UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

Filipe Samuel da Silva

Otimização de trajetória de nós MEC-UAVs voltados para processamento computacional em redes 5G

Recife

2024

Filipe Samuel da Silva

Otimização de trajetória de nós MEC-UAVs voltados para processamento computacional em redes 5G

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Área de Concentração: Redes e sistemas distribuídos

Orientador (a): Prof. Dr. Andson Marreiros Balieiro

Recife

2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Silva, Filipe Samuel da.
Otimização de trajetória de nós MEC-UAVs voltados para processamento computacional em redes 5G / Filipe Samuel da Silva. - Recife, 2024.
48 p : il., tab.

Orientador(a): Andson Marreiros Balieiro
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Engenharia da Computação - Bacharelado, 2024.
Inclui referências.

1. Computação na borda multi acesso. 2. Descarregamento Computacional. 3. Veículo aéreo não tripulado. 4. Aprendizado por Reforço. 5. Otimização de trajetória do UAV. I. Marreiros Balieiro, Andson. (Orientação). II. Título.

000 CDD (22.ed.)

FILIPE SAMUEL DA SILVA

Otimização de trajetória de nós MEC-UAVs voltados para processamento computacional em redes 5G

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em Engenharia da Computação.

Aprovado em: 18/10/2024

BANCA EXAMINADORA

_____

Prof. Dr. Andson Marreiros Balieiro (Orientador)

Universidade Federal de Pernambuco

_____

Prof. Dr. Jamilson Ramalho Dantas (Examinador Interno)

Universidade Federal de Pernambuco

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me dado a vida e me sustentado até aqui. Agradeço a minha mãe Edileide e ao meu pai Josenaldo, pelo muito carinho e pelo apoio sempre que preciso. Aos meus irmãos, Lucas e Silas, que estão sempre ao meu lado, apoiando e ajudando.

Agradeço ao meu professor e orientador, Andson, que muito me ajudou em minha pesquisa e na condução do trabalho desenvolvido. Agradeço a todos os amigos e familiares que nunca mediram esforços para estar ao nosso lado.

Agradeço também a Universidade Federal de Pernambuco e ao Centro de Informática, que são excelentes ambientes para o desenvolvimento profissional e pessoal.

RESUMO

O offloading de tarefas computacionais é um recurso promissor para as redes móveis 5G, que possuem uma variedade de aplicações e serviços que demandam diferentes níveis de capacidade dos dispositivos. Ele se utiliza de infraestruturas de computação na borda da rede para processar as requisições de usuários de dispositivos móveis, que particionam e paralelizam o processamento das aplicações. Dessa forma, garante-se a economia de recursos de energia dos usuários e uma maior capacidade de processamento. Contudo, as infraestruturas tradicionais de computação na borda multi-acesso (MEC) possuem limitações em cenários dinâmicos e apresentam um alto custo de implementação. Por isso, tem sido considerado o uso de nós de computação na rede através de veículos aéreos não tripulados (UAVs). Porém, existem desafios quanto à habilitação de sistemas de computação na borda via UAVs. Como os UAVs são dispositivos orientados a bateria, é necessário otimizar a movimentação deles a fim de evitar descarregamento precoce e assim estender a sua autonomia antes da recarga. Neste sentido, este trabalho apresenta uma solução baseada em aprendizagem por reforço para o problema de otimização de trajetória do MEC-UAV que fornece o serviço de processamento de tarefas aos usuários. A solução foi avaliada em termos de custo de energia, atraso e capacidade de processamento em comparação a outras soluções da literatura. Os Resultados mostraram que a solução proposta obteve taxa de atendimento superior quando comparada a solução baseada em MLP e uma maior porcentagem de tarefas respondidas comparada a política de movimentação por agrupamento de usuários.

Palavras-chaves: Computação na borda multi acesso, Descarregamento Computacional, Veículo aéreo não tripulado, Aprendizado por Reforço, Otimização de trajetória do UAV.

ABSTRACT

Offloading of computational tasks is a promising resource for 5G mobile networks, which have a variety of applications and services that demand different levels of device capacity. It uses computing infrastructures at the edge of the network to process requests from mobile device users, which partition and parallelize the processing of applications. In this way, energy resource savings for users and greater processing capacity are guaranteed. However, traditional multi-access edge computing (MEC) infrastructures have limitations in dynamic scenarios and are expensive to implement. Therefore, the use of computing nodes in the network through unmanned aerial vehicles (UAVs) has been considered. However, there are challenges in enabling edge computing systems via UAVs. Since UAVs are battery-driven devices, it is necessary to optimize their movement in order to avoid premature discharge and thus extend their autonomy before recharging. In this sense, this work presents a solution based on reinforcement learning for the trajectory optimization problem of the MEC-UAV that provides the task processing service to users. The solution was evaluated in terms of energy cost, delay and processing capacity compared to other solutions in the literature. The results showed that the proposed solution obtained a higher service rate when compared to the MLP-based solution and a higher percentage of answered tasks compared to the movement policy based on user grouping.

Keywords: Multi-Access Edge Computing, Computing offloading, Unmanned Aerial Vehicle, Reinforcement Learning, UAV trajectory optimization.

LISTA DE FIGURAS

LISTA DE CÓDIGOS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Quinta Geração (5G) de Redes Móveis é a geração mais recente de tecnologia de redes móveis e sistemas de comunicação. Ela traz novidades em relação a sua infraestrutura, modelo de funcionamento, aplicações e serviços suportados (AL-FALAHY; ALANI, 2017). Entre as aplicações propostas, está a utilização de sistemas de computação em nuvem para prover maior poder de processamento aos usuários de dispositivos móveis, viabilizando aplicações de reconhecimento e detecção biométrica, realidade virtual e aumentada, modelos de deteção de objetos e até integração e auxílio no controle de carros autônomos.

Os serviços das redes 5G são separados em 3 grandes grupos: Banda larga móvel aprimorada (eMBB), Comunicação do tipo de máquina massiva (mMTC) e Comunicação de baixa latência ultra confiável (URLLC) (WIJETHILAKA; LIYANAGE, 2021). O eMBB está relacionado ao fornecimento de maior largura de banda para conexões com altas taxas de transferência de dados, como em aplicações de streaming de vídeo. O mMTC se refere a integração de muitos e diversos dispositivos de Internet das Coisas (IoT) que vão transferir informações e comunicação para sistemas, por exemplo em fábricas, agricultura ou cidades inteligentes. Já o URLLC está relacionado a garantia de baixíssima latência de comunicação e alta confiabilidade na entrega dos pacotes entre os dispositivos da rede. Para garantir a conectividade e boa experiência do usuário no uso de aplicações multimídia é imprescindível fornecer um bom serviço de baixa latência. Isso se torna ainda mais importante no cenário de aplicações críticas, que necessitam de um menor tempo de resposta e garantia de baixa latência sem perda de pacotes.

Para que seja possível a integração dessas aplicações, foram criadas infraestruturas de computação, que, diferentemente da infraestrutura tradicional de computação em nuvem, estão localizadas mais próximas ao usuário final. Essas infraestruturas são chamadas de Computação de Borda de Multi-acesso (MEC). Elas representam nós de computação e armazenamento posicionados mais próximos ao usuário final (AHMED; REHMANI, 2017). Com isso, eles oferecem além de capacidade computacional, tempo de resposta reduzido da aplicação do usuários e diminui o tráfego direcionado a nuvem central ou outras redes externas, mitigando o congestionamento na rede. Porém, para instalação de nós de MEC, há a necessidade de um espaço fixo para instalação e manutenção, o que pode não ser muito resistente a situações de exceção, como situações de desastre ou calamidade pública. Além disso, para algumas regiões

ou lugares pode não ser financeiramente interessante a sua implementação, tendo em vista seu custo em relação ao faturamento, ou no casos de período de uso momentâneo, como em shows e jogos esportivos.

Uma alternativa para solucionar esses casos é a utilização de dispositivos aéreos não tripulados (UAVs), também conhecidos como drones, para fornecer uma infraestrutura móvel de computação de boda multi-acesso (MEC-UAV). Também conhecidos como MEC assistido por UAV, eles são muito úteis em ambientes onde não existe uma infraestrutura terrestre disponível e particularmente úteis nas situações em que os sistemas MEC terrestres convencionais são destruídos por eventos naturais (ZHOU et al., 2020). Também existem outros cenários como: shows e eventos de grande porte, locais de difícil acesso terrestre ou de preservação, ambientes onde existem picos de acesso ou congestionamentos temporalmente limitados. Em geral, o uso de MEC-UAV está muito relacionado a ambientes dinâmicos, onde não é viável ou interessante o aprovisionamento de uma infraestrutura terrestre.

Para as chamadas redes 5G Avançadas e a próxima geração de redes móveis, a sexta (6G), existe a iniciativa de integração da infraestrutura terrestre com a aérea, visando uma conectividade tridimensional, que envolve: estações rádio base terrestres (BSs), satélites e dispositivos aéreos, como UAVs, balões e aeronaves. Quando embarcados com recursos de computação, eles ampliam a área de cobertura e oferecem melhor poder de computação. Os usuários de dispositivos móveis podem se utilizar desses sistemas para melhorar o seu poder de processamento e garantir mais autonomia de bateria dos seus dispositivos a partir do descarregamento (Offloading) de tarefas computacionais para serem processadas nesses nós remotos.

No processo de offloading, algumas tarefas podem ser executadas no dispositivo e outras enviadas para serem processadas remotamente. Isso é feito para otimizar o tempo de execução de alguma aplicação, dado o maior poder computacional dos nós remotos, o consumo de bateria do dipositivo móvel do usuário e até mesmo a execução paralela de determinadas aplicações. Isso permite o particionamento da aplicação e execução distribuída (LIN et al., 2019). Assim o usuário pode acessar serviços e aplicações com qualidade melhor, que seria inviável caso fossem todos processsados localmente no dispositivo do usuário.

Porém existem desafios quanto a habilitação de sistemas de computação na borda via UAVs. É previsto que as redes MEC habilitadas por UAV serão amplamente implantadas quando o custo dos UAVs diminuir o suficiente (ZHOU et al., 2020). Além disso, como são dispositivos orientados a bateria, é necessário otimizar a movimentação deles a fim de evitar

descarregamento precoce e assim estender a sua autonomia antes da recarga. Para fornecer o serviço de offloading computacional, o UAV precisa estar geográficamente próximo ao usuário que irá realizar a transmissão dos dados para execução da tarefa. Além disso, ele deve ser capaz de ajustar suas localizações para realizar o descarregamento em tempo real do resultado do seu processamento das tarefas dos usuários (XU et al., 2021), de modo que o ciclo de processamento e retorno do resultado para o dispositivo do usuário seja concluído dentro dos limites desejados. Nesse sentido, o UAV possui a necessidade de encontrar os melhores caminhos em seu percurso, para garantir o atendimento da maior quantidade de usuários e otimizar a taxa tarefas concluidas com sucesso, onde o retorno do resultado acontece dentro do definido.

Muitos trabalhos na literatura sobre otimização da trajetória do UAV consideram principalmente latência e o custo de energia do descarregamento dos dados da tarefa do usuário móvel para os nós remotos e processamento nos nós remotos. Como o trabalho de Zhang et al. (2021), que considera a latência, o consumo de energia e o número de tarefas completadas. O trabalho de Zhan et al. (2020) considera o tempo de conclusão e o consumo de energia. Enquanto o trabalho de Zhang et al. (2019) considera o consumo de energia do UAV e dos dispositivos móveis e a latência em relação a alocação de recursos de computação. Em (HORTELANO et al., 2023) é feita uma comparação entre diversos artigos sobre o que os trabalhos consideram na definição do ambiente de interação. Poucos consideram em conjunto, o retorno dos dados para o dispositivo móvel do usuário e a variação no tamanho das tarefas (tempo de processamento). Neste sentido, este trabalho preenche estas lacunas e endereça o problema de otimização de trajetória de nós MEC-UAVs voltados para o offloading computacional. Para lidar com a dinamicidade do ambiente, uma solução baseada em Aprendizado por Reforço é desenvolvida, onde os UAVs podem ajustar suas operações, respondendo rapidamente as mudanças no ambiente (DU et al., 2023). A solução busca indicar a melhor ação de movimentação a ser tomada pelo MEC-UAV, com o objetivo de maximizar a quantidade de requisições de offloading atendidas, reduzir a perda de tarefas e o consumo de energia, dada as as posições dos usuários, energia do UAV e a sua capacidade de processamento.

