário (*User Equipment* - UE) se comunicando com a ERB no *uplink* e K comunicações D2D *in-band* (*underlay*) aplicadas ao ProSe e com cenário *in-coverage*.

Essas comunicações utilizarão um canal OFDM com N *resource blocks* (RB) disponíveis para comunicação, em que $N = M$. Cada UE utiliza apenas um RB para comunicação, enquanto um par D2D pode utilizar mais de um RB, simultaneamente, para realizar a sua comunicação.

3.2.1 Interferência entre comunicações

A alocação de duas comunicações (D2D ou não) em um mesmo RB causa interferência mútua entre elas, como mostrado na Figura 3.2.

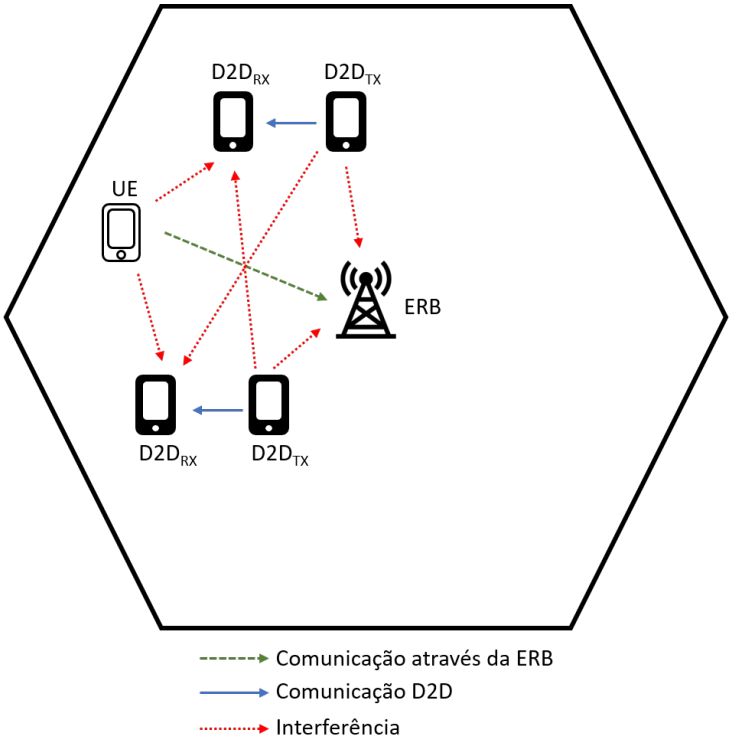


Figura 3.2: Ilustração das comunicações de um RB e das interferências causadas entre si.

Essa interferência pode ser mensurada a partir da Relação Sinal Ruído mais Interferência (*Signal-to-Noise plus Interference Ratio* - SNIR) do canal de comunicação, definida matematicamente a partir da Equação 3.1.

$$SNIR = \frac{P_S}{P_I + P_N} \tag{3.1}$$

em que P_S é a potência do sinal recebido pelo receptor da comunicação, P_I é a potência interferente recebida e P_N é a potência do ruído do canal.

Em um sistema OFDM, a potência interferente P_I é a soma das potências das demais comunicações do mesmo RB que chegam até o receptor da comunicação em questão. Assim, aumentar a potência de uma comunicação, mantendo todas as demais potências constantes, promove o aumento da SNIR desta comunicação, mas acarreta a diminuição das demais SNIRs, já que a interferência que ela causará nas demais também aumentará.

3.2.2 Capacidade de transmissão e Eficiência Espectral

A partir da SNIR é possível estimar a capacidade de transmissão do canal utilizando o Teorema de Shannon–Hartley, definido na Equação 3.2.

$$C_n = B_n \log_2(1 + SNIR) \tag{3.2}$$

em que C_n é a capacidade do canal em bits/s, B_n é a largura de banda do canal em Hz e a SNIR é a Razão Sinal Ruído mais Interferência do canal em valor absoluto.

A capacidade calculada a partir da Equação 3.2 é utilizada para estimar a taxa de transmissão que cada comunicação é capaz de usufruir em um RB. Para o caso de comunicações D2D, que utilizam mais de um RB para comunicação, a capacidade total é calculada usando a Equação 3.3.

$$C_c = \sum_{n=1}^{N} C_n \tag{3.3}$$

em que C_c é a capacidade de transmissão total da comunicação em bits/s e C_n é a capacidade de transmissão da comunicação no n-ésimo RB.

A capacidade de transmissão do sistema é o somatório da capacidade de todas as comunicações envolvidas, assim:

$$C_T = \sum_{c=1}^{M+K} C_c \tag{3.4}$$

em que C_T é a capacidade de transmissão total do sistema em bits/s.

A eficiência espectral do sistema é a variável que mensura o aproveitamento do espectro em uso. Por esse motivo, essa grandeza é função da capacidade do sistema, como mostrado na Equação 3.5:

$$\epsilon = \frac{C_T}{B_T} \tag{3.5}$$

em que ϵ é a eficiência espectral em bits/s/Hz e B_T é a banda total utilizada pelo sistema em Hz.

3.2.3 *Outage*

Em sistemas de comunicações reais, cada aplicação necessita de taxas de transmissão específicas, que garantem que o serviço seja entregue com qualidade ao usuário. Tal grandeza é chamada de taxa de transmissão mínima do serviço em questão.

Para o 5G, é previsto que a comunicação da maneira tradicional entre UE e ERB seja a comunicação primária do sistema, com a comunicação D2D *in-band* sendo realizada em caráter secundário.

Assim, a alocação de recursos deve sempre priorizar a comunicação primária, evitando ao máximo que a sua taxa de transmissão seja inferior à taxa de transmissão mínima do serviço que está sendo requisitado pelo usuário, situação que, neste trabalho, será denominada *outage*.

Apesar dessa priorização, em alguns casos é possível efetuar uma comunicação D2D sem provocar interferência suficiente para violar a taxa de transmissão mínima da comunicação prioritária. Isso faz com que a eficiência espectral da célula aumente, sem aumentar a taxa de *outage* das comunicações prioritárias.

Casos como esse acontecem geralmente quando os dispositivos em comunicação D2D estão distantes do dispositivo receptor da comunicação prioritária e/ou quando os dispositivos da comunicação prioritária estão próximos o suficiente da ERB para que a potência interferente da comunicação D2D seja pequena em relação à potência recebida, mantendo a SNIR da comunicação em um nível seguro.

Entretanto, não é apenas a distância que determina o grau de interferência que uma comunicação exerce sobre a outra, já que existem fatores como o *shadowing* e o desvanecimento de pequena escala que provocam efeitos variados, dependentes das características do ambiente e de outros eventos que necessitam de uma modelagem probabilística.

Assim, uma boa estratégia de alocação de recursos precisa levar em consideração as características do ambiente e a situação em que os dispositivos se encontram para buscar maximizar a eficiência espectral do sistema, ao mesmo tempo que protege as comunicações prioritárias do *outage*, dois objetivos conflitantes devido à interferência que uma comunicação produz nas outras.

3.3 Conclusão

A partir do detalhamento das aplicações esperadas para as comunicações D2D e do modo que elas ocorrem, definiu-se os objetivos que guiarão o alocador de recursos do sistema, que é a garantia daQoS das comunicações prioritárias e a maximização das taxas de transmissão das comunicações D2D.

Partindo dessas premissas, os algoritmos para alocação serão desenvolvidos a partir das configurações expostas no Capítulo 4, que também detalha o modelo utilizado para o sistema de comunicações e o processo de treinamento e teste do alocador.

4 Modelo Proposto

O modelo proposto neste trabalho para alocação de recursos é um algoritmo que utiliza estratégias de *Deep Reinforcement Learning* (DRL) apresentadas no Capítulo 2, buscando maximizar a eficiência espectral do sistema de comunicações, sem aumentar a taxa de *outage* das comunicações prioritárias do sistema. Esse capítulo detalha as características do sistema em que o algoritmo fará a alocação de recursos, a arquitetura do modelo proposto, além do processo de treinamento e utilização do modelo.

4.1 Modelagem do sistema de comunicações

A modelagem do sistema de comunicações foi introduzida na Seção 3.2, se tratando de uma única célula com M comunicações prioritárias entre UE e a ERB no *uplink* e K comunicações D2D *in-band* (*underlay*) utilizando N *resource blocks* (RB) de um sistema OFDM.

Nesse sistema, as comunicações D2D podem distribuir a sua potência de transmissão em mais de um RB, simultaneamente, enquanto as comunicações prioritárias utilizam um único RB para se comunicarem.

4.1.1 Características da célula

O cenário utilizado para alocação de recursos foi uma célula de formato quadrado com lados de 1 km, com a ERB localizada no centro dela. Tanto os dispositivos em comunicação primária quanto em comunicação secundária possuem liberdade para se movimentarem para qualquer posição da célula, com a limitação de que o transmissor e o receptor de uma mesma comunicação D2D não podem estar a mais de 20 metros um do outro.

Os dispositivos se locomovem em trajetórias retilíneas e, se algum alcançar a borda da célula, a sua trajetória é refletida de forma que os ângulos de reflexão e de incidência possuam o mesmo valor, seguindo a Primeira Lei da Reflexão. A Tabela 4.1 reúne as demais configurações de modelagem da célula.

Tabela 4.1: Configurações de modelagem da célula de comunicação.

Parâmetro	**Valor**
Formato da célula	quadrado
Dimensão da célula	1000 m
Número de UEs ativos	M
Número de pares D2D ativos	K
Distância máxima entre pares D2D	20 m
Altura da ERB	25 m
Altura dos dispositivos	1.5 m

4.1.2 Modelo de canal

A modelagem do canal se tratou da simulação de um ambiente urbano, com *path-loss* e desvanecimento de larga escala (*shadowing*). A perda de caminho foi modelada de acordo com os modelos micro urbanos do relatório da ITU-R [38], em que foram utilizadas diferentes equações para definir a perda entre UE e ERB, entre dispositivos D2D e entre diferentes tipos de comunicações [39]:

- *Path loss* entre UE e ERB (L_{UE-ERB}):

$$L_{UE-ERB} = 40.9 + 36.7\log_{10}(d) + 26\log_{10}(\frac{f_c}{5})$$

- *Path loss* entre D2D Tx e D2D Rx ($L_{D2D-D2D}$):

$$L_{D2D-D2D} = 49 + 40\log_{10}(\frac{d}{1000}) + 30\log_{10}(\frac{f_c}{1000})$$

- *Path loss* entre diferentes tipos de comunicações (L_{interf}):

$$L_{interf} = 22.7 + 36.7\log_{10}(d) + 30\log_{10}(f_c)$$

em que d é a distância entre os dispositivos em metros e f_c é a frequência da portadora em GHz.

O desvanecimento de larga escala foi modelado a partir de uma distribuição log-normal com média 0 e desvio padrão 8 dB, adotando distância de descorrelação espacial igual a 50 m [40].

Optou-se por não modelar o desvanecimento de pequena escala para simplificação da modelagem.

Tabela 4.2: Configurações de modelagem do canal de comunicação.

Parâmetro	**Valor**
Frequência da portadora (f_c)	2 GHz
Número de *resource blocks*	N
Banda ocupada por cada RB	180 kHz
Potência de transmissão mínima dos dispositivos	-20 dBm
Potência de transmissão máxima dos dispositivos	25 dBm
SNIR mínima das comunicações prioritárias	10 dB
Desvio padrão do *shadowing*	8 dB
Distância de descorrelação espacial do *shadowing*	50 m
Ganho da antena da ERB	17.8 dBi
Ganho da antena dos dispositivos	0 dBi
Potência do ruído térmico (P_N)	- 151 dBW

A potência do ruído térmico foi calculada considerando a temperatura equivalente no receptor igual a 290 K e a banda do RB igual a 180 kHz. As demais configurações de modelagem do canal de comunicação utilizado estão expostas na Tabela 4.2.

4.2 Estrutura do algoritmo desenvolvido para alocação de recursos

O algoritmo proposto para realizar a alocação de recursos do sistema de comunicações detalhado na Seção 4.1 utiliza uma atuação centralizada, isto é, ele recebe um conjunto de informações sobre os dispositivos comunicantes presentes na célula e decide qual será a potência que cada dispositivo transmissor vai utilizar e como a potência será distribuída ao longo dos RBs, no caso de comunicações D2D.

Dessa maneira, é necessário definir quais serão as entradas e saídas do algoritmo, além de definir outras configurações importantes para o aprendizado do modelo.

4.2.1 Entradas e saídas

Como o algoritmo utilizará técnicas de DRL para possibilitar o seu processo de aprendizagem, é preciso definir quais informações do sistema formarão o estado do ambiente, quais serão as ações realizadas e como será feito o cálculo da recompensa.