1.1 OBJETIVOS

O objetivo geral deste trabalho é desenvolver uma solução baseada em aprendizado por reforço para o problema de otimização da trajetória do MEC-UAV em um cenário de offloading

de tarefas computacionais. Desta forma, busca-se que a solução permita uma maior autonomia dos usuários clientes da rede 5G, que estão se utilizando da infraestrutura MEC-UAV. Para alcançar este objetivo, os seguintes objetivos específicos foram definidos.

- Descrever/representar o ambiente MEC-UAV para offloading computacional, considerando o comportamento dos usuários e as tarefas para processamento com diferentes localizações e tamanhos, respectivamente.
- Desenvolver um modelo de rede neural, que, a partir do treinamento em aprendizado por reforço, possa interagir com o ambiente, determinando a próxima posição do UAV que irá atender as requisições dos usuários e assim aprender a melhor trajetória.
- Formular as métricas para cálculo de custo de energia e de movimentação, taxa de atendimento a usuários, de processamento das tarefas e quantidade de tarefas processadas. Definir as variáveis do ambiente em relação a usuários e requisições e dos parâmetros de treinamento da rede.
- Analisar a solucão em comparação a outras solucões da literatura, considerando metricas como parcela de usuários atendidos, o custo de energia acumulado para diferentes cenários de movimentação.

1.2 ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado da seguinte forma. O Capítulo 2 apresenta os conceitos-chave para o entendimento deste trabalho. Ele aborda as redes 5G, suas propostas e seu estado em relação as tecnologias e a serviços de computação em nuvem; o offloading de tarefas computacionais sob o aspecto de oferecer poder computacional a dispositivos limitados; a infraesturura de computação na borda, necessária para estabelecer serviços de conectividade e baixa latência; os nós MEC-UAV como alternativa para infraestrutura de computação na borda; o uso de aprendizado por reforço como solucão para o problema de offloading de tarefas em ambiente MEC-UAVs. No capítulo 3, tem-se descrição do ambiente MEC-UAV voltado para a tarefa de processamento de tarefas enviadas pelo usuário móvel e a a formulação do problema de otimização de trajetória do nó MEC-UAV. A apresentação da solução desenvolvida, baseada em aprendizado por reforço também integra este capítulo. O Capítulo 4 traz a descrição do processo de avaliação, descrevendo os cenários e parametros adotados, assim como os

resultados obtidos em comparação com outras soluçoes da literatura. O Capítulo 5 conclui este trabalho e aponta direcões futuras de melhorias, avanços e pesquisas.

2 FUNDAMENTAÇÃO TEÓRICA

Essa seção discute os avanços e desafios em tecnologias emergentes no campo de redes de comunicação e computação. Tecnologias que possibilitam o offloading de tarefas de dispositivos móveis, usando infraestrutura de computação na borda da rede a partir de nós de computação habilitados por UAVs. Também discute uma estratégia para otimizar o controle das rotas de um UAV a partir da técnica de aprendizagem por reforço.

2.1 REDES 5G

As redes 5G representam uma grande evolução do modelo de comunicação móvel. Elas permitem uma experiência de uso aprimorada, trazendo maior velocidade, cobertura e conectividade. Mas a sua principal vantagem está nos serviços que ela oferece. Em poucas palavras, a arquitetura de rede 5G é projetada para suportar três classes principais de serviços fundamentais: Banda larga móvel aprimorada (enhanced Mobile BroadBand - eMBB), Comunicação do tipo de máquina massiva (massive Machine Type Communications - mMTC) e Comunicação de baixa latência ultraconfiável (Ultra-Reliable Low-Latency Communications - URLLC) (WIJETHILAKA; LIYANAGE, 2021).

O serviço de eMBB é focado em fornecer alta velocidade de transmissão de dados e capacidade da rede, para aplicações que exigem alta largura de banda, como transmissões ao vivo, realidade virtual, aplicações de reconhecimento inteligente e em contexto. O objetivo do serviço eMBB é maximizar a taxa de dados, ao mesmo tempo em que garante uma confiabilidade moderada, com taxa de erro de pacote (PER) na ordem de 10^{-3} (POPOVSKI et al., 2018). Ele endereça aplicações centradas no ser humano, para um acesso com alta taxa de dados a serviços móveis, conteúdo multimídia e dados (JIANG et al., 2021).

Enquanto o eMBB é uma extensão do serviço LTE-Avançado cujo objetivo é maximizar a taxa de pico de dados, o mMTC é projetado para suportar um grande número de dispositivos de Internet das Coisas (IoT) enviando pequenos dados esporadicamente apenas durante a fase ativa (ALSENWI et al., 2019). Um dispositivo mMTC fica ativo de maneira intermitente e usa uma taxa de transmissão fixa, normalmente baixa, no uplink (POPOVSKI et al., 2018). O mMTC busca integrar e conectar dispositivos IoT, sensores 'em qualquer lugar'; Redução no uso de energia em quase 90%; Alta duração de bateria; Já o URLLC fornece comunicação de

ultraconfiabilidade com taxas de erro de pacote baixíssimas e baixas latências fim a fim (E2E) para dar suporte a aplicações 5G, como assistência médica e cidade inteligente (FOURATI; MAALOUL; CHAARI, 2021). No 5G, o requisito mínimo para latência do plano do usuário é de 4 ms para eMBB e 1 ms para URLLC (JIANG et al., 2021) Isso traz grandes necessidades de integração de tecnologias emergentes e novas estratégias para atingir esses requisitos.

Todos esses esforços, tem objetivo de atingir um alto nível de integração entre sistemas, aplicações em tempo real e boa experiência de uso. Dispositivos vestíveis inteligentes (por exemplo: pulseiras, relógios, óculos), eletrodomésticos domésticos inteligentes (por exemplo, televisores, geladeiras, termostatos), sensores, carros autônomos e objetos móveis cognitivos (por exemplo: robôs, drones) prometem um mundo inteligente hiper conectado que pode inaugurar muitas oportunidades interessantes em muitos setores da vida como saúde, agricultura, transporte, manufatura, logística, segurança, educação e muito mais. (AGYAPONG et al., 2014)

Várias tecnologias emergentes que são empregadas na rede de 5ª geração incluem Redes Ultra Densas (UDN - Ultra-Dense Networks), Massive MIMO (Multiple Input Multiple Output em larga escala), Comunicação Dispositivo a Dispositivo (D2D - Device-to-Device Communication), Acesso Aleatório Espacialmente Compacto (SCA - Spatially Compact Access), Rádio Full-Duplex e Rádio Cognitivo (DOGRA; JHA; JAIN, 2021). Essas tecnologias são o caminho para a integração de todo esse universo hiperconectado.

Segundo Agyapong et al. (2014), o fatiamento de rede está se tornando uma realidade em futuras redes de telecomunicações, devido ao suporte de várias tecnologias, como Redes Definidas por Software (SDN - Software-Defined Networking), Virtualização de Funções de Rede (NFV - Network Function Virtualization) e computação em nuvem. O que permite a criação de diversas redes virtuais, garantindo segurança, confiabilidade e serviços cada vez mais especializados. A Figura 1 mostra um exemplo de como seria realizado o fatiamento da rede a partir de Blockchain.

Figura 1 – Exemplo de fatiamento da rede

Fonte: (WIJETHILAKA; LIYANAGE, 2021)

2.2 OFFLOADING COMPUTACIONAL

Offloading computacional é uma estratégia popular para poupar o uso de energia em dispositivos móveis, na qual aplicações, fazem uso de infraestruturas ricas em recurso, para transferir o uso de computação para essas infraestruturas (JYOTHIRMAI; GOPAL; SAILAJA, 2023). O principio do offloading computacional é aproveitar infraestruturas poderosas (ex: servidores remotos) para aumentar a capacidade computacional de dispositivos menos potentes (ex: Dispositivos móveis) (LIN et al., 2019). Normalmente essas infraestruturas externas possuem um poder computacional muito maior e não há limitação de uso de bateria ou energia durante esse processamento. Tornando-se uma boa estratégia para preservar a autonomia do usuário em sua experiência de uso.

No offloading computacional, um dispositivo móvel envia(transfere), parte de sua computação para a Cloud. Este processo envolve o particionamento da aplicação, a decisão do descarregamento(Offloading) e a execução distribuida de tarefas (LIN et al., 2019). Com a alta carga de processamento que muitas aplicações vem precisando, como o uso em aplicações de Inteligência Artificial, Reconhecimento Facial, Processamento de linguagem natural, Processamento de Dados, Realidade Virtual, Realidade Aumentada, Processamento de Imagens e Jogos. Esses tipos de aplicações móveis são tipicamente consumidoras de recursos, exigingo computação intensiva e alto consumo de energia (CHEN et al., 2016). Existe um aumento con-

siderável da dependência desses sistemas para garantir uma boa experiência a um cliente com um dispositivo que possui limitações de hardware e um consumo razoável de bateria.

Esse processo de offloading é diferente da arquitetura tradicional cliente-servidor, onde todo o processamento é feito no servidor. Diferente também do modelo de migração de processamento, presente em sistemas de multiprocessamento e grid computing. A principal diferença é que o descarregamento computacional migra programas, para servidores fora do "ambiente de computação imediato"dos usuários (KUMAR et al., 2013).

De acordo com Jyothirmai, Gopal e Sailaja (2023), os principais critérios para o offloading envolvem: Métricas; Particionamento da aplicação; Balanceamento de carga; Trade-off entre energia-consumo; Disponibiliade; Segurança; Tomada de decisão; Flexibilidade; Utilização de Recursos. Tudo isso torna o processo de decisão muito variável a depender do ambiente e as condições de cada dispositivo e de cada modelo e forma da infraestrurura.

É possível a utilização de serviço de offloading a partir de infraestrutura de cloud pública, como Amazon EC2, Windows Azure. Porém esse tipo de abordagem pode trazer para a experiência do usuário, alta latência. Para resolver isso, foram propostos modelos baseados em cloudlets (SATYANARAYANAN et al., 2009) para computação em nuvem, reduzindo o atraso do offloading. Segundo Chen et al. (2016) as duas grandes desvantagens para a computação em nuvem móvel baseada em cloudlet são:

1) devido à cobertura limitada de redes WiFi (normalmente disponíveis para ambientes internos), a computação em nuvem móvel baseada em cloudlet não pode garantir o fornecimento de serviços onipresentes em todos os lugares;

2) devido à restrição de espaço, a computação em nuvem móvel baseada em cloudlet geralmente utiliza um servidor/cluster de computação com recursos de computação pequenos/médios, o que pode não satisfazer a QoS(critério de qualidade de serviço) de um grande número de usuários.

2.2.1 Infraestrutura de computação na borda

Mobile Cloud Computing (MCC) é a integração de Cloud Computing e Mobile Computing, na qual dispositivos móveis realizam o offloading de computação, se aproveitando do poder da nuvem para agilizar a execução da aplicação e economizar no consumo de energia (LIN et al., 2019). O uso de Mobile Cloud Computing permite o offloading de operações que demandam recursos de dispositivos com recursos limitados para máquinas poderosas na nuvem. Ao aplicar

tal abordagem, as execuções dos aplicativos podem se beneficiar de recursos extras (SILVA et al., 2018). Porém, segundo Dinh et al. (2018) MCC tem uma desvantagem crítica: Os servidores de nuvem, são, usualmente, logicamente e espacialmente distantes dos usuários móveis. O que leva a enormes latências de comunicação. O MCC sofre desvantagens consideráveis, por exemplo, baixa escalabilidade, alta latência, problemas de privacidade e segurança e carga extrema sobre largura de banda limitada (PHAM et al., 2020).

Para atacar esses problemas de MCC, temos a infraestrutura de computação na borda da rede, Computação Multi acesso na borda (Multi-access Edge Computing - MEC), que oferece aos dispositivos acesso a recursos computacionais próximos ao usuário final. Dispositivos terminais móveis podem acessar o servidor de computação de borda para melhorar o poder de computação de tarefas móveis, como análise e processamento de dados (LI; JIANG, 2020). Comparado com o MCC, o MEC tem as vantagens de atingir menor latência, economizar energia para dispositivos móveis, oferecer suporte à computação com reconhecimento de contexto e aumentar a privacidade e a segurança para aplicativos móveis (MAO et al., 2017). Este, surge como alternativa ao modelo tradicional de infraestrutura de computação em nuvem móvel, como Amazon e Google Cloud, que para determinados tipos de aplicações, possui alta latência. Isso é especialmente indesejável para aplicativos móveis nos quais um tempo de resposta imediato é crítico para os usuários, como aplicativos de aumento de realidade e sistemas de jogos multijogador móveis (WANG et al., 2019) Em constraste, MEC tem o potencial para significativamente reduzir a latência, evitar o congestionamento e prolongar o tempo de vida da bateria de dispositivos móveis, ao realizar o offloading de tarefas de computação do dispositivo móvel para um servidor MEC fisicamente próximo (MAO; ZHANG; LETAIEF, 2016).

Ao fornecer capacidades de armazenamento e computação em nós na borda, a MEC é prevista para fornecer serviços de computação em nuvem na borda para redes sem fio com baixa latência (DINH et al., 2018). Especificamente, tarefas intensivas em computação podem ser transferidas de dispositivos IoT, para dispositivos de borda, para reduzir seu consumo de energia(MAO et al., 2017). Ao dotar redes de acesso de rádio ubíquas com poderosos recursos de computação, a computação em nuvem móvel na borda foi concebida para fornecer serviços de aumento da computação, ágeis e abrangentes para usuários de dispositivos móveis a qualquer hora e em qualquer lugar (CHEN et al., 2016). Isso permite que o usuário final possa decidir qual o momento ideal para se utilizar dessa infraestrutura e tomar decisões com objetivo de otimizar sua experiência de uso.

Em comparação, com distâncias de propagação curtas e protocolos simples, o MEC tem

o potencial de realizar latência de nível tátil para aplicativos críticos a latência em aplicações 5G (MAO et al., 2017). Não há necessidade de transferir tarefas de computação para o centro de nuvem, mas diretamente de usuários móveis para nós de computação de borda, o que pode reduzir muito o uso de recursos de latência e largura de banda (LI; JIANG, 2020) Existe, porém um desafio na implementação de servidores na borda. Os locais dos servidores de borda, são críticos para os atrasos de acesso de usuários móveis e a utilização de recursos desses servidores, especialmente em cidades inteligentes que incluem várias centenas ou milhares de estações base por meio das quais os usuários móveis acessam os servidores de borda (WANG et al., 2019).

2.3 NÓS MEC-UAV

Os Nós de computação na borda de multi-acesso (MEC), possuem boas vantagens para o processamento de tarefas e descarregamento de tarefas de usuários que possuem dispositivos com limitações de bateria ou de poder de processamento. Apesar desse potencial, os servidores de ponta (ES) de localização fixa são frequentemente associados a altos custos e escalabilidade limitada, o que restringe sua aplicabilidade em cenários complexos ou urgentes (WANG et al., 2024). Nesses cenários se faz necessário o uso de alternativas que possibilitem flexibilidade(mobilidade) e adaptação ao ambiente. Os veículos aéreos não tripulados (UAVs), que têm caraterísticas de flexibilidade de utilização e são invulneráveis a catástrofes naturais, podem ser plataformas promissoras para a realização de computação na borda (CHEN et al., 2023).

Veículos aéreos não tripulados(UAVs) são dispositivos que se movimentam de maneira autônoma e podem sobrevoar sobre regiões residenciais ou regiões de campo aberto. O uso de UAVs para fornecimento de infraestrutura de Mobile Edge Computing, foram propostos por alguns autores (JEONG; SIMEONE; KANG, 2018), (ZHOU et al., 2018), como alternativa ao modelo tradicional de MEC. Os autores Xu et al. (2021) trazem algumas características que diferenciam o uso de UAVs para auxiliar os nós MEC: Em primeiro lugar, o UAV é capaz de ajustar suas localizações de acordo com as táticas de descarregamento em tempo real dos usuários; Sua trajetória pode ser cuidadosamente planejada para propósitos específicos, como economia de energia, melhoria de rendimento; beneficiado por sua alta altitude, o MEC assistido por UAV geralmente evita efeitos geográficos, o que contribui para fortalecer e expandir a cobertura; o UAV também é menos afetado por deficiências de canal, por sua alta possibilidade

de links LoS com usuários terrestres.

Por um lado, os UAVs podem ser servidores de borda aéreos para executar computações pesadas, descarregando usuários terrestres. Por outro lado, os UAVs podem atuar como usuários aéreos e se associar a estações de base BSs terrestres para descarregar suas tarefas (PHAM et al., 2020). Nessas arquiteturas, um UAV pode ser considerado um usuário que tem tarefas de computação a serem executadas, ou um relé para auxiliar os usuários a descarregar tarefas de computação, ou um servidor MEC para executar tarefas de computação. (ZHOU et al., 2020). A controlabilidade superior, as capacidades de implantação rápida e a mobilidade excepcional dos sistemas UMEC têm capturado cada vez mais a atenção dos pesquisadores, alimentando o interesse em diversos setores na exploração de uma ampla gama de aplicações contemporâneas (WANG et al., 2024).

Usar drones como estações base aéreas é uma nova abordagem em redes sem fio para aumentar a cobertura, capacidade, eficiência energética e confiabilidade (WIJETHILAKA; LIYANAGE, 2021). Por exemplo, os UAVs equipados com várias capacidades úteis, como estações base miniaturizadas e módulos de computação embarcados, podem descarregar tarefas de computação de usuários móveis de maneira energeticamente eficiente voando perto delas (CHEN et al., 2023).

2.4 USO DE APRENDIZAGEM POR REFORÇO

O Aprendizado por Reforço (RL) é um paradigma de aprendizado de máquina, onde agentes (neste caso, UAVs) aprendem políticas de decisão através da interação com o ambiente e das recompensas recebidas. O sistema MEC assistido por UAVs é frequentemente modelado como um Processo de Decisão de Markov (MDP), onde os UAVs devem aprender a otimizar suas rotas de voo e as decisões de offloading usando interações com o ambiente (ZHANG et al., 2021). No contexto dos nós MEC-UAV, o RL pode ser aplicado para otimizar a trajetória dos UAVs, decidir quando e como realizar o offloading de tarefas e até mesmo adaptar a provisão de serviços em resposta a alterações no ambiente, como mudanças nos padrões de tráfego de dados ou condições meteorológicas.

Com o Aprendizado por Reforço, os UAVs podem aprender a ajustar suas operações em tempo real, respondendo rapidamente a mudanças no ambiente ou na demanda de rede, o que é crucial para garantir um serviço contínuo e eficiente em cenários complexos (DU et al., 2023) Nesse cenário, o objetivo pode ser otimizar o uso de recursos, minimizar a latência e

o consumo de energia, enquanto maximiza a eficiência do offloading de tarefas. Reduzir o consumo de energia e otimizar o uso dos UAVs como nós de computação, pode aumentar a vida útil da atividade do UAV e a cobertura das redes de computação na borda (ZHANG et al., 2021).

Apesar das vantagens, a aplicação do RL em ambientes dinâmicos enfrenta desafios, incluindo a necessidade de grandes volumes de dados de treino e a complexidade do ajuste de parâmetros dos modelos de RL para garantir resultados estáveis e eficientes (LUO et al., 2023). Os ambientes do sistema MEC assistido por UAV, não são totalmente observáveis para otimização dos modelos e é desafiador formular os ambientes MEC complexos para o modelo de otimização convencional (ZHANG et al., 2021).

Em geral, as distribuições de dispositivos IoT e suas demandas computacionais na rede, são heterogêneas em domínios temporais e espaciais. (DU et al., 2023) O sistema MEC assistido por UAV, exige um modelo de controle que possa maximizar o número de tarefas processadas antes de expirarem e minimizar o consumo de energia (ZHANG et al., 2021) O uso de Aprendizado por Reforço profundo DRL pode fornecer um esquema em tempo real para serviços inteligentes de comunicação e computação de UAV em um ambiente complexo e variável no tempo (CHEN et al., 2023).

3 DESCRIÇÃO DO PROBLEMA

Este capítulo define os critérios para modelagem e construção do cenário de movimentação do MEC-UAV, que fornece processamento computacional a usuários clientes da rede 5G. Ele formula o problema de otimização de trajetória do MEC-UAV,suas restricões, bem como a solução desenvolvida para atacar esse problema. Ainda são explicados alguns códigos do módulo implementado, necessários para executar a simulação.

3.1 O AMBIENTE MEC-UAV PARA OFFLOADING COMPUTACIONAL

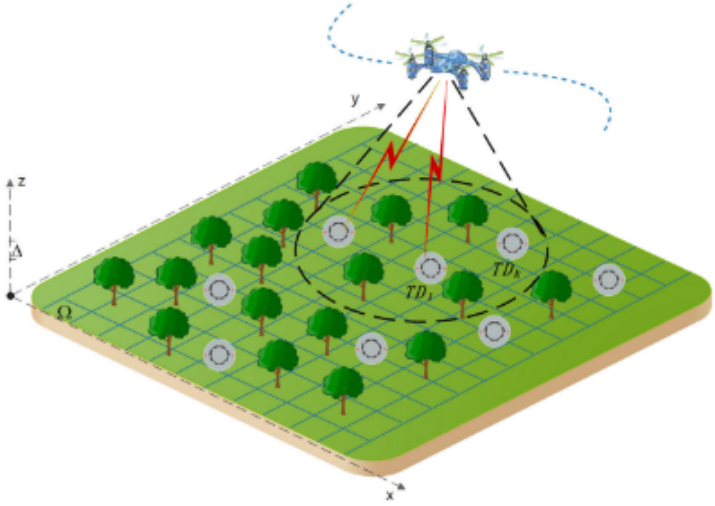
Este trabalho considera o cenário de usuários de dispositivos móveis, clientes da Rede 5G, que precisam efetuar o descarregamento de tarefas do seu dispositivo para um dispositivo na borda da rede baseado em UAV. Este por sua vez, irá executar a computação da tarefa e retornar o resultado dos dados processados aos dispositivos móveis. Como essas tarefas podem consumir considerável enegria da bateria do smartphone e, assim causar seu descarregamento mais rápido, por exemplo, a decisão de enviar uma tarefa para ser processada por um nó externo busca estender a autonomia de bateria do dispositivo do usuário e possibilite que ele consiga processar tarefas que estão além da capacidade do seu dispositivo móvel, ou seja, que ele tenha acesso a maior poder de processamento.

Como nós de computação de borda da rede, os Veículos Aéreos não tripulados (UAVs), também chamados de drones, atuam garantindo proximidade ao usuário, flexibilidade de implantação, frente a desastres naturais ou situações extremas, boa disponibilidade e poder de processamento para os dispositivos móveis. Os UAVs estabelecem uma conexão com usuários que estão na suas proximidades e garantem menor latência de comunicação. Eles também possuem comunicação com linha de visada(LoS) com os usuários, provendo melhor qualidade na transmissão em comparação às infraestruturas tradicionais (RANJHA et al., 2024). Nesse aspecto, o uso de MEC-UAV facilita o servico de comunicação de baixa latência ultraconfiável (URLLC), anunciado pela redes 5G e que estarão inclusos nas novas gerações de redes de comunicação, incluindo as redes 6G.