Estado

O estado do ambiente utilizado foi um vetor com a concatenação das coordenadas espaciais dos T dispositivos comunicantes e a capacidade de transmissão total de cada uma das comunicações da célula, calculada conforme Equação 3.4. Essa representação do estado permite que o algoritmo utilize a distância entre os dispositivos e as características do ambiente que cada um desses dispositivos está vivenciando, para definir a melhor maneira de alocar os recursos. Assim, o vetor resultante dessa concatenação será a entrada que a rede neural do agente utilizará para decidir as ações que serão realizadas.

Ação

A partir do estado, o algoritmo deve decidir quais serão as potências de transmissão de cada comunicação da célula e como essas potências serão distribuídas através dos RBs, para as comunicações D2D.

Assim, a saída da rede neural do agente será um vetor unidimensional com a concatenação das potências de transmissão das comunicações prioritárias e das potências das comunicações D2D distribuídas ao longo dos RBs. A Figura 4.1 mostra a transformação do vetor unidimensional de dimensão $M + K \cdot N$ para um vetor bidimensional, em que as linhas representam as comunicações do sistema e as colunas representam os RBs.
em que x_i e y_i são as coordenadas espaciais do i-ésimo dispositivo, C_j é a capacidade da j-ésima comunicação, P_m é a potência da m-ésima comunicação prioritária e $P_{k,n}$ é a potência da k-ésima comunicação D2D no n-ésimo RB.

É importante ressaltar que os valores que saem da rede neural já estão em dBm e, por isso, os RBs não utilizados por um UE são preenchidos com $-\infty$ que, em valor absoluto, equivale a 0 W.

Assim, o algoritmo recebe o estado em que o ambiente se encontra e define as potências que serão utilizadas pelas comunicações até que ocorra uma nova alocação. O intervalo que o algoritmo adota entre uma alocação e outra é de 0,1 segundo, de forma que durante esse intervalo, a alocação válida será a última realizada.

Após a alocação das potências indicada pelo agente, é preciso definir uma maneira de avaliar se os seus efeitos resultantes foram positivos ou negativos, para isso servem as recompensas.

Recompensa

Após a definição das potências das comunicações, o valor de SNIR de cada comunicação é modificado, em razão da mudança de posição dos dispositivos e das mudanças provocadas pela própria alocação de potências. Assim, para determinar a recompensa recebida em

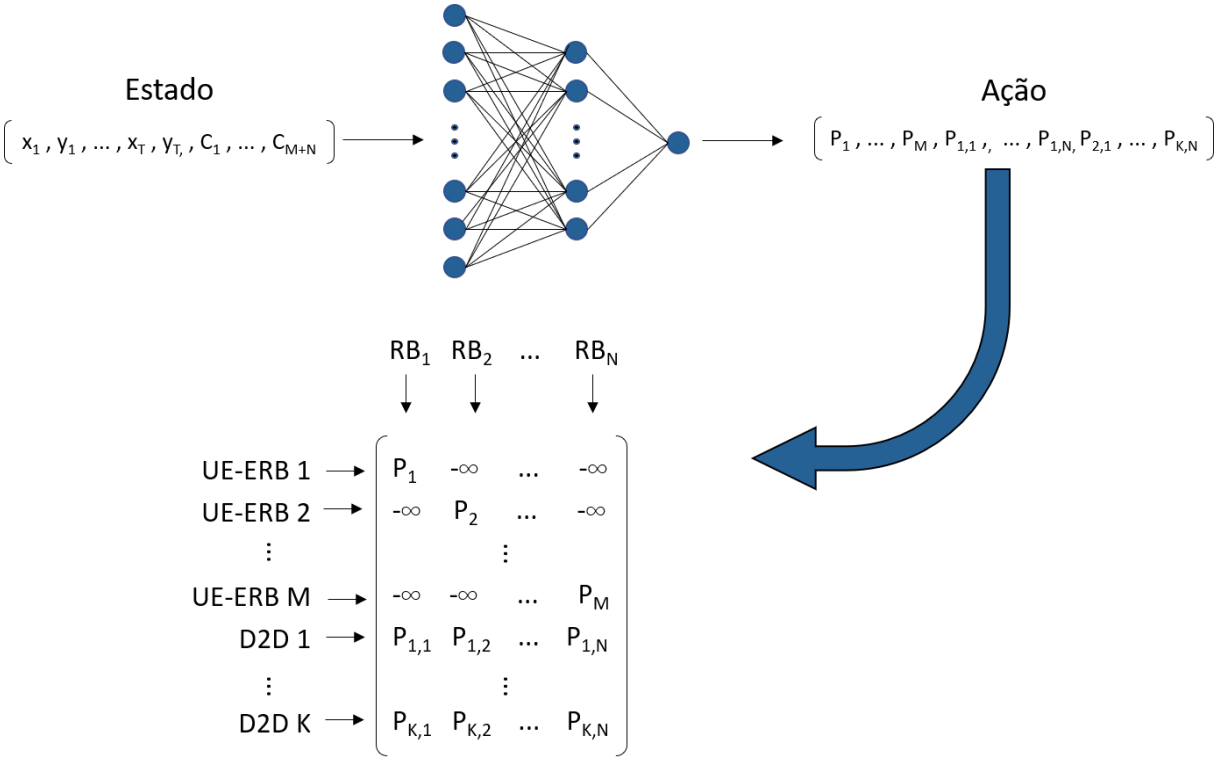


Figura 4.1: Ilustração com a transformação da saída da rede neural para a alocação de recursos feita para cada comunicação nos diferentes RBs disponíveis.

cada alocação, utilizou-se uma função de recompensa que é função das capacidades de transmissão das comunicações do sistema.

Como o intuito do algoritmo é maximizar a eficiência espectral da célula, evitando o aumento da probabilidade de *outage* das comunicações prioritárias, a função de recompensa utilizada é uma composição de duas funções:

$$R = r_{outage} + r_{capacidade} \tag{4.1}$$

em que as equações que definem r_{outage} e $r_{capacidade}$ estão detalhadas na Equação 4.2 e na Equação 4.3, respectivamente.

$$r_{outage} = \frac{1}{M} \sum_{j=1}^{M} \lambda_1 r_{outage,j} \tag{4.2}$$

$$r_{capacidade} = \frac{1}{K} \sum_{i=1}^{K} \lambda_2 r_{capacidade,i} \tag{4.3}$$

para

$$r_{outage,j} = \begin{cases} (2 - 2^{-SNIR_{UE_j} + (SNIR_{min}+1)}), & SNIR_{UE_j} \geq SNIR_{min} \\ (-3 + 0.3 \cdot SNIR_{UE_j})), & cc \end{cases} \tag{4.4}$$

$$r_{capacidade,i} = \begin{cases} C_{D2D_i}, & \text{comunicação prioritária do RB não está em } outage \\ -10, & cc \end{cases} \tag{4.5}$$

em que $SNIR_{UE_j}$ é a SNIR do j-ésimo UE do sistema, $SNIR_{min}$ é a SNIR que garante a capacidade mínima para o serviço requisitado, C_{D2D_i} é a capacidade de transmissão do i-ésimo D2D do sistema e λ_1 e λ_2 são constantes de ajuste da função de recompensa.

A função de recompensa $r_{outage,j}$ é uma função não linear constituída por duas funções, uma não linear que possui crescimento acelerado a partir do ponto que a SNIR do UE passa a ser superior que a SNIR mínima e uma linear, que aumenta à medida em que a SNIR do UE se aproxima da SNIR mínima. O gráfico dessa função está mostrado na Figura 4.2.

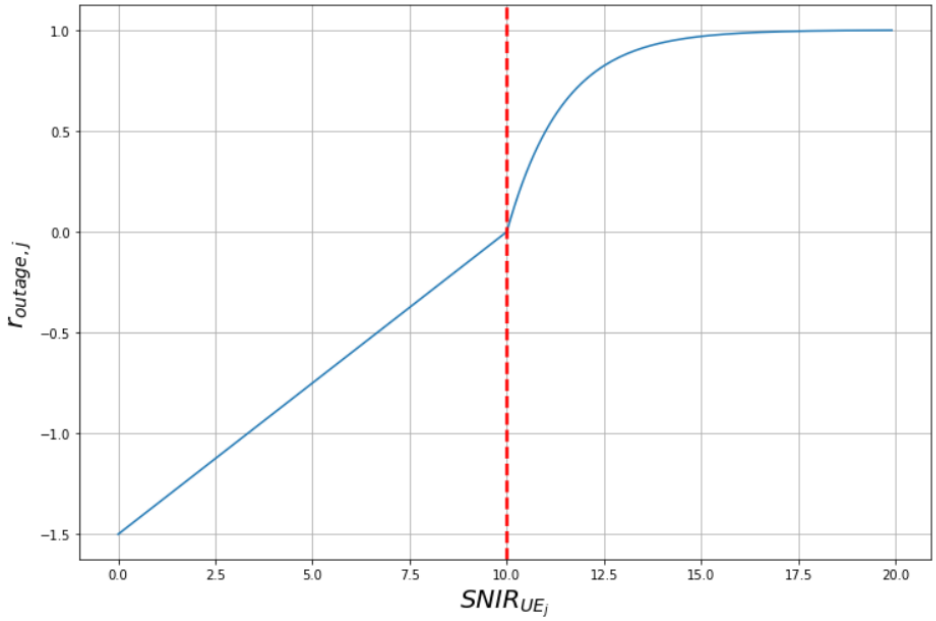


Figura 4.2: Gráfico da recompensa $r_{outage,j}$ em função de $SNIR_{UE_j}$, para $SNIR_{min} = 10$ dB (linha vertical em vermelho) e $\lambda_1 = 0.5$.

Já a função de recompensa $r_{capacidade,i}$ é uma função que cresce linearmente com a capacidade do i-ésimo D2D, desde que a comunicação prioritária do RB em que a potência do i-ésimo D2D foi majoritariamente distribuída não esteja em *outage*. O intuito dessa função é recompensar alocações que maximizem a capacidade de comunicação, desde que isso não aumente a taxa de *outage* das comunicações prioritárias.

4.2.2 Processo de aprendizagem

A partir da definição sobre as representações dos estados, ações e recompensas, definiu-se que o processo de treinamento do modelo se basearia no DDPG e no TD3, pelo fato de as saídas do agente (as potências de transmissão) serem de natureza contínua.

As redes neurais do agente e do crítico são DNNs com duas camadas ocultas. Além disso, utilizou-se o *Experience Replay* e as *Target Networks* para melhorar a convergência do processo. A estratégia de exploração adotada foi a adição de ruído nos parâmetros da rede neural do agente antes da tomada de decisão.

Cada episódio do processo se inicia com a inicialização dos dispositivos em posições randômicas da célula, respeitando uma distribuição uniforme, e, ao longo do episódio, tais dispositivos se locomovem pela célula em direção a um ponto também escolhido de maneira randômica.

Curriculum Learning

Inspirado pelas recentes técnicas na área de *curriculum learning* aplicadas em algoritmos de DRL [41], optou-se por iniciar o processo de treinamento com episódios em que a interferência sofrida pela comunicação prioritária era, em geral, menos significativa. Para isso, a aleatoriedade na inicialização das posições dos dispositivos a cada novo episódio foi limitada. Tal limitação garantia que o UE ainda fosse inicializado em uma posição randômica, mas tal posição se encontrava dentro de um quadrado de lado L. À medida em que os episódios se passavam, o quadrado era aumentado, até chegar aos limites da célula, de forma que, nos últimos episódios do processo de treinamento, todos os dispositivos poderiam ser inicializados em qualquer ponto da célula.

A estratégia utilizada foi de, nos primeiros 25% dos episódios do processo de aprendizagem, limitar a inicialização do UE em um quadrado centralizado com $L = 300$ metros; entre o 25% e o 75%, limitar a um quadrado com $L = 600$ metros; e, por fim, depois de 75%, retirar qualquer limitação. Essa estratégia está detalhada na Figura 4.3

Todos os parâmetros do processo de aprendizagem estão expostos na Tabela 4.3.

4.2.3 Implementação e teste do algoritmo

Ao final do processo de aprendizagem, tanto a rede neural do agente quanto a do crítico possuirá uma combinação de parâmetros que permitirão que elas desempenhem seu papel adequadamente: o agente fazendo a alocação de recursos seguindo as premissas adotadas e o crítico sendo capaz de avaliar a alocação realizada pelo agente. Assim, para a implementação e testagem do algoritmo, é necessário utilizar apenas a rede neural do agente

Tabela 4.3: Configurações do processo de aprendizagem dos algoritmos baseados no DDPG e no TD3.

Parâmetro	**DDPG**	**TD3**
Tamanho da memória de *replay*	10^4	
Batch size	64	
Episódios de treinamento	1500	
Número de camadas ocultas das redes neurais	2	
Dimensão da primeira camada oculta das redes neurais	256	
Dimensão da segunda camada oculta das redes neurais	128	
Otimizadores das redes neurais	AdamW	
Taxa de aprendizagem do agente	$2 \cdot 10^{-3}$	
Taxa de aprendizagem do crítico	$5 \cdot 10^{-4}$	
Fator de desconto do retorno (γ)	0.9	
Constante de ajuste da função de r_{outage} (λ_1)	0.5	
Constante de ajuste da função de $r_{capacidade}$ (λ_2)	0.02	
Fator de suavização da atualização (τ)	0.05	
Intervalo de sincronização das redes *target*	1	
Desvio padrão inicial do ruído de exploração (σ_0)	5	
Coeficiente de adaptação do ruído de exploração (α)	1.04	
Intervalo para adaptação do ruído de exploração (T_{adapt})	5	
Intervalo para atualização do agente (T_{update})	-	2
Desvio padrão do ruído adicionado nas ações (σ_μ)	-	0.2
Limitantes do ruído adicionado nas ações (c)	-	0.5

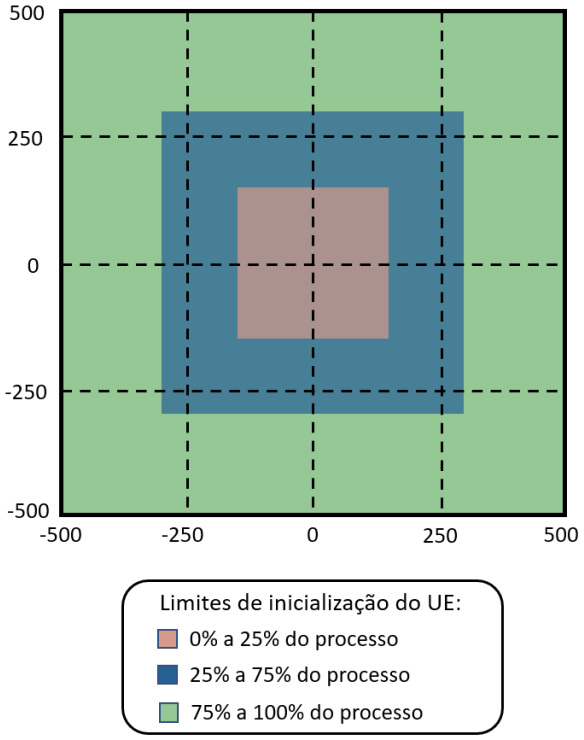


Figura 4.3: Ilustração dos limites de inicialização do UE durante o processo de aprendizagem do algoritmo seguindo uma estratégia baseada nas técnicas de *curriculum learning*.

com os parâmetros desenvolvidos no processo de aprendizagem, de forma que o crítico é importante apenas para o treinamento.

Dessa forma, para utilização desse algoritmo, basta implementar uma rotina que receba as informações do sistema de comunicações, agregá-las para formar o estado do ambiente como foi definido e passá-lo pela rede neural do agente, reorganizando o vetor de saída para a matriz com as potências em cada RB (como mostrado na Figura 4.1).

Apesar de a rede neural ter sido treinada para fazer alocação em exemplos diversos, existem situações críticas em que é muito difícil proteger a comunicação prioritária, mesmo maximizando a potência do UE e minimizando as potências dos pares D2D.

Por esse motivo, no modelo proposto, é adotada uma estratégia para evitar a ocorrência de *outage* da comunicação prioritária em situações em que a ERB e um dispositivo transmissor de uma comunicação D2D estão muito próximos.

A estratégia consiste em delimitar um raio de segurança R_S em torno da ERB, de forma que, se um dispositivo transmissor de uma comunicação D2D estiver a uma distância menor do que R_S, o algoritmo de alocação impede a realização da comunicação, mantendo-a desabilitada enquanto o D2D Tx estiver dentro da circunferência de raio R_S. O raio de proteção R_S adotado foi de 100 metros.

4.3 Conclusão

Este capítulo apresentou a modelagem do sistema de comunicações, dos algoritmos para alocação de recursos e dos processos de treinamento de teste deles. Assim, no Capítulo 5, estão condensados os resultados obtidos durante o processo de treinamento do modelo e depois do treino, utilizando rotinas de teste, que foram divididos em testes gerais e testes específicos.

5 Resultados

Esse capítulo reúne os resultados obtidos a partir da alocação de recursos realizada pelo modelo proposto em diferentes situações. Para averiguação do desempenho do modelo, serão feitas análises durante o processo de treinamento, para visualização da convergência do algoritmo, e durante situações de teste após o treinamento do modelo.

5.1 Processo de treinamento

Como visto no Capítulo 2 e no Capítulo 4, os processos de treinamento do DDPG e do TD3 são bastante similares, apresentando apenas algumas diferenças sutis, o que faz com que os resultados também sejam próximos. Os algoritmos foram treinados seguindo as configurações mostradas na Seção 4.2.2.

Os gráficos com a média das recompensas obtidas em cada episódio do processo de treinamento, mostrado na Figura 5.1 é capaz de ilustrar a convergência do aprendizado para sistemas com diferentes valores de *resourceblocks*. Além da média, a Figura 5.1 expõe o intervalo de confiança dos valores de recompensa alcançados por ambos os algoritmos. Para a elaboração dessas curvas, os algoritmos foram treinados com diferentes números de pares D2D no sistema de comunicações (K), de forma que $1 \leq K \leq 12$.

Além da visualização da convergência do processo de treinamento através da aquisição das recompensas, é necessário verificar que o agente está aprendendo a fazer a alocação de recursos como se espera. Como uma das principais atribuições do algoritmo é evitar que as comunicações prioritárias entrem em *outage*, a Figura 5.2 ilustra como o modelo aprende a proteger a comunicação prioritária ao longo dos episódios.

Como esperado, a alocação realizada pelos modelos no início do processo de treinamento produz altas taxas de *outage*, mas ao longo dos episódios a taxa reduz gradativamente.

É possível perceber também os efeitos do *curriculum learning* ao longo do processo de treinamento. Como os modelos foram treinados através de 1500 episódios, a mudança dos limites de inicialização detalhada na Figura 4.3 acontece nos episódios 375 e 1125.

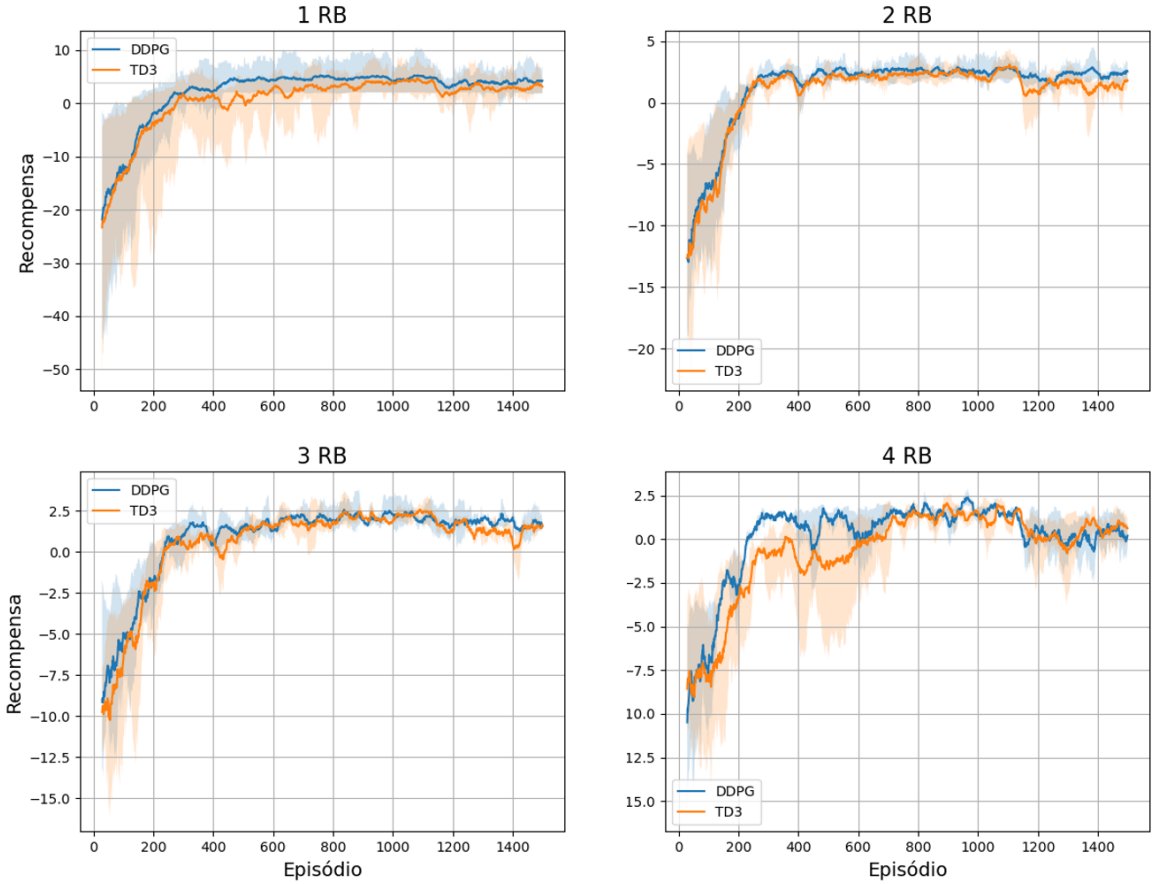


Figura 5.1: Média da recompensa recebida pelo algoritmo em cada episódio do processo de treinamento dos modelos em sistemas com 1 RB, 2 RBs, 3 RBs e 4 RBs.

No gráfico, percebe-se que há um aumento da taxa de *outage* média imediatamente após esses episódios, que é controlada com o passar dos episódios.

Analisando a Figura 5.1 e a Figura 5.2, é possível perceber que o aumento de dimensionalidade da saída da rede neural causada pela estratégia ilustrada na Figura 4.1 faz com que o desempenho do agente seja comprometido. O motivo para essa queda de rendimento é que, à medida em que o número de RBs (N) aumenta, a rede neural precisa aprender, utilizando apenas um valor de recompensa, a distribuir a potência dos dispositivos em $M + N \cdot K$ posições, em que M é o número de UEs e K é o número de pares D2D. Dessa forma, percebe-se que a média de recompensas obtidas ao longo dos episódios se estabiliza em valores mais altos e a taxa de *outage* em valores mais baixos para sistemas com menos RBs.

Após o treinamento, os modelos foram submetidos a situações de teste, como foi detalhado no Capítulo 4.

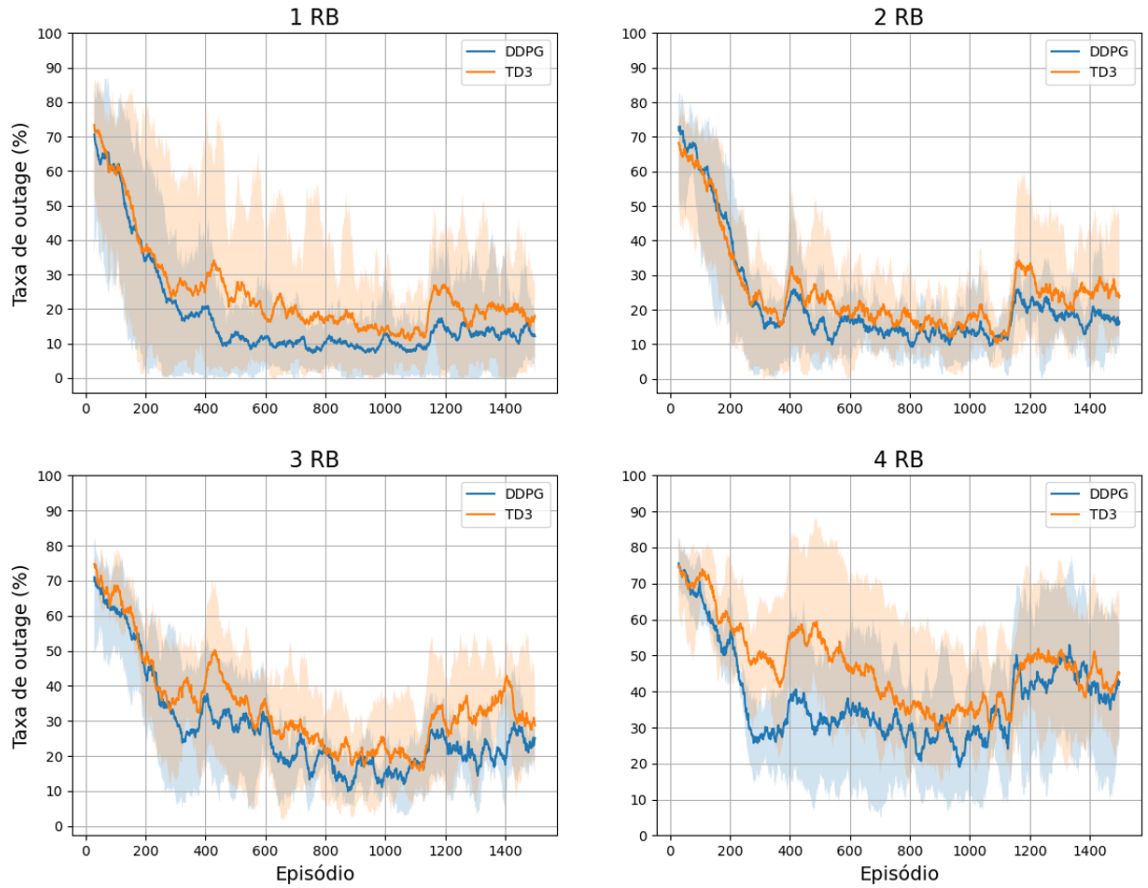


Figura 5.2: Taxa de *outage* em cada episódio do processo de treinamento dos modelos em sistemas com 1 RB, 2 RBs, 3 RBs e 4 RBs.