Os drones podem atuar nesse cenário, conforme ilustrado pela Figura 2, se movimentando e fornecendo infraestrutura, a medida que os usuários vão necessitando de serviços de computação em nuvem e vão decidindo efetuar o offloading das suas tarefas. Um UAV com MEC

tem a capacidade de processar a tarefa localmente ou transmitir essa tarefa para um outro nó da rede, até mesmo outro drone, com maior poder de processamento, onde será feita a computação da tarefa. Para o caso em que a tarefa será processada internamente, esse processamento de tarefas irá consumir energia da bateria e recursos computacionais do UAV. A movimentação ao longo do área para cobrir os usuários também consome bateria do UAV.

Figura 2 – Representação do ambiente de interação do MEC-UAV.



Fonte: (LIN et al., 2023)

O MEC-UAV se movimenta em uma região de espaço tridimensional, com os usuários usuários dispostos em uma perspectiva bidimensional. Essa é a região total onde é fornecida a infraestrutura MEC-UAV. Dentro dessa região, há subáreas menores que denotam a área de cobertura em que os usuários conseguem enviar as tarefas para processamento no MEC-UAV e obter os resultados do processamento.

Os MEC-UAVs possuem aspectos que precisam ser analisados, como a dependência de bateria, capacidade de processamento limitado e menor do que as nuvens centrais e a necessidade de controle. Ainda que autocontrolado ou controlado externamente, a movimentação para uma transição inteligente entre os espaços, tendo em vista a melhor gestão e otimização dos recursos, é um desafio.

3.2 FORMULAÇÃO DO PROBLEMA

Considerando um ambiente MEC-UAV composto de n_t usuários móveis distribuídos aleatoriamente em uma área de $A\ m^2$ e que em cada slot de tempo t, cada usuário i, localizado nas coordenadas $(x_i[t], y_i[t])$ do plano, possui a probabilidade P_{it} de ter tarefa para enviar

ao nó MEC-UAV para ser processada e que a tarefa é enviada caso o usuário i esteja dentro do raio de cobertura R do MEC-UAV, posicionado nas coordenadas $(X[t], Y[t], H)$, com H denotando a altura fixa do UAV, a distância euclidiana ($d_i[t]$) entre o nó i e o UAV no slot t pode ser calculada pela Equação 3.1. Dada a movimentação do UAV entre dois slots de tempo consecutivos t e $t-1$, o custo energético do voo do UAV da posição $(X[t-1], Y[t-1], H)$ para a $(X[t], Y[t], H)$ é dado pela Equação 3.2, que segue o descrito em (LI et al., 2022), onde P_f é a potência de voo e V é a velocidade do UAV.

$$d_i[t] = \sqrt{(X[t] - x_i[t])^2 + (Y[t] - y_i[t])^2} \tag{3.1}$$

$$E_{\text{fly}}[t] = P_f \left((X[t] - X[t-1])^2 + (Y[t] - Y[t-1])^2 \right)^{1/2} V^{-1} \tag{3.2}$$

Uma vez que o UAV receberá tarefas dos usuários dentro do seu raio de cobertura, o custo energético para para a transferência da tarefa e processamento do usuário i para o UAV é dado pela Equação 3.3, onde P_h é a potência de sobrevoo e T_{it}^{trans} é o tempo de envio da tarefa, dado pela Equação 3.4. Esta por sua vez considera o número de bits da tarefa para computação e a taxa de transmissão do usuário i para o UAV, dada pela Equação 3.5. A taxa R_{it} é calculada baseada na Lei de Shannon, onde β é da largura de banda do sistema, ρ denota a potência de transmissão do usuário, σ_2 representa a potência do ruido gaussiano branco aditivo e h refere-se ao ganho de potencia do canal considerando a visada direta entre o UAV e o usuário i no instante t. Este ganho é dado pela Equação 3.6.

$$E_{\text{hov}}[t] = P_h T_{\text{i}}^{\text{trans}}[t] \tag{3.3}$$

$$T_i^{trans} = \frac{D_m[t]}{R_i[t]} \tag{3.4}$$

$$R_i[t] = \beta log_2(1 + \frac{\rho h}{\sigma_2}) \tag{3.5}$$

$$h_i[t] = \beta_0 (d_i[t])^{-2} \tag{3.6}$$

Sendo $K_p[t]$ a quantidade de tarefas sendo processados pelo UAV no slot de tempo t, N_a a quantidade de usuários que estão transferindo uma tarefa para ser processada pelo UAV, a energia total consumida ao longo da execução pelo UAV é dada pela Equação 3.7.

$$E_{total} = \sum_{t}(E_{fly}[t] + E_{hov}[t]K_p[t] + \sum_{n=1}^{N_a} E_{hov}[t]) \tag{3.7}$$

Considerando que cada tarefa do usuário i tem uma restrição de latencia de retorno da resposta l_i a ser atendida, que representa o tempo máximo que UAV dispõe para devolver o resultado do processamento ao usuário, dado em slots de tempo k, a taxa de de tarefas não devolvidas L ao longo da operação do MEC-UAV é dada pelo complemento da razão entre o número de tarefas retornadas com sucesso $\tau_{task,s}$ e o número de tarefas admitidas $\tau_{task,a}$ pelo MEC-UAV para processamento, como ilustra a Equação 3.8. Já a taxa de usuários com requisicões aceitas pelo UAV para processamento A é dada pela Equação 3.9, que considera o número de requisiçoes de tarefas aceitas $\tau_{req,a}$ e o número total de requisições $\tau_{req,\mathrm{t}}$. Dado que o MEC-UAV se movimenta na área alvo para atender aos usuários, o tempo que um usuário consegue enviar a sua requisicão para processamento pelo MEC-UAV pode variar ao longo da operação devido a fatores como cobertura, por exemplo, o tempo médio para receber as requisicões é denotado por W, que é a média dos tempos de aguardo de todas as tarefas aceitas. A Tabela 1 sumariza os símbolos adotados na descrição do ambiente MEC-UAV.

$$\mathsf{L} = 1 - \frac{\tau_{task,s}}{\tau_{task,a}} \tag{3.8}$$

$$\mathsf{A} = \frac{\tau_{req,a}}{\tau_{req,t}} \tag{3.9}$$

Dado as descrições anteriores, o seguinte problema de otimização de trajetória do MEC-UAV para atender demandas de processamento de usuários móveis pode ser formulado. Dado um conjunto de usuários com tarefas a serem processadas remotamente no nó MEC-UAV, definir a sequencia de posicoes do MEC-UAV $\{X[t], Y[t]\}$ que minimiza o consumo total de energia gasto na operação, a taxa de requisições atendidas, a taxa de requisições respondidas e o tempo de espera do usuário até o atendimento da requisição. Sendo λ_1, λ_2, λ_3 e λ_4 os pesos de cada parcela que são atribuídos. O problema está definido na Equação 3.10.

$$\mathsf{P}: \min_{\{X[t],Y[t]\}} \lambda_1 E_{total} + \lambda_2 W + \lambda_3(1 - A) + \lambda_4 L \tag{3.10}$$

Tabela 1 – Lista de Símbolos

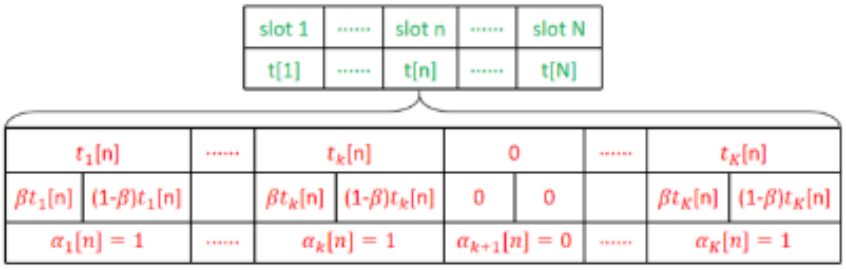
Símbolo	**Significado**
P_f	Potencia de voo (em Watts)
P_h	Potencia de sobrevoo
D_m	Numero de bits para computação da tarefa do usuário
C_m	Número de ciclos de CPU, para computação da tarefa de 1-bit
F_m	Frequência estimada do ciclo de CPU do UAV
β	Largura de Banda do sistema
ρ	Potencia de transmissão do usuário
σ_2	potência do Ruido aditivo Gaussiano branco
β_0	ganho de potência em referencia a distância de 1 metro
V	Velocidade do UAV
H	Altura do UAV
$\tau_{req,a}$	Requisições atendidas
$\tau_{req,t}$	Requisições solicitadas
A	Taxa de requisições aceitas
$\tau_{task,s}$	Tarefas retornadas com sucesso
$\tau_{task,a}$	Tarefas admitidas
L	Taxa de requisições não respondidas
K_p	Quantidade de tarefas processadas no tempo
N_a	Quantidade de usuários transferindo uma tarefa no tempo

3.3 A SOLUÇÃO BASEADA EM APRENDIZADO POR REFORÇO PROPOSTA

A solução proposta para o problema 3.10 é baseada em aprendizagem por reforço e envolve o treinamento e a avaliação da movimentação de um nó MEC-UAV em um cenário com usuários móveis que tomam decisões arbitrárias sobre a realização do offloading de tarefas. Nesse contexto, o MEC-UAV atua como um nó de computação de borda, processando dados e executando tarefas que seriam custosas em termos de consumo de bateria e/ou capacidade computacional para os dispositivos móveis.

A Figura 3 exemplifica a passagem do tempo, dividido em intervalos discretos (time slots), onde em cada time slot, um conjunto máximo de tarefas dos usuários pode ser executada pelo UAV. Cada timeslot é dividido em subslots, que representam o processamento de mais de uma tarefa ao longo do tempo no mesmo timeslot. Considera-se que as tarefas podem ocupar mais do que um sublot, necessitando que o UAV processe a requisição utilizando múltiplos slots.

Figura 3 – Divisão do tempo em time slots.

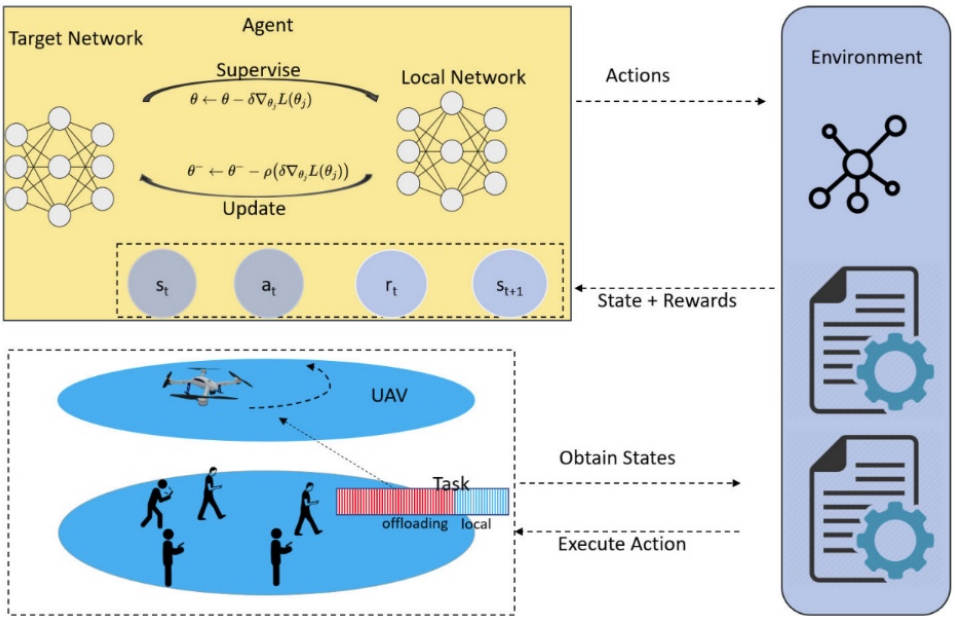


Fonte: (LIN et al., 2023)

Utilizando aprendizagem por reforço, a solução visa definir a melhor rota para o UAV a cada intervalo de tempo, considerando as variáveis do ambiente, como a localização dos usuários, a carga de trabalho e o custo de energia. O solucão baseada em aprendizagem por reforço adota redes neurais profundas. A Figura 4 ilustra o dinâmica das etapas que compõem a solução. Após após uma ação o estado dos usuários e do UAV são alterados e fornecidos ao modelo para tomar a próxima decisão de movimento, a próxima ação e obter as recompensas e o próximo estado.