5.2 Teste dos modelos

O processo de testagem dos modelos se dividiu em duas óticas:

- A primeira ótica é generalista, em que o modelo é submetido a um grande número de simulações e, ao final, as informações são extraídas a partir das grandezas estatísticas computadas nesse conjunto;
- A segunda é específica, em que o modelo é testado em situações bem definidas e as informações são extraídas a partir dos dados coletados em cada simulação.

Assim, a apresentação dos resultados será dividida nessas duas óticas.

5.2.1 Testes gerais

Para a testagem geral dos algoritmos, foi desenvolvido um roteiro com situações geradas aleatoriamente. Utilizando esse mesmo roteiro, cada um dos algoritmos foi testado, afim

de garantir a equidade na comparação entre cada um dos algoritmos.

Assim como os resultados de treino, os gráficos gerados nesses roteiros de teste foram divididos a partir do número de RBs, já que o desempenho do agente se mostrou dependente desse valor, que é o principal responsável por aumentar a dimensionalidade do problema de maneira significativa.

A Figura 5.3 condensa as taxas de *outage* e a eficiência espectral obtidas a partir da alocação de recursos realizada pelos algoritmos treinados, o DDPG e o TD3, além de fazer uma comparação com uma alocação de recursos realizada aleatoriamente que respeita os limites de potência máxima e mínima impostos aos algoritmos treinados (expostos na Tabela 4.2) e também respeita a distância mínima entre um D2D Tx e a ERB (R_S) para que uma comunicação D2D seja efetuada.

A Figura 5.3 mostra que a alocação realizada por ambos os algoritmos inteligentes, o DDPG e o TD3, é muito mais eficiente em proteger a comunicação prioritária do que a alocação de recursos feita aleatoriamente, que define as potências de transmissão seguindo uma distribuição uniforme limitada entre os valores da potência mínima e máxima dos dispositivos.

Ainda na Figura 5.3, é possível perceber que a alocação aleatória resulta nos maiores valores de eficiência espectral do sistema, o que se deve à não preocupação com a alta taxa de *outage* da comunicação prioritária, de forma que a alocação define altas potências para as comunicações D2D, o que aumenta as taxas de comunicações dessas comunicações, mas também aumenta a interferência causada na comunicação entre UE e ERB, provocando o *outage*.

Assim, a análise da eficiência espectral deve ser condicionada à taxa de *outage*, se esta última não for controlada pelo algoritmo utilizado, a segunda análise é praticamente irrelevante.

Nesse sentido, outra constatação tirada a partir da Figura 5.3, é que, no geral, o agente treinado utilizando o DDPG foi mais eficiente do que o TD3 na proteção das comunicações prioritárias, obtendo taxas de *outage* inferiores. Somado a isso, percebe-se que as eficiências espectrais obtidas por ambos os algoritmos possuem valores bastante próximos, sofrendo pequenas variações de um caso para o outro.

Esses dois fatores sugerem que o algoritmo treinado utilizando o DDPG seja mais eficiente em fazer a alocação de recursos em sistemas como esse, o que constitui um resultado inesperado, visto que o TD3 é considerada uma versão aprimorada do DDPG, que apresenta maior estabilidade e melhor desempenho [32]. Por esse motivo e para simplificação do entendimento dos gráficos, as análises subsequentes mostrarão apenas os resultados obtidos a partir da utilização do DDPG, que se mostrou o melhor candidato para o desempenho do exercício.

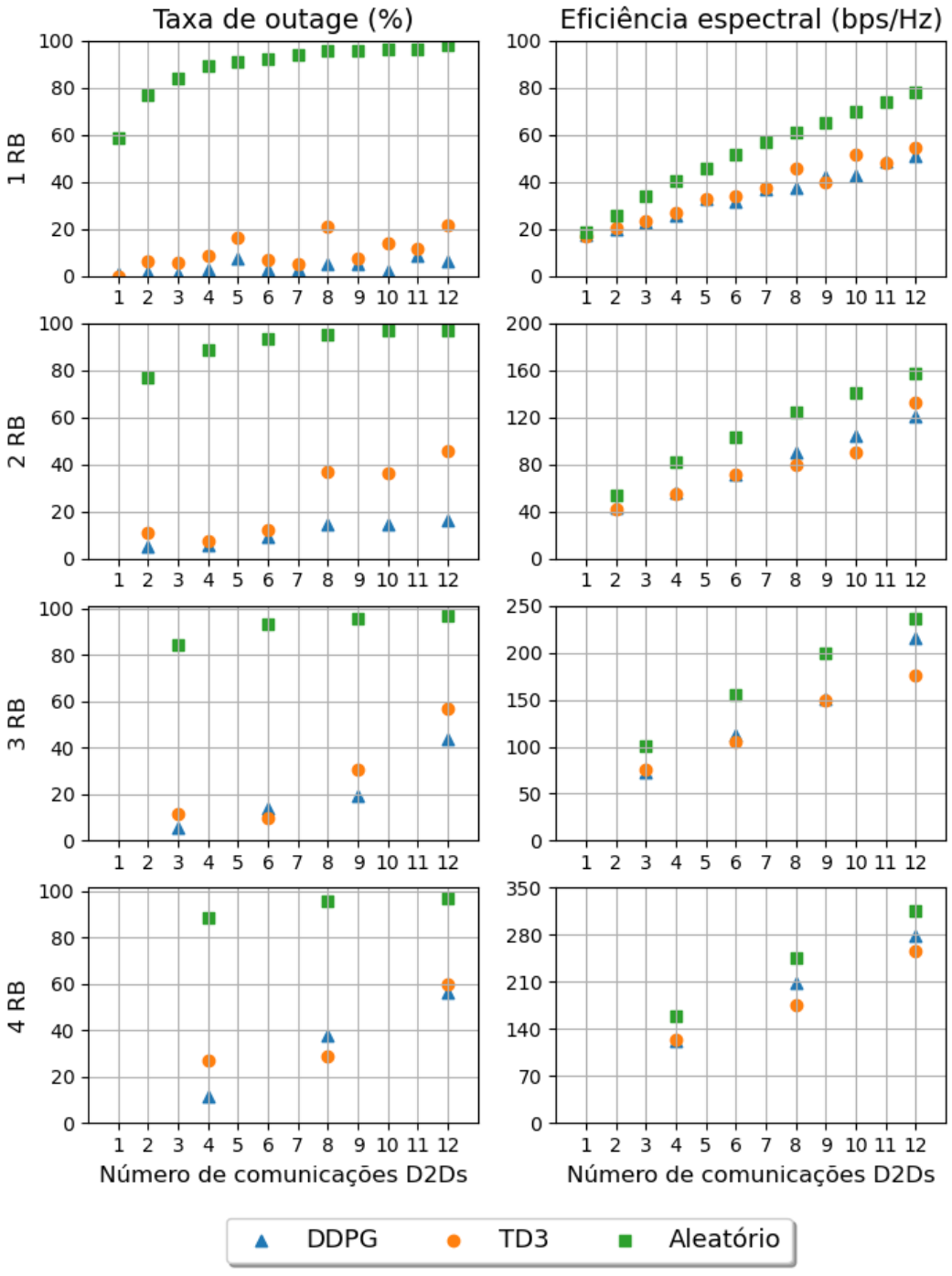


Figura 5.3: Taxa de *outage* e eficiência espectral dos modelos em função do número de comunicações D2D em sistemas com 1 RB, 2 RBs, 3 RBs e 4 RBs.

Os gráficos da Figura 5.3 confirmam o que havia sido observado na Figura 5.2, o aumento do número de RBs acarreta o aumento da taxa de *outage*, mesmo se compararmos com sistemas que tenham a mesma razão $\frac{K}{N}$, isto é, a relação entre número de comunicações D2D e número de RBs disponíveis. Para facilitar essa visualização, a Figura 5.4 traz os resultados em função da razão $\frac{K}{N}$ para sistemas com diferentes quantidades de RBs disponíveis.

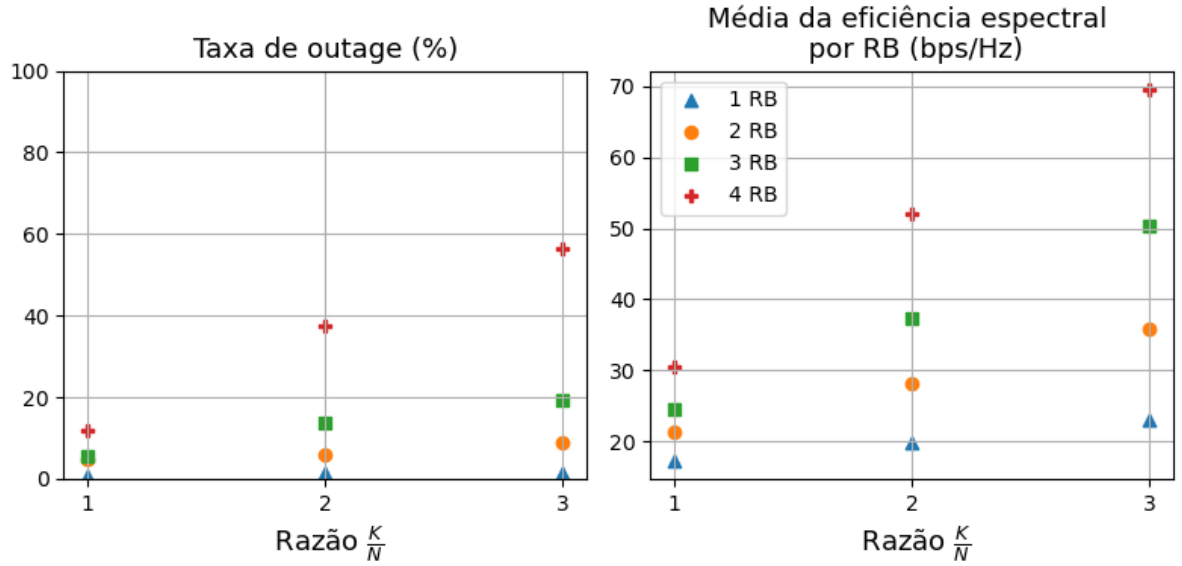


Figura 5.4: Taxa de *outage* e eficiência espectral obtidos pelo DDPG em função da razão $\frac{K}{N}$ em sistemas com 1 RB, 2 RBs, 3 RBs e 4 RBs.

Além da taxa de *outage* e da eficiência espectral, deseja-se verificar se o algoritmo proposto faz a alocação de maneira inteligente, levando em consideração as posições dos dispositivos para tal. Uma maneira de se fazer isso é plotar a média da potência das comunicações D2D em função da posição do UE e em função da posição do transmissor D2D. A Figura 5.5 agrupa essas plotagens em visualizações 2D e 3D.

Na Figura 5.5, nos gráficos a e b, é possível perceber que o algoritmo aloca potências maiores para as comunicações D2D quando o UE está próximo da ERB, isto é, em momentos em que a perda no caminho da comunicação prioritária é menor, permitindo que os pares D2D se comuniquem sem provocar *outage* nesta.

Paralelamente, nos gráficos c e d, é possível perceber que o algoritmo aloca potências maiores para as comunicações D2D quando o transmissor D2D está mais distante da ERB, já que nessa situação a perda no caminho entre ambas é maior, causando menos interferência na comunicação prioritária.

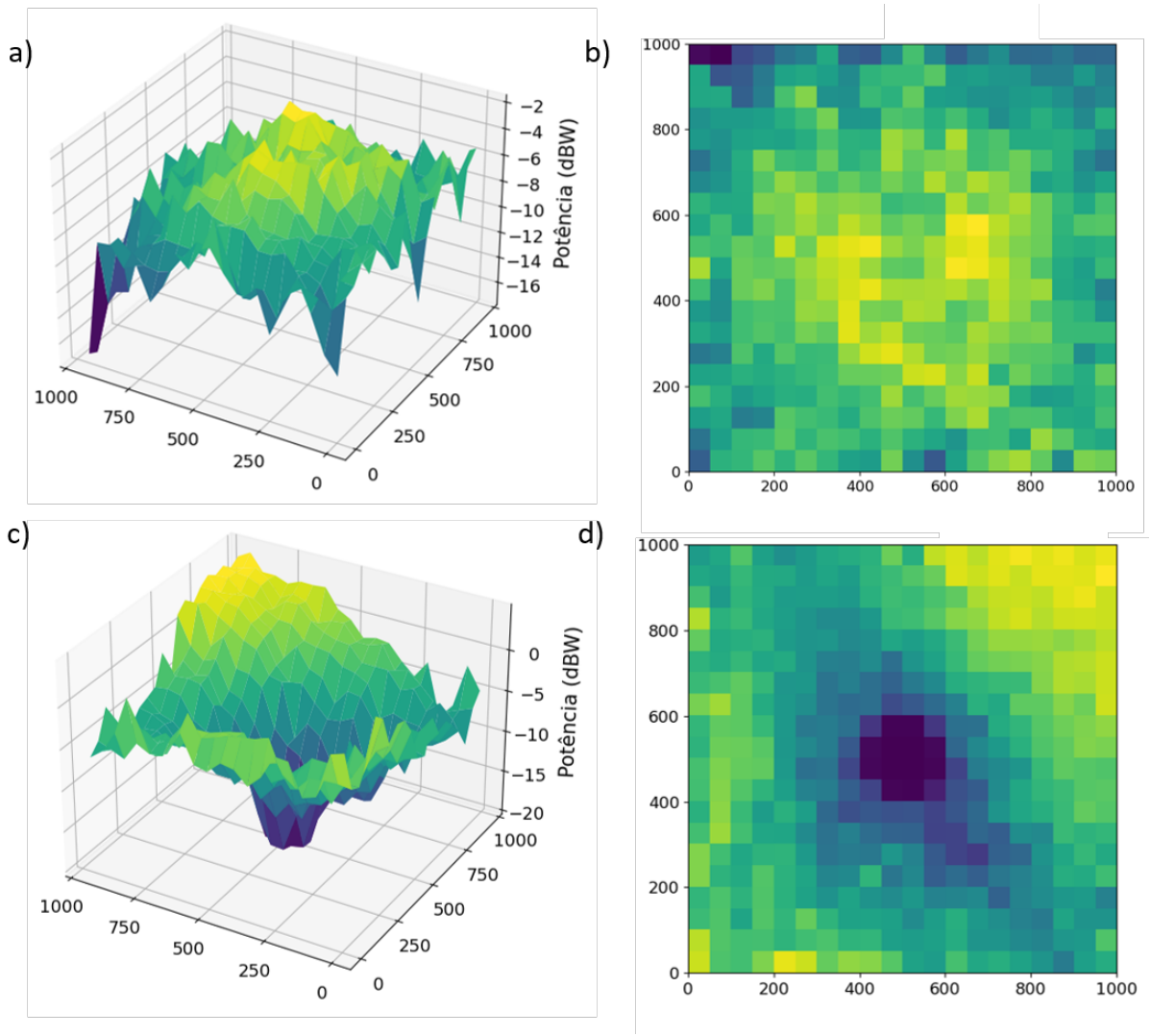


Figura 5.5: Média das potências das comunicações D2D alocadas pelo DDPG em função da posição dos dispositivos. a) Visualização 3D da média em função da posição do UE. b) Visualização 2D da média em função da posição do UE. c) Visualização 3D da média em função da posição do transmissor da comunicação D2D em questão. d) Visualização 2D da média em função da posição do transmissor da comunicação D2D em questão.

Assim, os testes gerais mostram que o modelo aprende a fazer a alocação de maneira inteligente, protegendo a comunicação prioritária e aumentando a capacidade de transmissão das comunicações D2D, quando possível. Entretanto, percebe-se que o desempenho do algoritmo piora rapidamente à medida que a dimensionalidade das entradas e saídas aumenta, problema comum em algoritmo monoagente com muitos parâmetros para decidir.

Além dos testes gerais, que são importantes para diagnosticar o desempenho do modelo através das principais variáveis, é interessante fazer testes em situações específicas para a

visualização dos detalhes da atuação do algoritmo durante um episódio.

5.2.2 Testes específicos

A realização dos testes específicos se deu a partir da predefinição sobre quais seriam as trajetórias que cada dispositivo faria no episódio em questão. A partir dessa definição, o ambiente é organizado como previsto e o algoritmo é utilizado para fazer a alocação de recursos de cada uma das comunicações envolvidas.

Para facilitar a visualização dos resultados, esses testes foram feitos em um sistema com apenas 1 RB contendo uma comunicação prioritária e duas comunicações D2D. Além disso, os pares D2D possuem mesma velocidade, 40 km/h, ao passo que o UE se locomove pela célula com velocidade de 80 km/h.

Os resultados serão expostos em duas figuras:

- A primeira é a evolução das posições dos dispositivos comunicantes na célula, com cada seta representando onde o dispositivo estava no início do episódio (início da seta) e onde ele estava no fim do episódio (ponta da seta). Para facilitar a visualização, optou-se por utilizar apenas uma seta para representar ambos os dispositivos dos pares D2D, já que a distância entre ambos é pequena, se comparada às demais.
- A segunda condensa as potências alocadas para cada comunicação do sistema e os resultados gerados a partir da definição dessas potências, que são a eficiência espectral do sistema e a taxa de *outage* da comunicação prioritária.

Situação 1

A primeira situação de teste, mostrada na Figura 5.6, é um episódio em que todos os dispositivos se aproximam da ERB e existe equidistância entre os pares D2D e o UE e entre os pares D2D e a ERB. A alocação feita pelo modelo está exposta na Figura 5.7.

A Figura 5.7 mostra que a alocação realizada pelo modelo possui baixa variância entre eventos subsequentes, de forma que a evolução ao longo do episódio se assimila a uma curva contínua. Apesar de esse comportamento parecer intuitivo, o algoritmo não recebe qualquer informação temporal quanto às alocações feitas no passado como entrada e a rede neural utilizada, a DNN, não possui memória, o que faz com que a definição das potências em cada evento seja descorrelatada da definição das potências em eventos passados. Apesar disso, o gráfico mostra que as curvas possuem baixíssima variância em eventos subsequentes, o que indica que o algoritmo consegue definir as tendências de crescimento ou decrescimento das potências sem produzir um resíduo considerável.

Outra constatação a partir da Figura 5.7 é que o modelo define as potências das comunicações D2D de maneira quase idêntica, o que é esperado para situações simétricas como

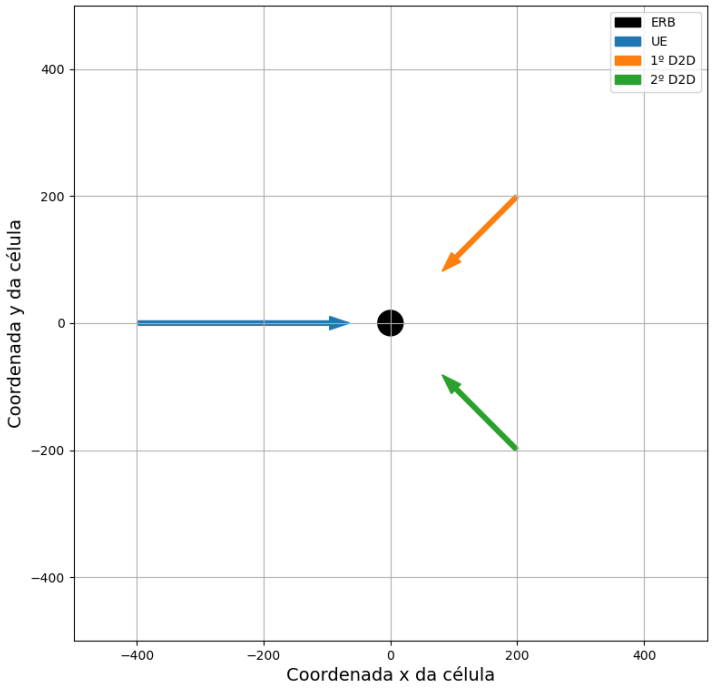


Figura 5.6: Evolução dos dispositivos na célula ao longo do episódio de teste da situação 1.

esta, a não ser que haja efeitos distintos produzidos pelo *shadowing*. Percebe-se, também, que as potências de ambas as comunicações D2D são mantidas praticamente constantes durante todo o episódio, provavelmente porque existem duas forças se contrapondo: a aproximação entre UE e ERB, que promove um aumento da SNIR na comunicação prioritária; e a aproximação entre os D2Ds Tx e a ERB, que promove uma diminuição da SNIR da comunicação prioritária.

A alocação realizada nesse episódio de teste consegue proteger a comunicação prioritária durante todos os eventos, não ocorrendo *outage* em nenhum momento, e aumentando a eficiência espectral da comunicação primária, ao passo que a eficiência espectral das comunicações D2D segue uma tendência decrescente, motivada pela aproximação de todos os dispositivos simultaneamente.

Situação 2

A segunda situação de teste, mostrada na Figura 5.8, é praticamente oposta à primeira. Neste episódio, todos os dispositivos iniciam próximos à ERB, mas depois se afastam dela, mantendo equidistância entre os pares D2D e o UE e entre os pares D2D e a ERB, o que garante simetria ao problema. A alocação feita pelo modelo está exposta na Figura 5.9

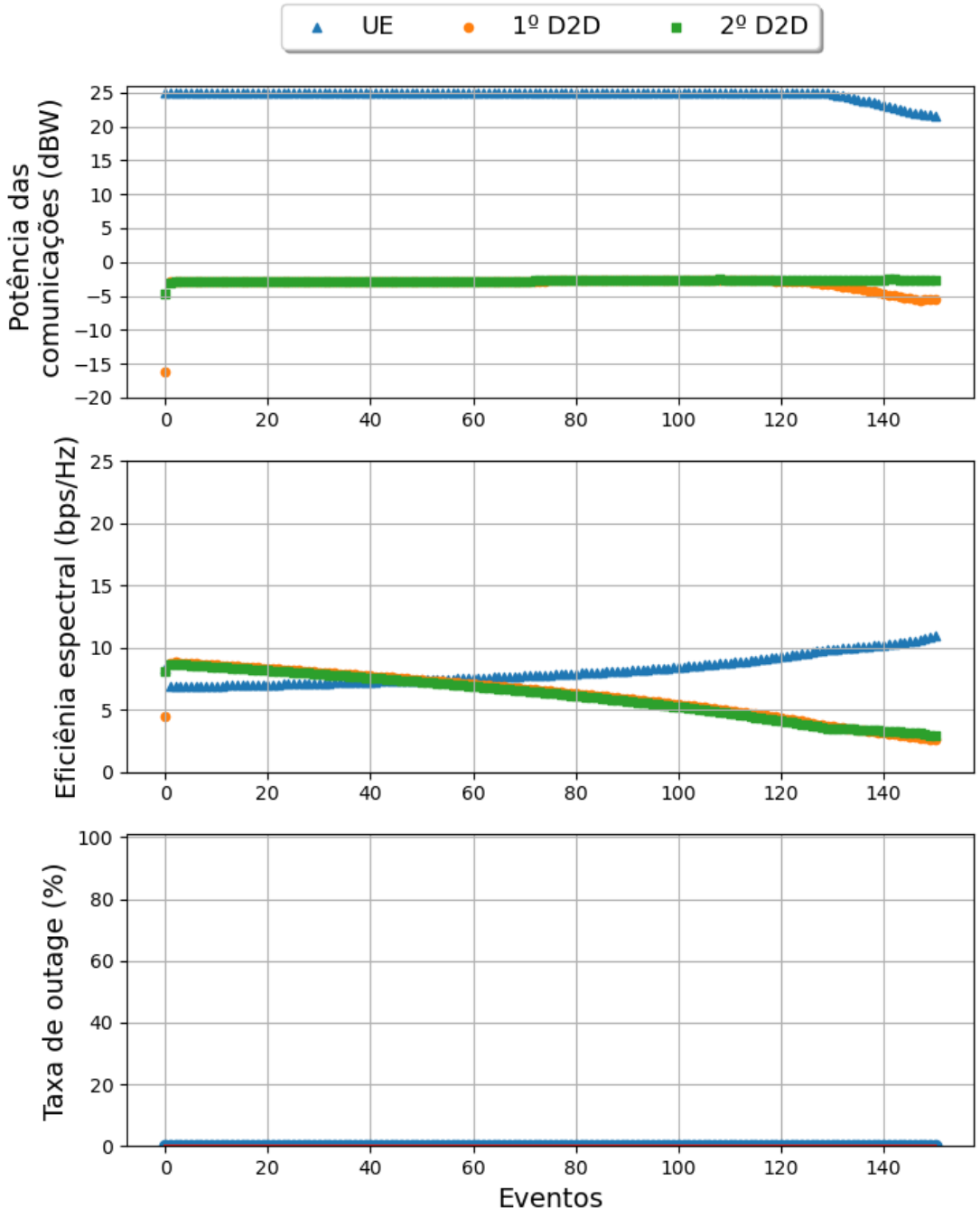


Figura 5.7: Alocação de recursos realizada pelo algoritmo ao longo do episódio de teste da situação 1.