As abordagens de aprendizado por reforço profundo (Deep Reinforcement Learning - DRL) permitem que os UAVs façam decisões baseadas na observação das mudanças na rede, otimizando tanto o consumo de energia quanto o tempo de execução das tarefas (ZHANG et al., 2021). O modelo de rede neural utiliza redes neurais profundas a partir do modelo DQN, permitindo a extração de características importantes do ambiente para a definição mais refinada da melhor rota.

Figura 4 – Processo de Aprendizado por Reforço.



Fonte: (ZHANG et al., 2021)

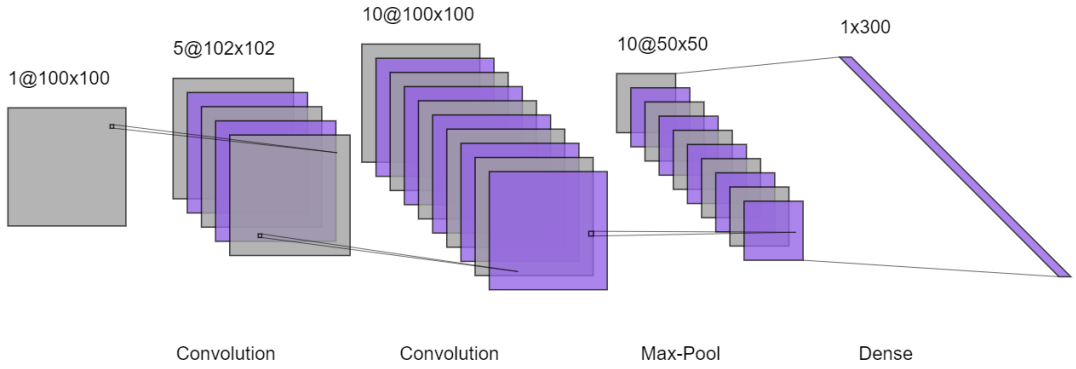
Considera-se que o UAV possui limitação de bateria e de processamento. Devido à carga útil e energia limitadas, o UAV tem capacidade de computação e tempo de voo limitados (ZHANG et al., 2021). Simulando o comportamento de um UAV que está tendo que administrar seus próprios recursos, o UAV toma decisões e se movimenta considerando o consumo de sua bateria e os usuários que estão naquela região de cobertura em contraste com seu atual estado de processamento e fila de tarefas.

O Algoritmo 1 para a rede Q profunda (Deep Q-Network - DQN) treina o modelo de redes neurais, reagindo as mudanças no ambiente e obtendo a atualização dos pesos baseados na função de recompensa, que premia e pune as ações do UAV. Ele armazena as experiências passadas em uma pilha (replay buffer). As transições entre os estados e uma amostra aleatória dessas esperiências são usadas a cada passo para atualizar a rede neural.

O modelo de rede neural profunda utilizado combina uma rede neural convolucional (CNN) com uma rede neural recorrente com memória de curto prazo (LSTM). Na entrada da CNN é fornecida a informação do mapa do ambiente em grid. Uma matriz no tamanho do espaço de movimentação 100x100, como uma imagem, onde os usuários estão posicionados e o seu estado e sua posição são reppresentados como valores nessa matriz. . Já na entrada da LSTM, são inseridas informações contextuais de cada usuário, como posição e distância até o uav. Esse modelo foi formulado para processar dados de entrada que incluem tanto informações espaciais, quanto sequenciais. Inicialmente, uma parte da entrada é recebida em formato de

grid representando uma matriz bidimensional e é processada por camadas convolucionais, que aplicam filtros para extrair características espaciais. Essas camadas possuem filtros de tamanho 3x3, com passo (stride) de 1 e preenchimento (padding) de 2, seguidas por uma camada de agrupamento (pooling), que reduz a dimensionalidade de saída através do agrupamento pelo valor máximo (Max Pooling), conforme a Figura 5.

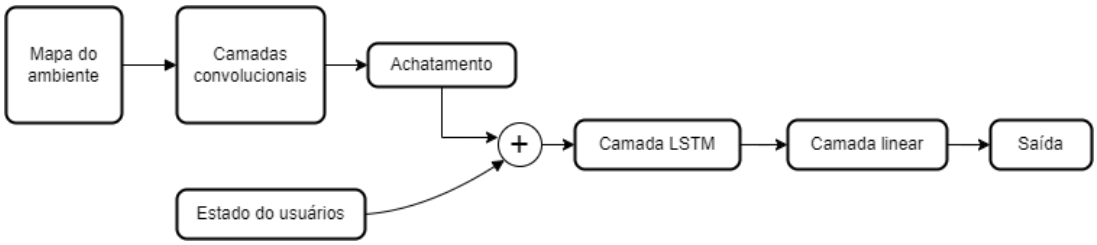
Figura 5 – Extração de características espaciais com CNN



Fonte: Próprio autor

Um achatamento (Flatten) é feito com a saída da camada convolucional, convertendo o mapa de características em um vetor unidimensional. Esta saída achatada é passada então por uma camada totalmente conectada para redução e refinamento das características extraídas. Após essa transformação, o vetor resultante é combinado com os dados do estado do sistema e fornecidos na entrada da LSTM para que a LSTM possa processar esses dados e obter informações contextuais ao longo do tempo. A LSTM possui ainda uma camada intermediária linear e no fim uma camada de saída, que possui 6 opções de movimento. A Figura 6 ilutra a arquitetura da rede.

Figura 6 – Arquitetura da rede neural



Fonte: Próprio autor

Sendo $o[n]$, a saída da rede, tem-se que a direção é dada pela Equação 3.11, a velocidade pela Equação 3.12 e a próxima posição pela Equação 3.13.

$$c[n+1] = \begin{cases} c[n], & \text{se } o[n] \in 0,1,2,3,(velocidade) \\ (c[n] - \frac{\pi}{4}) \mod 2\pi, & \text{se } o[n] = 4,(esquerda) \\ (c[n] + \frac{\pi}{4}) \mod 2\pi, & \text{se } o[n] = 5,(direita) \end{cases} \tag{3.11}$$

$$v[n+1] = \begin{cases} 0, & \text{se } o[n] = 0, \\ v[n] - 1, & \text{se } o[n] = 1, \\ v[n], & \text{se } o[n] = 2, \\ max(1 + v[n], 5), & \text{se } o[n] = 3, \\ v[n], & \text{se } o[n] \in 4,5, \end{cases} \tag{3.12}$$

$$\begin{aligned} x[n+1] &= x[n] + v[n] \cdot \cos(d[n]) \\ y[n+1] &= y[n] + v[n] \cdot \sin(d[n]) \end{aligned} \tag{3.13}$$

Através dessa abordagem, a solução busca otimizar a movimentação do UAV, em relação a quantidade de usuários atendidos, tempo de resposta e consumo de energia ao transitar pelo ambiente e fazer o processamento das tarefas dos usuários dentro da área de cobertura.

Algoritmo 1: Algoritmo da Deep Q-Network (DQN) para treinamento do modelo

Initialize replay memory D to capacity N
Initialize action-value model with random weights θ
Set learning rate α and discount factor γ
Set exploration rate ϵ (decay factor ϵ_{decay})
for each episode **do**
  Initialize state s_1 with environment setup
  for each step of episode **do**
    Get current system state s_t from environment
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
    Execute action a_t, receive reward r_t, and observe new state s_{t+1}
    Store transition $(s_t, a_t, r_t, s_{t+1}, \text{done})$ in D
    Sample random mini-batch from D
    for each transition in mini-batch **do**
      Compute target y_j
      Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} Q(s_{j+1}, a'; \theta) & \text{otherwise} \end{cases}$
      Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ to update θ
    end for
    Clip gradients to avoid exploding gradients
    Update exploration rate $\epsilon \leftarrow \epsilon \cdot \epsilon_{\text{decay}}$
  end for
end for

3.4 MÓDULO PARA A SIMULAÇÃO

Foram implementados alguns métodos em um módulo Python para a simulação do ambiente de interação do UAV. Na classe **EnvironmentRL** o método de inicilização que define os parâmetros do ambiente, onde o espaço de ação é de 6 opções de movimentação e o estado do sistema é divido em um mapa de duas dimensões do grid do ambiente ($clients\_map\_grid$) com as posições sendo os estados dos usuários móveis em cada segmento e um vetor de infor-

mações dos usuários junto ao status do UAV ($state\_system\_array$). Também os parâmetros do início da simulação estão no Código Fonte 1:

Código Fonte 1 – Método de Inicialização do ambiente

```
def __init__(self, initial_clients = 25, grid_size = GRID_SIZE, coverage_radius
    =15):
    super(EnvironmentRL, self).__init__()
    self.coverage_radius = coverage_radius
    self.new_user_prob = 0.04
    self.user_disappearance_prob = 0.004
    self.initial_clients = initial_clients
    self.grid_size = grid_size
    self.total_users_request=0
    self.max_battery_inital = 100
    self.battery = self.max_battery_initial
    self.num_max_clientes = 100
    self.num_status = 8
    self.max_weight_task = 25
    self.uav_status = 5
    self.state_system_array = np.zeros(self.uav_status + self.num_max_clientes *
        self.num_status)
    self.clients_map_grid = np.zeros((self.grid_size, self.grid_size), dtype=int)
    self.action_space = gym.spaces.Discrete(5)
    self.reset()
```

Fonte: Elaborado pelo autor (2024)

A cada passo são calculados os valores da recompensa e o novo estado do sistema após uma ação do UAV. Nesse passo também são feitos alguns movimentos probabilísticos que modificam o estado dos usuários e tornam o ambiente mais dinâmico. São obtidas as informações do cenário e retornadas para posterior avaliação. Onde também é calculado os custos de energia de movimentação, transferência e processamento. Este passo está apresentado no Código Fonte 2:

Código Fonte 2 – Passo de movimentação do UAV em interação com o ambiente

```
def step(self, action):
    self.undeterministic_random_movement()
    acao, penalty, pos_penalty = self.take_action(action)
    reward, status = self.get_reward(penalty, pos_penalty)
    self.observation, info = self.get_observation(reward, status)
    done = self.is_done(reward)
    self.state = self._get_state()
    return self.state, reward, done, {}, info
```

Fonte: Elaborado pelo autor (2024)

A mudança do cenário que pode ocorrer a cada passo da simulação envolve o surgimento de um usuário novo, a saída de um usuário que está em estado parado e a mudança de estado de um usuário que pode solicitar uma tarefa para ser processada com uma probabilidade P_i. Os estados dos usuários estão definidos como: 1 - Usuário parado, 2 - usuário solicitando offloading, 3 - usuário aguardando resposta. Essa dinâmica do ambiente está implementada no Código Fonte 3 a seguir:

Código Fonte 3 – Mudança probabilística do ambiente

```
def undeterministic_random_movement(self):
    #Geracao de um novo usuario
    if np.random.rand() < self.new_user_prob:
        while True:
            pos = np.random.randint(0, self.grid_size, size=2)
            if self.clients_map_grid[pos[0]][pos[1]] == 0:
                break
        new_user_position = pos.tolist()
        self.users_positions.append(new_user_position)
        self.users_time.append(0)
        self.response_time.append(0)
        self.user_states = np.append(self.user_states, 1)
        self.process_tasks = np.append(self.process_tasks, 1)
        self.clients_map_grid[new_user_position[0]][new_user_position[1]] = 1

    index = 0
    # Probalidade de um usuario sair do ambiente
    while index < self.user_states.shape[0]:
        if self.user_states[index] == 1:
            if np.random.rand() < self.user_disappearance_prob:
                user_index = index
                self.clients_map_grid[self.users_positions[user_index][0]][self.
                    users_positions[user_index][1]] = 0
                self.users_positions.pop(user_index)
                self.users_time.pop(user_index)
                self.response_time.pop(user_index)
                self.user_states = np.delete(self.user_states,user_index, 0)
                self.process_tasks = np.delete(self.process_tasks, user_index, 0)
                self.fila.ajustar_ids_pacotes(user_index)
        index += 1


    #Probabilidade de um usuario solicitar offloading
    for index, user in enumerate(self.user_states):
        if user == 1:
            if np.random.rand() < 0.015:
                self.clients_map_grid[self.users_positions[index][0]][self.
```

```
                    users_positions[index][1]] = 2
                self.user_states[index] = 2
                self.users_time[index] = 1
                self.total_users_request += 1
                taks_weight = np.random.randint(1, self.max_weight_task)
                self.process_tasks[index] = taks_weight
```

Fonte: Elaborado pelo autor (2024)

A cada passo (time slot), o UAV pode processar até 5 subslots de tarefas simultâneamente e a política de processamento é sempre priorizar os pacotes mais antigos. O tamanho das tarefas do usuário varia de 1 subslot até 25 subslots. Quando um usuário que está solicitando offloading está dentro da área de cobertura, o UAV recebe a requisição, o estado do usuário muda e a sua tarefa entra para a fila de processamento. Após a execução da tarefa pelo UAV e o retorno dos dados ou a perda da tarefa, o usuário que solicitou a requisição irá mudar de estado para estado parado.

Esse ambiente gerado interage com o modelo de aprendizado por reforço a cada passo, retornando o estado do ambiente e obtendo a ação indicada pelo modelo para a próxima movimentação do UAV. As recompensas são calculadas e usadas para ajustar o modelo durante o momento de treinamento. No momento da simulação, as métricas e valores dos estados de cada componente do sistema, como o estado dos usuários, o tempo de processamento de cada tarefa e o custo de energia por timeslot são calculadas, retornadas como informações do sistema e armazenadas para posterior avaliação.

4 ANÁLISE DE RESULTADOS

Este capítulo apresenta os cenários de simulação e os resultados de avaliação da solucão proposta em comparação com outras abordagens da literatura. As abordagens são avaliadas em termos de metricas como números de usuários atendidos, consumo de energia e tempo para processamento. São apresentados os resultados obtidos em 10 instâncias de simulação, com cada uma possuindo 1000 passos.

4.1 CENÁRIO DE SIMULAÇÃO

A simulação foi realizada em um ambiente de movimentação limitada a uma área em forma de grid quadrado de 100mx100m, com cada segmento tendo o tamanho de 1 metro e os usuários poderiam surgir em qualquer um desses segmentos. O tempo de duração do timeslot é de 1s e o MEC-UAV é colocado em um desses segmentos e se movimenta a uma velocidade máxima de 5 segmentos por time slot (5m/s) e ele pode atender os usuários que estão dentro do raio de cobertura. A Tabela 2 sumariza os parâmetros adotados na simulação e treinamentoda rede neural.

Tabela 2 – Parâmetros de simulação e treinamento

Parâmetro	**Valor**
Número de episódios	1500
Número de time slots por episódio	1000
Número inicial de usuários	25
Probabilidade de surgimento de um usuário	0.04
Probabilidade de saída dos usuários	0.004
Probabilidade de solicitar uma tarefa (P_i)	0.015
Duração do timeslot	1s
Tamanho do grid	[100m, 100m]
Tamanho do segmento	1m
Raio de cobertura (r_c)	15m
Taxa de aprendizado lr	0.002
Taxa de desconto γ	0.90
Memória máxima de replay	100.000
Tamanho do Batch	4000
ϵ inicial	0.9
ϵ_{decay}	0.9995
Fator de subtração f do ϵ por episódio	0.005

O treinamento da solucão foi feito utilizando 1500 episódios, com 1000 time slots por episódio, a uma taxa de aprendizado de 0,004 e memória para replay da experiência de treinamento de até 100.000, onde as experiências passadas armazenadas também serão usadas para treinar o modelo. O tamanho do batch de treinamento, onde uma amostra dessa memória é aleatóriamente escolhida, é de 4000, e estas amostras são usados a cada time slot para melhorar o desempenho do treinamento. Durante o treinamento, a cada time slot, a taxa de exploração de movimentos aleatórios ϵ sofre um decaimento a partir de ϵ_{decay} e a cada episódio é reduzido o valor de ϵ por um fator de subtração f durante os primeiros 160 episódios.

A cada time slot, os usuários que estão dentro do raio de cobertura e possuem tarefas a serem enviadas para processamento, irão efetuar o offloading e aguardar a resposta do processamento. Os usuários possuem um tempo máximo de espera para retorno do resultado (l_i), conforme a Tabela 3. Caso a quantidade de tarefas sendo executadas seja um impedimento para que o UAV execute a nova tarefa no tempo devido, a tarefa será perdida. Cada tarefa pode possuir diferentes tempos de execução, dados em termos de time slots. A diferenciação das tarefas está em quantos subslots de um timeslot serão necessários para processar completamente aquela tarefa. Consideramos o tamanho máximo de subslots que uma tarefa pode

demandar é 25 e o mínimo 1 e o tempo máximo que o usuário pode aguardar a resposta da tarefa é até 6 time slots. Além disso, considera-se que o UAV pode processar até 5 subslots por time slot.

Tabela 3 – Custo de processamento de uma tarefa do usuário

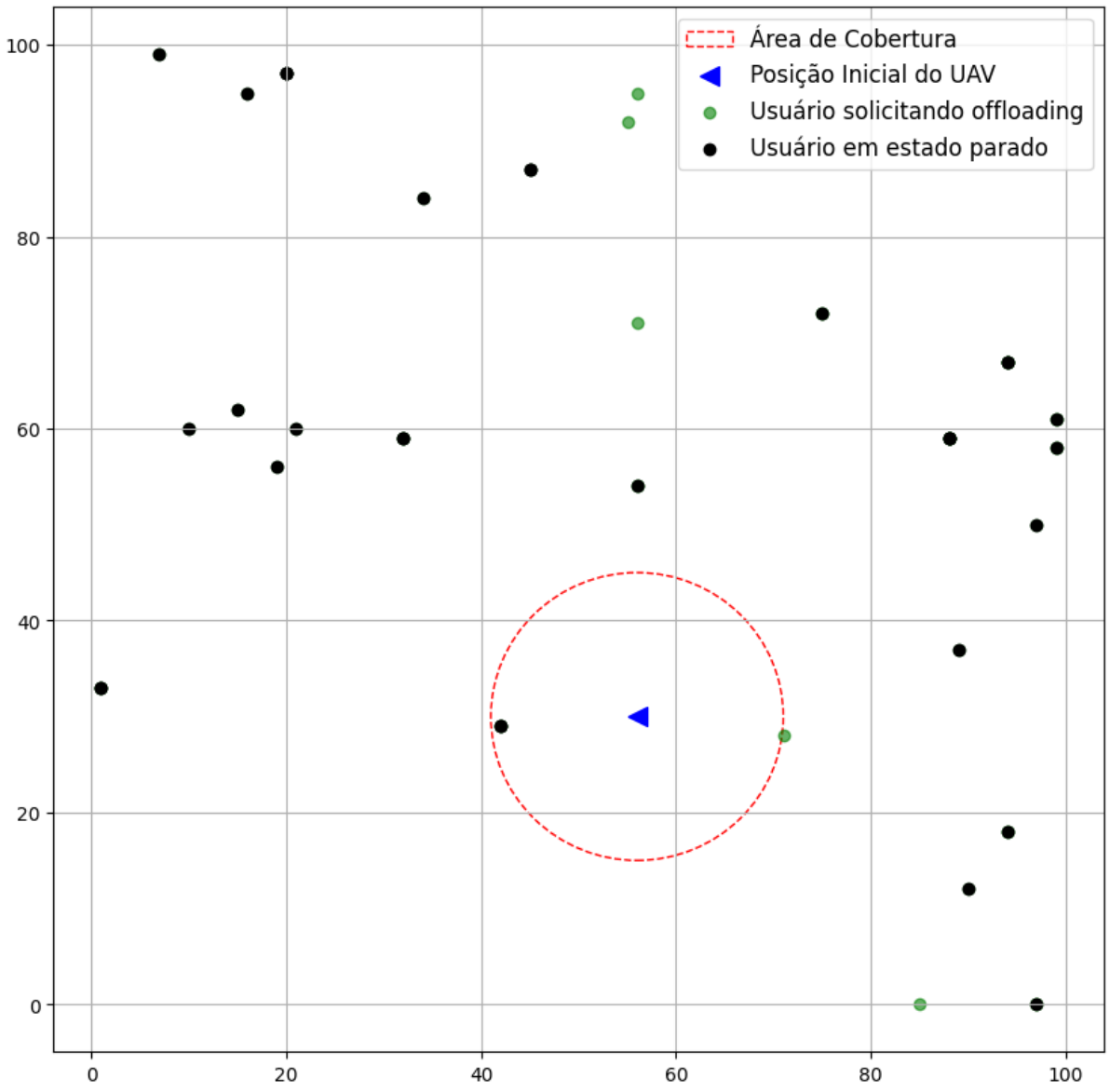
Parâmetro	**Valor**
Custo k de processamento de uma tarefa em subslots	$1 \leq k \leq 25$
Tempo máximo do usuário aguardando processamento l_i	6 time slots
Máximo processamento paralelo de subslots do UAV K_p	5

Tabela 4 – Parâmetros para custo de energia

Parâmetro	**Valor**
P_h	0.08
P_f	0.11
C_m	1000
D_m	10,000,000
F_m	2,000,000,000
Vf_{max}	5m/s
β	20Mhz
β_0	1
ρ	0.1
σ_2	1×10^{-9}

A Tabela 4 apresenta os valores dos parâmetros usados nos cálculo de custo de energia para movimentação e processamento. A cada time slot, o movimento e a quantidade de usuários irá influenciar no gasto de bateria do UAV. A Figura 7 ilustra um exemplo de cenário de movimentação do UAV no grid, a posição dos usários e a área de cobertura do UAV.

Figura 7 – Exemplo do cenário de simulação



4.2 MÉTRICAS E SOLUÇÕES PARA COMPARAÇÃO

Para avaliar a desempenho da solução proposta foram usadas as métricas de quantidade de usuários atendidos, média de tempo de atraso para atendimento, quantidade de tarefas completadas, tempo de processamento, quantidade de usuários com tarefas processados por time slot, porcentagem de usários respondidos, custo de energia de movimentação, custo de energia de processamento e consumo de bateria ao longo do tempo. Além disso, a comparação com outras abordagens da literatura foi conduzida, considerando as seguintes:

- **MLP:** A primeira solução a ser comparada é uma rede neural perceptron multi camadas

(MLP) simples, com 3 camadas escondidas, que obtém o estado do ambiente incluindo as posições do grid e transforma em um vetor unidimensional de características.

- **K-Means:** A segunda solução para comparação é baseado na movimentação do UAV em direção ao usuário ou grupo de usuários mais próximo, através do método de agrupamento K-Means. Esta solução agrupa os usuários que estão solicitando o offloading, baseado nas posições deles.

- **Random:** O terceiro modelo é uma política de movimentação aleatória, onde uma das 6 possibilidades de movimentação é escolhida em cada slot de tempo.

4.3 RESULTADOS

A Tabela 5 apresenta os resultados médios obtidos pela solução proposta na Seção 3.3 em comparação com os outros modelos e em termos das métricas indicados na Seção 4.2. Em termos do tempo médio de atraso para atendimento de usuários, o modelo K-Means apresenta o melhor desempenho, dado o seu movimento sempre em direção aos usuários mais próximos que estão solicitando. Porém, isso também impactou no custo de energia devido a alta carga de processamento e movimento frequente em alta velocidade em direção aos usuários.

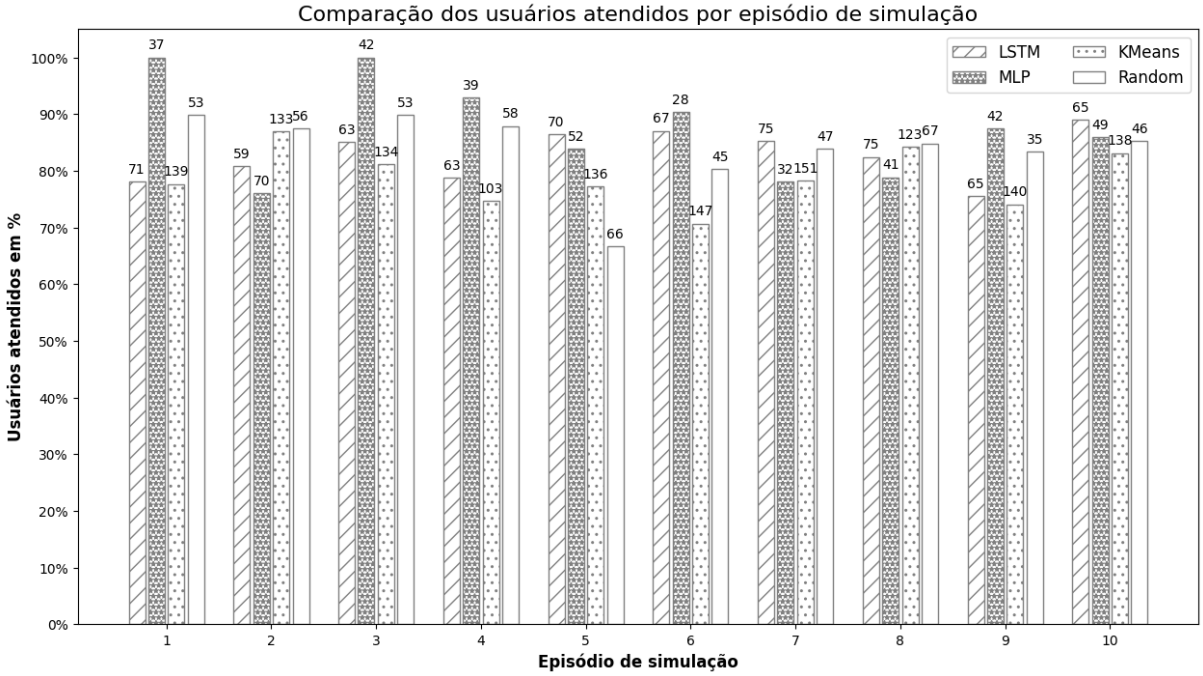
Quanto a proporção de usuários atendidos, tem-se que a LSTM apresentou acum desemepenho 73% enquanto o modelo MLP acabou tendo apenas 58%. Porém, por processar menos pacotes, tanto o MLP, quanto o Random tiveram uma porcentagem maior de tarefas completadas. Isso fica evidente quando se observa a métrica de usuários tendo suas tarefas processados por timeslot. Já em relação ao K-Means, o LSTM teve desempenho superior em termos de taxa de tarefas completadas. Esses resultado se dá devido a capacidade limitada de processamento paralelo do UAV.

A Figura 8 apresenta a quantidade de usuários respondidos em cada instância de simulação. Nota-se que o modelo LSTM apresentou uma média superior ao modelo com K-Means na maioria (7) dos casos. E em alguns casos sendo melhor que o modelo MLP e o Random. Na maioria dos casos, o modelo MLP apresentou o melhor resultado. Isso se explica pela quantidade menor de tarefas sendo processadas pelo modelo MLP. Enquanto o modelo K-Means obteve uma quantidade muito maior de tarefas para processar, que acabou gerando mais perdas de tarefas.

Tabela 5 – Comparação entre os algoritmos

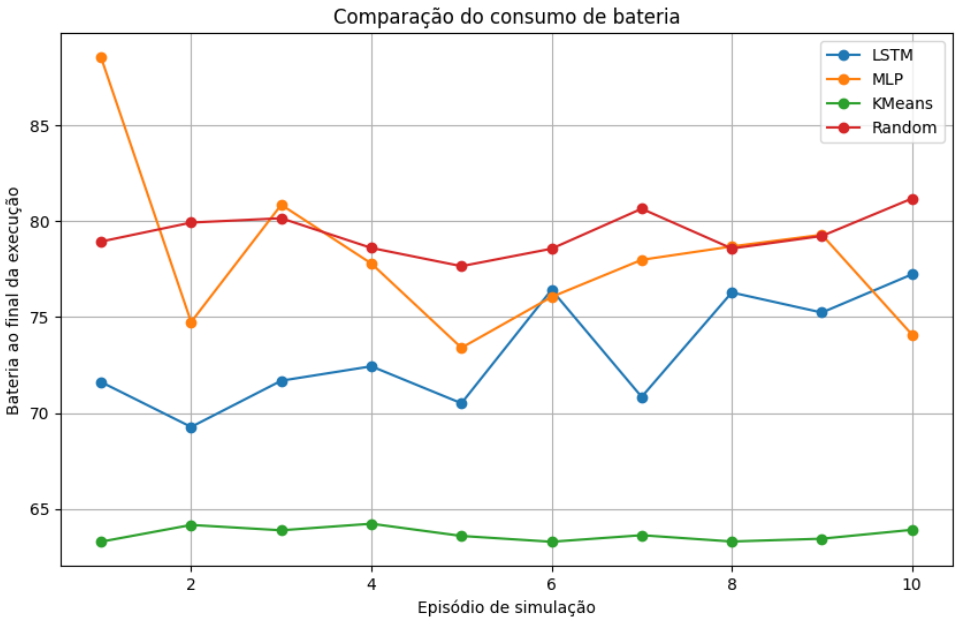
Valores médios	**LSTM**	**MLP**	**K-Means**	**Random**
Tempo para atendimento (s)	164.56	135.45	32.02	125.37
Usuários que solicitaram atendimento	110.10	86.60	175.50	93.10
Usuários atendidos	81.40	50.40	171.30	63.40
Proporção de usuários atendidos	73,93%	58,20%	97,61%	68,10%
Tarefas completadas	82.84%	87.35%	78.81%	83.93%
Tarefas perdidas	17.16%	12.65%	21.19%	16.07%
Tempo de Processamento (s)	3.00	3.00	2.92	3.06
Usuários tendo seus pacotes processados por timeslot	0.2868	0.1717	0.6156	0.2248
Custo de Energia de Voo (J)	28.60	22.94	39.84	22.06
Custo de Energia de Processamento (J)	2.19	1.36	4.40	1.76
Bateria ao Final	73.16%	78.15%	63.68%	79.35%

Figura 8 – Proporção de usuários que tiveram sua resposta, após o offloading



O consumo de bateria total ao final de cada instância de simulação é apresentado na Figura 9. O modelo K-Means foi o que mais consumiu energia em sua movimentação, com uma média de consumo de 39.84 J de custo de movimentação e a porcentagem de bateria do UAV com 63,68% e o LSTM sendo o segundo maior consumidor consumindo em média 28.60 J de movimentação e com a bateria em média com 73,16% ao final da execução. Já o modelo Random foi o que menos consumiu energia.

Figura 9 – Consumo de bateria total do UAV em cada simulação



As Figuras 10, 11, 12 e 13 apresentam a trajetória do UAV realizada em uma instância de simulação, sob a orientação de cada um dos modelos e perspectiva futura. Elas mostram a posição de todos os usuários que surgiram e foram ou não atendidos e a estratégia de movimentação do UAV em cada semento do grid, com a trajetória completa do UAV, desde o ponto inicial, onde o UAV foi colocado, até o ponto final.

O modelo LSTM apresenta um estilo de movimentação mais coerente, que sinaliza um aprendizado mais assertivo em relação ao objetivo. Apesar de não ter tido as melhores métricas em todos os parâmetros, exibe um comportamento de movimentação mais coerente com um tipo de movimentação real em que o UAV busca atender todos os usuários em todas as regiões de forma mais igualitária. Os modelos MLP e random acabaram restringindo sua movimentação em determinados espaços limitados. E o modelo K-Means, apesar de percorrer todo o cenário, acabou consumindo muita energia, praticando sempre movimentos longos e com alta velocidade, o que não é acompanhado pela capacidade de processamento do UAV.

Figura 10 – Trajetória LSTM

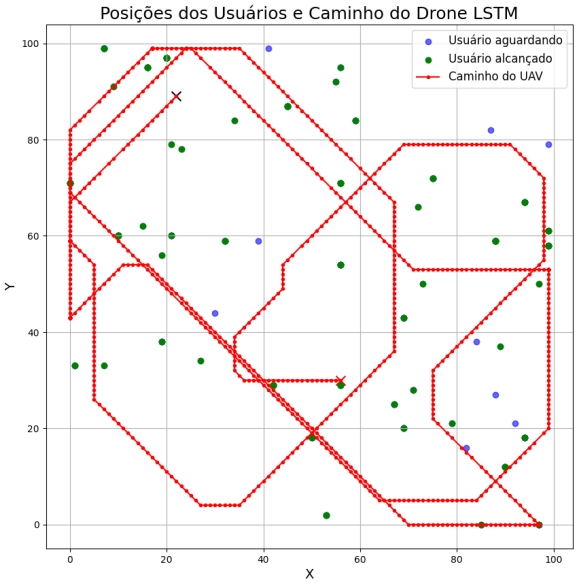


Figura 11 – Trajetória MLP

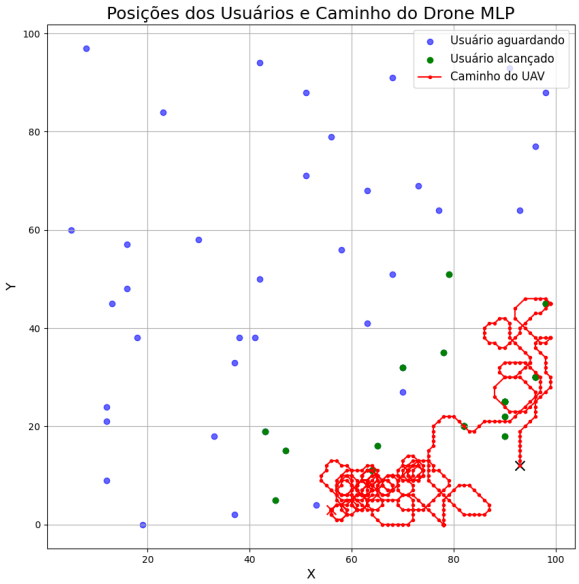


Figura 12 – Trajetória K-Means

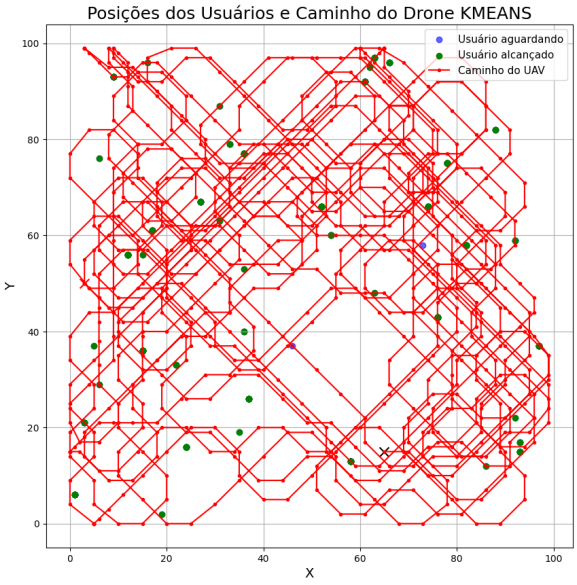
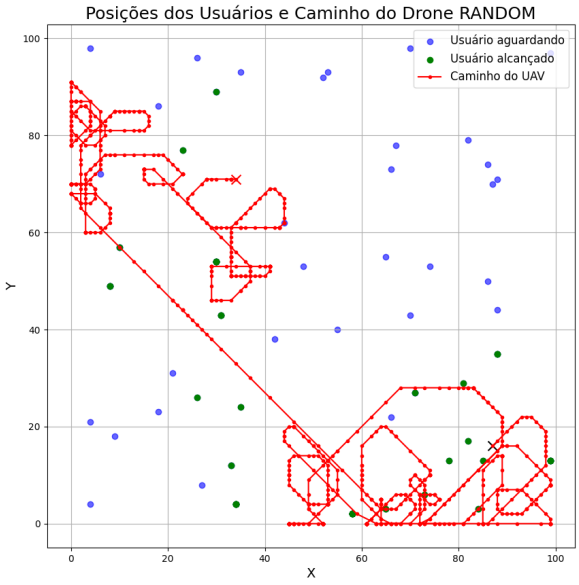


Figura 13 – Trajetória Random



5 ANÁLISE DE RESULTADOS

Este trabalho apresentou uma soluçao baseada em aprendizado por reforço para o problema da definição de trajetória de UAVs voltados ao fornecimento de processamento computacional a usuários de redes móveis 5G. Comparou-se a solução proposta com diferentes soluções baseadas em redes neurais, clusterização e de movimentação aleatória.