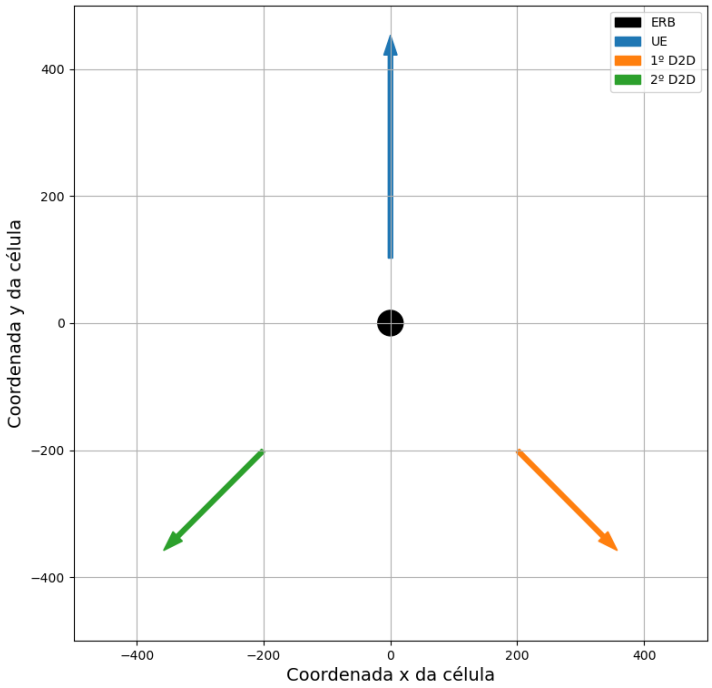


Figura 5.8: Evolução dos dispositivos na célula ao longo do episódio de teste da situação 2.

A Figura 5.9 mostra que a definição de potências nesse caso também possui baixa variância entre eventos subsequentes, o que indica robustez no treinamento do algoritmo, que entende o ambiente a ponto de fazer alocações correlatas entre os eventos, mesmo que as definições de potência sejam independentes umas das outras.

Durante o episódio, a potência da comunicação prioritária possui comportamento decrescente ao longo dos primeiros eventos, mas a partir da metade do episódio ela se estabiliza em um valor próximo a 15 dBW, sofrendo pequeno aumento nos últimos eventos. Já para as comunicações D2D, diferentemente da alocação na situação 1, a definição das potências na situação 2 possui grandes diferenças, apesar da simetria existente. Um possível motivo para isso são os efeitos produzidos pelo *shadowing*, que possuem dependência espacial, de forma que é possível que o 2º par D2D esteja em um ambiente em que os efeitos de desvanecimento do *shadowing* são maiores do que os efeitos vivenciados pelo 1º D2D.

Analisando o segundo gráfico da Figura 5.9, percebe-se que o algoritmo não mantém a eficiência espectral das comunicações D2D em níveis próximos, mas ambos possuem valores relativamente altos. Quanto à eficiência espectral da comunicação prioritária, a alocação feita faz com que ela seja decrescente durante a maior parte do episódio, mas, ao final do episódio, ela possui um pequeno crescimento, porque o seu valor se aproximou da

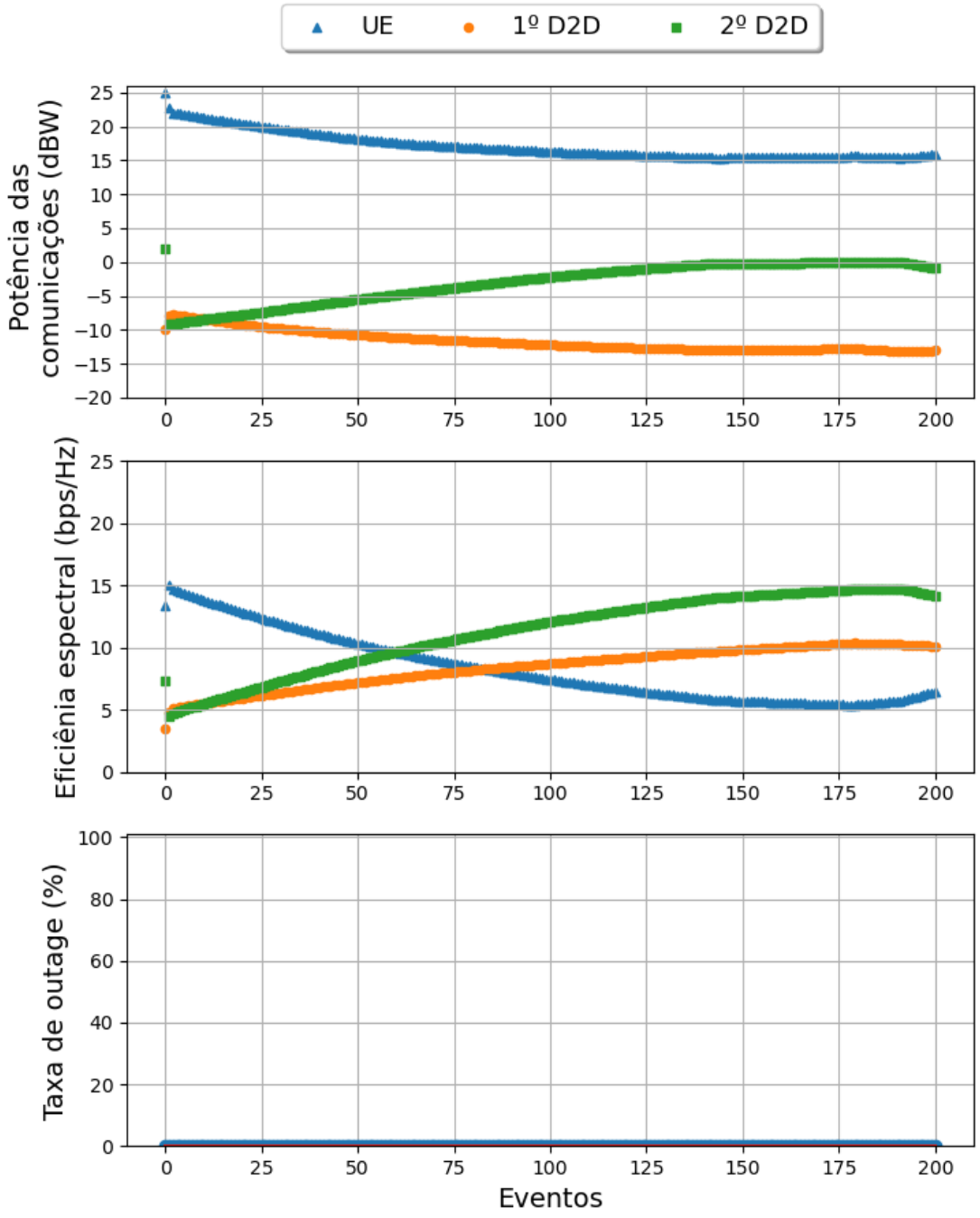


Figura 5.9: Alocação de recursos realizada pelo algoritmo ao longo do episódio de teste da situação 2.

eficiência espectral mínima adotada (para a SNIR mínima de 10 dB, a eficiência espectral mínima é aproximadamente 3.5 dB).

Novamente, a taxa de *outage* é de 0% durante o episódio, mostrando que o algoritmo protege a comunicação prioritária, que é o principal objetivo do modelo.

Situação 3

A terceira situação de teste, mostrada na Figura 5.10, é praticamente uma combinação entre as duas situações anteriores, nela, o UE se aproxima da ERB, mas os pares D2D se afastam dela, também mantendo equidistância entre os pares D2D e o UE e entre os pares D2D e a ERB. A alocação feita pelo modelo está exposta na Figura 5.11.

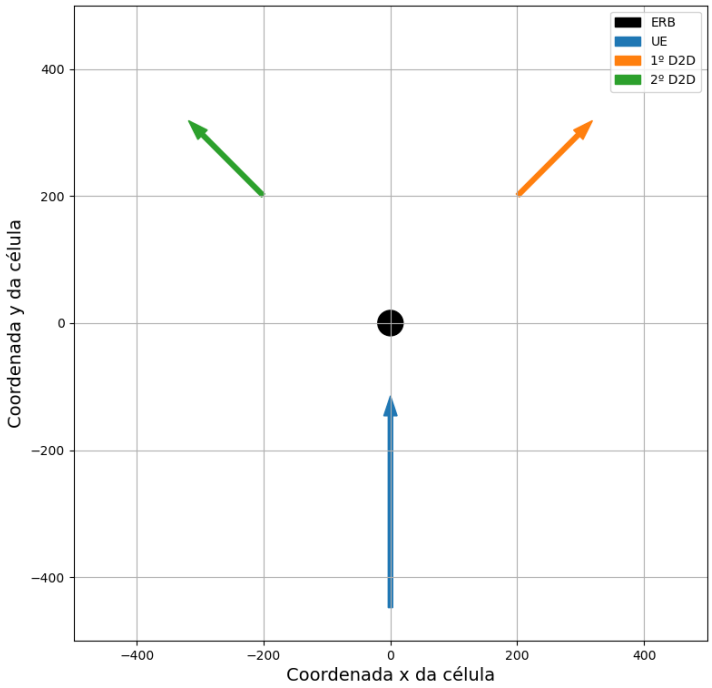


Figura 5.10: Evolução dos dispositivos na célula ao longo do episódio de teste da situação 3.

A Figura 5.11 também ilustra baixa variância entre eventos subsequentes, o que ratifica as análises feitas para as outras duas situações.

Por outro lado, a definição das potências nesse episódio parece incoerente com o que era esperado. Apesar da simetria existente entre os pares D2D, as potências alocadas para ambas possuem valores muito diferentes durante todo o episódio. No início, o 2º par D2D é privilegiado em detrimento do 1º par D2D. Com o passar dos eventos a condição de ambas se inverte, sem um motivo aparente, de forma que a alocação passa a privilegiar o 2º par D2D.