Considerando aspectos de limitação de bateria do UAV, limitação de processamento, tempo de processamento, variação na quantidade de usuários, mudança no estado dos usuários, apresentou-se resultados em relação a atraso de atendimento aos usuários, otimização do custo de energia e quantidade de usuários atendidos.

Na comparação, foi possível perceber a diferença entre arquiteturas com diferentes características, onde a arquitetura mais robusta que usa camadas CNN junto a LSTM, conseguiu obter melhores informações contextuais e apresentar um desempenho melhor e mais coerente com o problema de definição de trajetória. Com os modelos que não usam as redes neurais foi possível perceber que algumas políticas de movimentação tradicionais como o movimento até o usuário mais próximo apresentam um desempenho em determinadas características aceitável. O que é um bom indicador de que uma política de movimentação que considera o julgamento de ação, tanto a partir de um modelo profundo, quanto avaliando qual seria a movimentação mais trivial pode ser uma boa estratégia a ser adotada na definição de trajetória.

Como trabalhos futuros pretende-se refinar aspectos personalizados na definição da próxima ação do UAV, considerando a avaliação de movimentos triviais junto ao movimento indicado pela rede neural treinada. Avaliar alguns outros modelos de redes, arquiteturas e hiperparâmetros de treinamento mais sofisticados. Avaliar o cenário de movimentação de múltiplos UAVs para fornecimento de infraestrutura MEC. Também avaliar uma taxa de probabilidade de falha na comunicação e transferência de dados, com diferentes taxas de transferência e tempos de transferência variados.

REFERÊNCIAS

AGYAPONG, P. K.; IWAMURA, M.; STAEHLE, D.; KIESS, W.; BENJEBBOUR, A. Design considerations for a 5g network architecture. *IEEE Communications Magazine*, v. 52, n. 11, p. 65–75, 2014.

AHMED, E.; REHMANI, M. H. Mobile edge computing: Opportunities, solutions, and challenges. *Future Generation Computer Systems*, v. 70, p. 59–63, 2017. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X16303260>.

AL-FALAHY, N.; ALANI, O. Y. Technologies for 5g networks: Challenges and opportunities. *IT Professional*, v. 19, n. 1, p. 12–20, 2017.

ALSENWI, M.; TRAN, N. H.; BENNIS, M.; BAIRAGI, A. K.; HONG, C. S. embb-urllc resource slicing: A risk-sensitive approach. *IEEE Communications Letters*, v. 23, n. 4, p. 740–743, 2019.

CHEN, J.; CAO, X.; YANG, P.; XIAO, M.; REN, S.; ZHAO, Z.; WU, D. O. Deep reinforcement learning based resource allocation in multi-uav-aided mec networks. *IEEE Transactions on Communications*, v. 71, n. 1, p. 296–309, 2023.

CHEN, X.; JIAO, L.; LI, W.; FU, X. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, v. 24, n. 5, p. 2795–2808, 2016.

DINH, T. Q.; LA, Q. D.; QUEK, T. Q. S.; SHIN, H. Learning for computation offloading in mobile edge computing. *IEEE Transactions on Communications*, v. 66, n. 12, p. 6353–6367, 2018.

DOGRA, A.; JHA, R. K.; JAIN, S. A survey on beyond 5g network with the advent of 6g: Architecture and emerging technologies. *IEEE Access*, v. 9, p. 67512–67547, 2021.

DU, X.; LI, X.; ZHAO, N.; WANG, X. A joint trajectory and computation offloading scheme for uav-mec networks via multi-agent deep reinforcement learning. In: *ICC 2023 - IEEE International Conference on Communications*. [S.l.: s.n.], 2023. p. 5438–5443.

FOURATI, H.; MAALOUL, R.; CHAARI, L. A survey of 5g network systems: challenges and machine learning approaches. *International Journal of Machine Learning and Cybernetics*, v. 12, n. 2, p. 385–431, Feb 2021. ISSN 1868-808X. Disponível em: <https://doi.org/10.1007/s13042-020-01178-4>.

HORTELANO, D.; de Miguel, I.; BARROSO, R. J. D.; AGUADO, J. C.; MERAYO, N.; RUIZ, L.; ASENSIO, A.; MASIP-BRUIN, X.; FERNáNDEZ, P.; LORENZO, R. M.; ABRIL, E. J. A comprehensive survey on reinforcement-learning-based computation offloading techniques in edge computing systems. *Journal of Network and Computer Applications*, v. 216, p. 103669, 2023. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804523000887>.

JEONG, S.; SIMEONE, O.; KANG, J. Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning. *IEEE Transactions on Vehicular Technology*, v. 67, n. 3, p. 2049–2063, 2018.

JIANG, W.; HAN, B.; HABIBI, M. A.; SCHOTTEN, H. D. The road towards 6g: A comprehensive survey. *IEEE Open Journal of the Communications Society*, v. 2, p. 334–366, 2021.

JYOTHIRMAI, M.; GOPAL, K.; SAILAJA, M. A review of mobile computation offloading techniques. In: HEMANTH, J.; PELUSI, D.; CHEN, J. I.-Z. (Ed.). *Intelligent Cyber Physical Systems and Internet of Things*. Cham: Springer International Publishing, 2023. p. 519–532. ISBN 978-3-031-18497-0.

KUMAR, K.; LIU, J.; LU, Y.-H.; BHARGAVA, B. A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, v. 18, n. 1, p. 129–140, Feb 2013. ISSN 1572-8153. Disponível em: <https://doi.org/10.1007/s11036-012-0368-0>.

LI, B.; LIU, Y.; TAN, L.; PAN, H.; ZHANG, Y. Digital twin assisted task offloading for aerial edge computing and networks. *IEEE Transactions on Vehicular Technology*, v. 71, n. 10, p. 10863–10877, 2022.

LI, Y.; JIANG, C. Distributed task offloading strategy to low load base stations in mobile edge computing environment. *Computer Communications*, v. 164, p. 240–248, 2020. ISSN 0140-3664. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0140366420319691>.

LIN, L.; LIAO, X.; JIN, H.; LI, P. Computation offloading toward edge computing. *Proceedings of the IEEE*, v. 107, n. 8, p. 1584–1607, 2019.

LIN, N.; TANG, H.; ZHAO, L.; WAN, S.; HAWBANI, A.; GUIZANI, M. A pddqnlp algorithm for energy efficient computation offloading in uav-assisted mec. *IEEE Transactions on Wireless Communications*, v. 22, n. 12, p. 8876–8890, 2023.

LUO, Q.; LUAN, T. H.; SHI, W.; FAN, P. Deep reinforcement learning based computation offloading and trajectory planning for multi-uav cooperative target search. *IEEE Journal on Selected Areas in Communications*, v. 41, n. 2, p. 504–520, 2023.

MAO, Y.; YOU, C.; ZHANG, J.; HUANG, K.; LETAIEF, K. B. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, v. 19, n. 4, p. 2322–2358, 2017.

MAO, Y.; ZHANG, J.; LETAIEF, K. B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications*, v. 34, n. 12, p. 3590–3605, 2016.

PHAM, Q.-V.; FANG, F.; HA, V. N.; PIRAN, M. J.; LE, M.; LE, L. B.; HWANG, W.-J.; DING, Z. A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art. *IEEE Access*, v. 8, p. 116974–117017, 2020.

POPOVSKI, P.; TRILLINGSGAARD, K. F.; SIMEONE, O.; DURISI, G. 5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view. *IEEE Access*, v. 6, p. 55765–55779, 2018.

RANJHA, A.; NABOULSI, D.; EMARY, M. E.; GAGNON, F. Facilitating urllc vis-á-vis uav-enabled relaying for mec systems in 6-g networks. *IEEE Transactions on Reliability*, p. 1–15, 2024.

SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, v. 8, n. 4, p. 14–23, 2009.

SILVA, F. A.; KOSTA, S.; RODRIGUES, M.; OLIVEIRA, D.; MACIEL, T.; MEI, A.; MACIEL, P. Mobile cloud performance evaluation using stochastic models. *IEEE Transactions on Mobile Computing*, v. 17, n. 5, p. 1134–1147, 2018.

WANG, S.; ZHAO, Y.; XU, J.; YUAN, J.; HSU, C.-H. Edge server placement in mobile edge computing. *Journal of Parallel and Distributed Computing*, v. 127, p. 160–168, 2019. ISSN 0743-7315. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0743731518304398>.

WANG, Z.; WEI, T.; SUN, G.; LIU, X.; YU, H.; NIYATO, D. Multi-uav enabled mec networks: Optimizing delay through intelligent 3d trajectory planning and resource allocation. 09 2024.

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

XU, Y.; ZHANG, T.; LIU, Y.; YANG, D.; XIAO, L.; TAO, M. Uav-assisted mec networks with aerial and ground cooperation. *IEEE Transactions on Wireless Communications*, v. 20, n. 12, p. 7712–7727, 2021.

ZHAN, C.; HU, H.; SUI, X.; LIU, Z.; NIYATO, D. Completion time and energy optimization in the uav-enabled mobile-edge computing system. *IEEE Internet of Things Journal*, v. 7, n. 8, p. 7808–7822, 2020.

ZHANG, J.; ZHOU, L.; TANG, Q.; NGAI, E. C.-H.; HU, X.; ZHAO, H.; WEI, J. Stochastic computation offloading and trajectory scheduling for uav-assisted mobile edge computing. *IEEE Internet of Things Journal*, v. 6, n. 2, p. 3688–3699, 2019.

ZHANG, L.; ZHANG, Z.-Y.; MIN, L.; TANG, C.; ZHANG, H.-Y.; WANG, Y.-H.; CAI, P. Task offloading and trajectory control for uav-assisted mobile edge computing using deep reinforcement learning. *IEEE Access*, v. 9, p. 53708–53719, 2021.

ZHOU, F.; HU, R.; LI, Z.; WANG, Y. Mobile edge computing in unmanned aerial vehicle networks. *IEEE Wireless Communications*, PP, p. 1–7, 01 2020.

ZHOU, F.; WU, Y.; SUN, H.; CHU, Z. Uav-enabled mobile edge computing: Offloading optimization and trajectory design. In: *2018 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2018. p. 1–6.