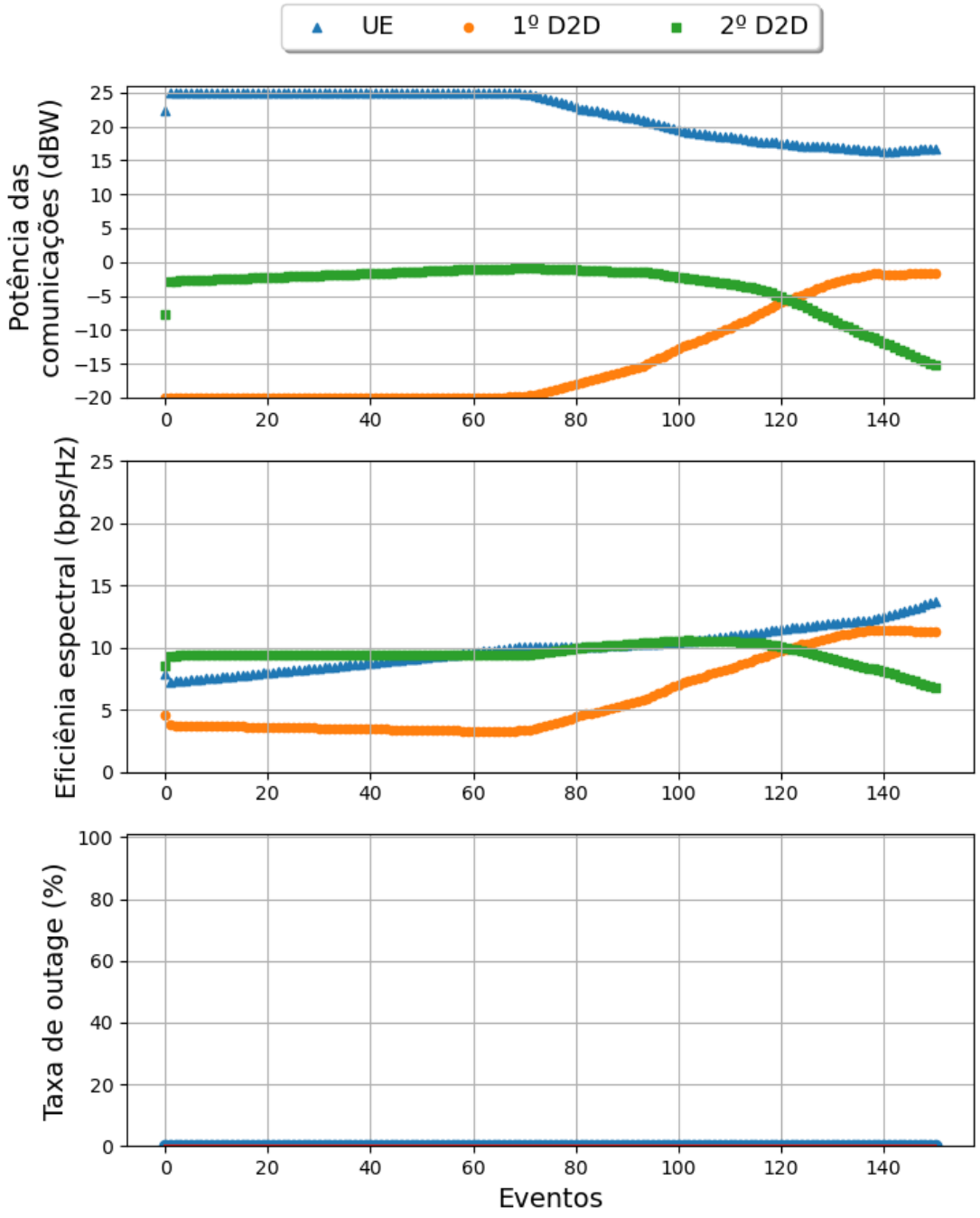


Figura 5.11: Alocação de recursos realizada pelo algoritmo ao longo do episódio de teste da situação 3.

Já a potência da comunicação prioritária possui tendência decrescente durante a maior parte do episódio, o que é esperado, já que a distância entre o UE e a ERB está diminuindo enquanto a distância entre os pares D2D e a ERB está aumentando. Dessa forma, o algoritmo protege a comunicação prioritária durante todo o episódio de teste, mantendo a taxa de *outage* igual a 0%.

Situação 4

Por fim, a quarta situação de teste, mostrada na Figura 5.12, ilustra uma circunstância crítica, em que o UE está longe da ERB e se locomovendo paralelamente à ERB e às comunicações D2D, enquanto os pares D2D estão próximos à ERB. Nessa situação, não há simetria entre os pares D2D. A alocação feita pelo modelo está exposta na Figura 5.13.

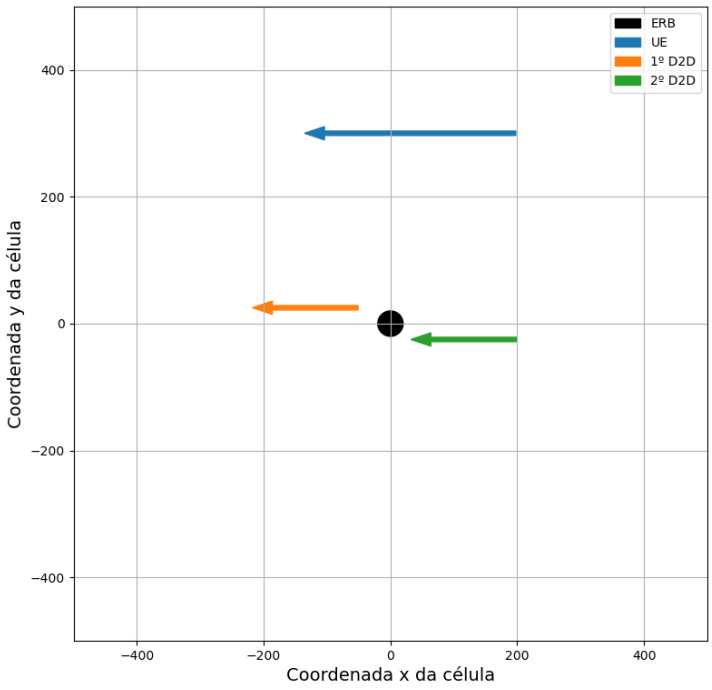


Figura 5.12: Evolução dos dispositivos na célula ao longo do episódio de teste da situação 4.

Nessa situação, é possível perceber o mecanismo de proteção à comunicação prioritária que desativa as comunicações D2D que estejam muito próximas à ERB, mais precisamente, em que o transmissor da comunicação esteja dentro de um raio R_S de 100 metros do centro da célula.

Nesse sentido, o 1º par D2D inicia o episódio com a comunicação desativada, enquanto o 2º par se comunica normalmente. A medida que o episódio avança, o 1º par se afasta da ERB, enquanto o 2º par se aproxima. Isso faz com que, por volta do evento 40,

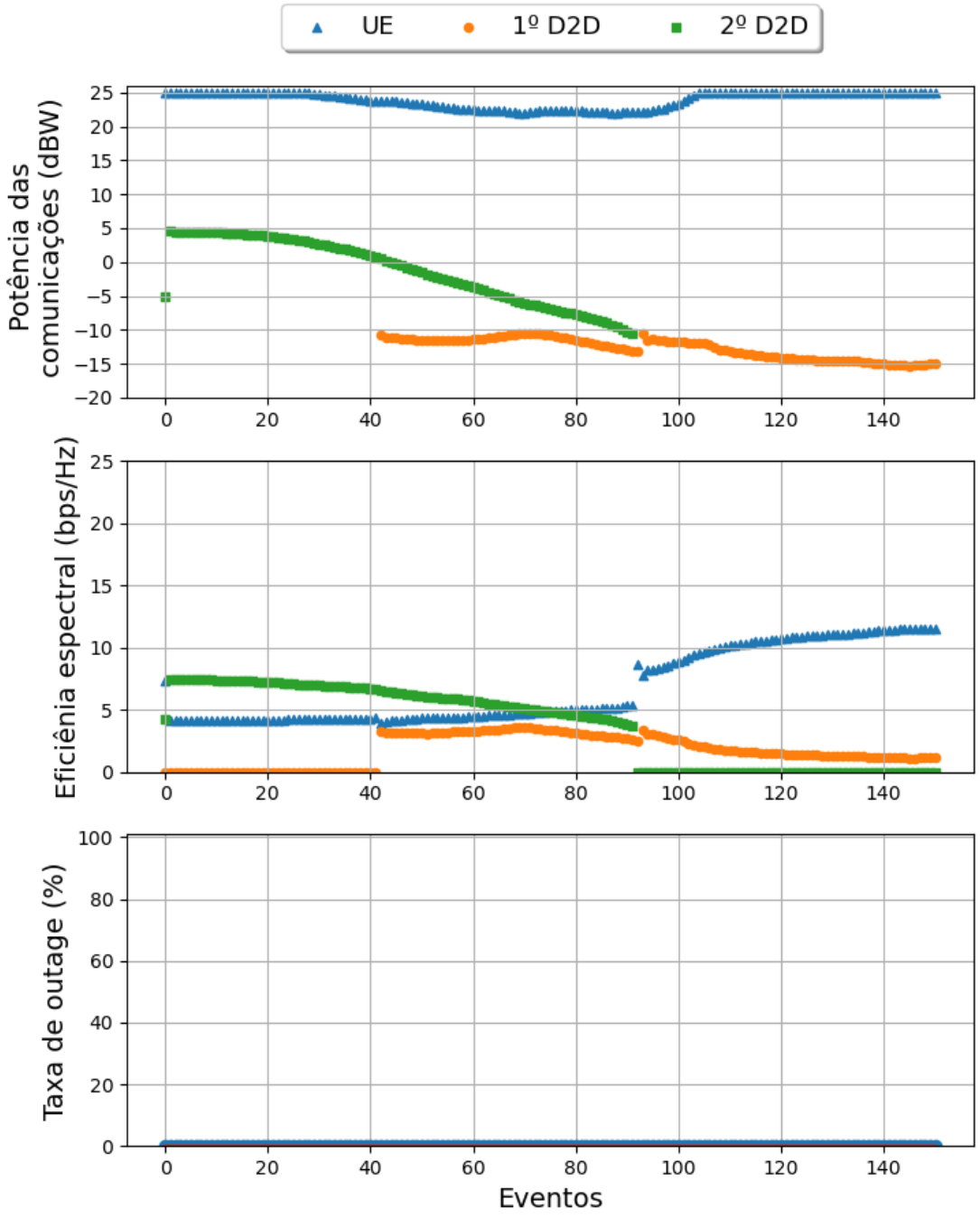


Figura 5.13: Alocação de recursos realizada pelo algoritmo ao longo do episódio de teste da situação 4.

a comunicação do 1º par seja habilitada, mantendo ambas até o evento 90, em que a comunicação do 2º par é desativada por ultrapassar o raio R_S, como pode ser visto na Figura 5.13.

É perceptível que as potências definidas possuem alguns saltos ao longo do episódio, motivados pela habilitação ou desabilitação de alguma comunicação D2D, que acontece de maneira abrupta.

Apesar da distância entre o UE e a ERB e da proximidade entre os pares D2D e a ERB, o algoritmo consegue proteger a comunicação prioritária durante todo o episódio, mantendo a taxa de *outage* em 0% novamente.

Apesar da situação crítica, após o evento 100, a comunicação prioritária apresenta valor da eficiência espectral alto, o que sugere que a potência do 1º par D2D poderia ter sido aumentada. Esse comportamento pode indicar que a rede neural tem dificuldade de aprender a atuar em situações em que é feita uma modificação brusca em alguma de suas saídas.

A partir dos resultados mostrados, é possível perceber que o algoritmo é muito eficiente em proteger a comunicação prioritária, principalmente para sistemas com baixa dimensionalidade. Além disso, é possível afirmar que o algoritmo é inteligente quanto à posição dos dispositivos, mesmo com um ambiente cuja quantidade de possíveis estados é enorme.

Somado a isso, percebe-se que a eficiência espectral do sistema aumenta consideravelmente com a inserção de comunicações D2D, mesmo que isso não reflita em um aumento significativo na taxa de *outage* (principalmente para sistemas com poucos RBs).

Por fim, os testes específicos mostram que a alocação é feita de maneira inteligente, mas que, em alguns momentos, as ações parecem não fazer sentido. Isso pode ser motivado pelo fato de que as simulações feitas possuem grau de liberdade total para movimentação dos dispositivos. Isso faz com que o algoritmo explore apenas alguns estados possíveis e generalize para todos os demais. Como visto, isso funciona para a maioria dos casos, mas é possível que, para alguns deles, essa generalização não seja suficiente.

5.3 Conclusão

A partir dos resultados, é perceptível que ambos os modelos convergiram durante o processo de treinamento, o que pode ser comprovado no processo de teste ao observar que as taxas de *outage* obtidas são muito menores do que quando a alocação é feita de maneira aleatória.

Apesar de ambos convergirem, entre os dois algoritmos inteligentes desenvolvidos, o DDPG teve resultados melhores do que os obtidos pelo TD3, já que ele conseguiu controlar

melhor as taxas de *outage*, mesmo sem obter taxas de transmissão das comunicações D2D muito menores.

A partir da percepção de que o DDPG fora mais efetivo, aprofundou-se na análise de seu desempenho e percebeu-se que ele realmente levou em conta as posições dos dispositivos para fazer a alocação, apesar de ter demonstrado alguns comportamentos inesperados nos testes específicos, o que pode ser explicado pela aleatoriedade do ambiente e do processo de treinamento e pelo espaço de estados ser enorme, sem a possibilidade de ser completamente mapeável.

Por fim, apesar de a alocação ter sido feita de maneira inteligente e de ter conseguido controlar as taxas de *outage*, ambos os algoritmos sofreram com o aumento da dimensionalidade do problema. A abordagem realizada para distribuição de potência das comunicações D2D ao longo dos RB foi efetiva em aumentar as possibilidades de alocação, mas aumentou a dimensionalidade de maneira significativa, afetando o desempenho da rede em situações com grande número de pares D2D e/ou de um sistema com muitos RBs disponíveis.

6 Conclusão

Este trabalho propôs dois algoritmos de alocação de recursos para sistemas móveis com comunicações D2D *in-band underlay* utilizando técnicas de Aprendizado por Reforço Profundo (*Deep Reinforcement Learning* - DRL), mais especificamente o DDPG e o TD3.

Os alocadores produzidos convergiram e conseguiram controlar bem as taxas de *outage* do sistema, se comparado com uma alocação aleatória. Entretanto, em situações em que a dimensionalidade do problema aumentou, isto é, em que houve muitos pares D2D e/ou muitos RBs disponíveis, o desempenho do algoritmo foi afetado consideravelmente, confirmando a dificuldade que as técnicas de DRL têm em atuar em problemas com ações de dimensionalidade alta, conforme discutido no Capítulo 1.

Ainda assim, o algoritmo foi capaz de alcançar altas taxas de transmissão e bom controle da QoS das comunicações prioritárias, ao fazer a alocação de recursos completa, isto é, o controle de potências e a alocação de espectro, levando em consideração as posições dos dispositivos.

Por fim, o modelo mostrou boa estabilidade, fazendo a alocação de maneira coerente, sem grande variabilidade entre as alocações realizadas em eventos subsequentes, o que poderia causar capacidades de comunicações extremamente variáveis e pouco confiáveis para os dispositivos do sistema.

6.1 Sugestões e trabalhos futuros

Com a tendência de inserção de técnicas de *Machine Learning* (ML) em sistemas de comunicações, os algoritmos de DRL para alocação são candidatos promissores para realizar a alocação de recursos. Assim, para trabalhos futuros, sugere-se:

- Utilizar outros algoritmos de DRL que venham mostrando bom desempenho em diferentes problemas, como o *Distributional Deep Deterministic Policy Gradient* (D4PG) e o *Soft Actor-Critic* (SAC);

- Utilizar técnicas que prevejam as mudanças que o canal sofrerá nos próximos eventos, para que a alocação seja feita levando em consideração não só o estado presente, como as expectativas para o futuro próximo;
- Utilizar técnicas que diminuam a dimensionalidade do problema antes de passar pelo algoritmo, como um sistema de alocação de espectro inicial que divida as comunicações em *clusters* a partir de suas posições e defina alguns RBs que podem ser utilizados por cada *clusters*, e, a partir de então, defina as potências de cada comunicação;
- Modelar o canal de outras formas, inserindo o desvanecimento de pequena escala e explorando outras possibilidades previstas para o 5G, como células de ambientes rurais e suburbanas;
- Modelar a locomoção dos dispositivos de maneira mais realista, neste problema, os dispositivos se locomoviam com velocidades fixas relativamente altas através de qualquer ponto do ambiente. Em sistemas reais, é provável que dispositivos em alta velocidade estejam restritos a rodovias, enquanto os outros estejam em velocidade baixa ou parados em qualquer ponto da célula;
- Aprofundar as técnicas de *Curriculum Learning*, desenvolvendo estratégias mais inteligentes para ensinar os algoritmos de maneira gradativa, objetivando melhorar o processo de aprendizagem do modelo;
- Buscar modelos que sejam mais escalonáveis, o que pode, talvez, ser solucionado usando RNNs como as redes neurais dos agentes, já que algumas dessas redes, como a LSTM, não são dependentes de entradas com tamanho fixo;
- Comparar os resultados obtidos com algoritmos presentes na literatura, como os algoritmos do tipo *greedy* ou do tipo *local search*;
- Considerar sistemas de comunicação com múltiplas células;
- Investigar a integração com a nuvem e/ou com servidores FOG;
- Buscar plataformas e *frameworks* que possam ser utilizadas para colocar o algoritmo em operação;
- Fazer a alocação visando garantir os requisitos mínimos de qualidade para diferentes serviços.

Referências

[1] Phongtraychack, Anachack e Dolgaya, Darya: *Evolution of mobile applications.* MATEC Web Conf., 155:01027, 2018. 1

[2] Guarda, Teresa, Maria Fernanda Augusto, Isabel Lopes, José Avelino Victor, Álvaro Rocha e Lilian Molina: *Mobile communication systems: Evolution and security.* Em *Developments and Advances in Defense and Security*, páginas 87–94, Singapore, 2020. Springer Singapore, ISBN 978-981-13-9155-2. 1

[3] Luong, Nguyen Cong, Xiao Lu, Dinh Thai Hoang, Dusit Tao Niyato e Dong In Kim: *Radio resource management in joint radar and communication: A comprehensive survey.* IEEE Communications Surveys & Tutorials, 23:780–814, 2021. 1

[4] Chen, Wuhui, Xiaoyu Qiu, Ting Cai, Hong Ning Dai, Zibin Zheng e Yan Zhang: *Deep reinforcement learning for internet of things: A comprehensive survey.* IEEE Communications Surveys Tutorials, 23(3):1659–1692, 2021. 1

[5] Song, Sooeun, Jaewook Jung, Minsu Choi, Changsung Lee, Jungkyu Sun e Jong Moon Chung: *Multipath based adaptive concurrent transfer for real-time video streaming over 5g multi-rat systems.* IEEE Access, 7:146470–146479, 2019. 1

[6] Kar, Udit e Debarshi Sanyal: *A critical review of 3gpp standardization of device-to-device communication in cellular networks.* SN Computer Science, 1, outubro 2019. 1, 2, 24, 25, 26

[7] Ali, Sher e Ayaz Ahmad: *Resource allocation, interference management, and mode selection in device-to-device communication: A survey.* Transactions on Emerging Telecommunications Technologies, 28(7):e3148, 2017. e3148 ett.3148. 2, 3

[8] Phunchongharn, Phond, Ekram Hossain e Dong In Kim: *Resource allocation for device-to-device communications underlaying lte-advanced networks.* IEEE Wireless Communications, 20(4):91–100, 2013. 2

[9] Hossen, Md Sakhawat, Md Yeakub Hassan, Faisal Hussain, Salimur Choudhury e Muhammad Mahbub Alam: *Relax online resource allocation algorithms for d2d communication.* International Journal of Communication Systems, 31(10):e3555. e3555 dac.3555. 2, 3

[10] Zulhasnine, Mohammad, Changcheng Huang e Anand Srinivasan: *Efficient resource allocation for device-to-device communication underlaying lte network.* Em *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, páginas 368–375, 2010. 3

[11] Islam, Mohammad Tauhidul, Abd Elhamid M. Taha, Selim Akl e Salimur Choudhury: *A local search algorithm for resource allocation for underlaying device-to-device communications.* Em *2015 IEEE Global Communications Conference (GLOBECOM)*, páginas 1–6, 2015. 3

[12] Islam, Mohammad Tauhidul, Abd Elhamid M. Taha, Selim Akl e Mervat Abu-Elkheir: *A stable matching algorithm for resource allocation for underlaying device-to-device communications.* Em *2016 IEEE International Conference on Communications (ICC)*, páginas 1–6, 2016. 3

[13] Zhang, Rongqing, Xiang Cheng, Liuqing Yang e Bingli Jiao: *Interference graph-based resource allocation (ingra) for d2d communications underlaying cellular networks.* IEEE Transactions on Vehicular Technology, 64(8):3844–3850, 2015. 3

[14] Sharma, Sandeepika e Brahmjit Singh: *Weighted cooperative reinforcement learning-based energy-efficient autonomous resource selection strategy for underlay d2d communication.* IET Commun., 13:2078–2087, 2019. 3

[15] Goodfellow, Ian, Bengio Yoshua e Courville Aaron: *Deep Learning.* MIT Press, 2016. 3, 7, 8, 9, 15

[16] Luong, Nguyen Cong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying Chang Liang e Dong In Kim: *Applications of deep reinforcement learning in communications and networking: A survey.* IEEE Communications Surveys Tutorials, 21(4):3133–3174, 2019. 3

[17] Li, Kaiwen, Tao Zhang e Rui Wang: *Deep reinforcement learning for multi-objective optimization.* junho 2019. 4

[18] Abels, Axel, Diederik Roijers, Tom Lenaerts, Ann Nowé e Denis Steckelmacher: *Dynamic weights in multi-objective deep reinforcement learning.* Em Chaudhuri, Kamalika e Ruslan Salakhutdinov (editores): *Proceedings of the 36th International Conference on Machine Learning*, volume 97 de *Proceedings of Machine Learning Research*, páginas 11–20. PMLR, 09–15 Jun 2019. 4

[19] Huang, Jingfei, Yang Yang, Gang He, Yang Xiao e Jun Liu: *Deep reinforcement learning-based dynamic spectrum access for d2d communication underlay cellular networks.* IEEE Communications Letters, 25(8):2614–2618, 2021. 4

[20] Li, Zheng, Caili Guo e Yidi Xuan: *A multi-agent deep reinforcement learning based spectrum allocation framework for d2d communications.* Em *2019 IEEE Global Communications Conference (GLOBECOM)*, páginas 1–6, 2019. 4

[21] Meng, Fan, Peng Chen, Lenan Wu e Julian Cheng: *Power allocation in multi-user cellular networks: Deep reinforcement learning approaches.* IEEE Transactions on Wireless Communications, 19(10):6255–6267, 2020. 4, 5

[22] Tan, Junjie, Ying Chang Liang, Lin Zhang e Gang Feng: *Deep reinforcement learning for joint channel selection and power control in d2d networks.* IEEE Transactions on Wireless Communications, 20(2):1363–1378, 2021. 4

[23] Wang, Dan, Hao Qin, Bin Song, Ke Xu, Xiaojiang Du e Mohsen Guizani: *Joint resource allocation and power control for d2d communication with deep reinforcement learning in mcc*. Physical Communication, 45:101262, 2021, ISSN 1874-4907. 4

[24] Ravichandiran, Sudharsan: *Deep reinforcement learning with python : master classic rl, deep rl, distributional rl, inverse rl, and more with openai gym and tensorflow / sudharsan ravichandiran.*, 2020, ISBN 1-83921-559-3. 4, 15

[25] Fukuzawa, Megumi, Naomi Hayashi: *Comparison of 3 different reinforcements of learning in dogs (canis familiaris)*. Journal of Veterinary Behavior, 8(4):221–224, 2013. 7

[26] Ji, Hongjing, Osama Alfarraj e Amr Tolba: *Artificial intelligence-empowered edge of vehicles: Architecture, enabling technologies, and applications*. IEEE Access, 8:61020–61034, 2020. 8, 10

[27] Williams, Ronald: *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. Machine Learning, 8:229–256, 2004. 10

[28] Sewak, Mohit: *Deep Reinforcement Learning*. Springer Singapore, 2019. 10, 12, 13, 17, 18

[29] Mousavi, Seyed Sajad, Michael Schukat e Enda Howley: *Deep reinforcement learning: an overview*. Em *Proceedings of SAI Intelligent Systems Conference*, páginas 426–440. Springer, 2016. 10, 11

[30] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra e Martin A. Riedmiller: *Playing atari with deep reinforcement learning*. ArXiv, abs/1312.5602, 2013. 13, 17

[31] Lillicrap, Timothy, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver e Daan Wierstra: *Continuous control with deep reinforcement learning*. CoRR, setembro 2015. 14

[32] Fujimoto, Scott, Herke van Hoof e David Meger: *Addressing function approximation error in actor-critic methods*. Em *ICML*, páginas 1582–1591, 2018. 14, 44

[33] Hasselt, H. V., Arthur Guez e David Silver: *Deep reinforcement learning with double q-learning*. ArXiv, abs/1509.06461, 2016. 14

[34] Haarnoja, Tuomas, Aurick Zhou, P. Abbeel e Sergey Levine: *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*. Em *ICML*, 2018. 14

[35] Kalyanakrishnan, Shivaram e Peter Stone: *Batch reinforcement learning in a complex domain*. New York, NY, USA, 2007. Association for Computing Machinery, ISBN 9788190426275. 17

[36] Nauta, Johannes, Yara Khaluf e Pieter Simoens: *Using the ornstein-uhlenbeck process for random exploration*. abril 2019. 17, 18

[37] Plappert, Matthias, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, P. Abbeel e Marcin Andrychowicz: *Parameter space noise for exploration.* ArXiv, abs/1706.01905, 2018. 18

[38] *Guidelines for evaluation of radio interface technologies for imt-advanced.* 2008. 32

[39] Sobhi-Givi, Sima, Azadeh Khazali, Hashem Kalbkhani, Mahrokh G. Shayesteh e Vahid Solouk: *Resource allocation and power control for underlay device-to-device communication in fractional frequency reuse cellular networks.* Telecommunication Systems: Modelling, Analysis, Design and Management, 65(4):677–697, August 2017. 32

[40] Du, Mao, Lin Yang e Jiayu Tu: *A novel approach to calculate the spatial–temporal correlation for traffic flow based on the structure of urban road networks and traffic dynamic theory.* Sensors, 21(14), 2021, ISSN 1424-8220. 32

[41] Kang, Jikun, Miao Liu, Abhinav Gupta, Chris Pal, Xue Liu e Jie Fu: *Learning multi-objective curricula for deep reinforcement learning.* 2021. 37