UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de São José do Rio Preto

Douglas Dias Lieira

Mecanismos para Alocação de Recursos IoT em Computação de Borda Utilizando Algoritmos Meta-heurísticos Bioinspirados

São José do Rio Preto

2021

Douglas Dias Lieira

Mecanismos para Alocação de Recursos IoT em Computação de Borda Utilizando Algoritmos Meta-heurísticos Bioinspirados

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Orientador: Prof. Dr. Rodolfo Ipolito Meneguette

São José do Rio Preto

2021

L719m

Lieira, Douglas Dias
Mecanismos para Alocação de Recursos IoT em Computação de Borda Utilizando Algoritmos Meta-heurísticos Bioinspirados / Douglas Dias Lieira. -- São José do Rio Preto, 2021
91 f. : il., tabs., mapas

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Rodolfo Ipolito Meneguette

1. Ciência da computação. 2. Computação de borda. 3. Alocação de recursos. 4. Algoritmos meta-heurísticos bioinspirados. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Douglas Dias Lieira

Mecanismos para Alocação de Recursos IoT em Computação de Borda Utilizando Algoritmos Meta-heurísticos Bioinspirados

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Comissão examinadora

Prof. Dr. Rodolfo Ipolito Meneguette
UNESP - Câmpus de São José do Rio Preto
Orientador

Prof. Dr. Geraldo Pereira Rocha Filho
UnB - Brasília

Prof. Dr. Geraldo Francisco Donegá Zafalon
UNESP - Câmpus São José do Rio Preto

São José do Rio Preto

20 de agosto de 2021

AGRADECIMENTOS

A Deus e a Nossa Senhora das Graças;
Aos meus pais, Osvaldo e Marlene, por toda estrutura familiar para que eu pudesse me dedicar aos estudos;
Aos meus irmãos, à minha família e a todos que me incentivaram durante esta jornada;
Em especial, à minha amada filha Maria Eduarda, minha maior motivação para tudo;
Ao Matheus Quessada, André Cristiani e Joahannes Bruno, pelos auxílios no desenvolvimento dos trabalhos científicos;
Finalmente, ao meu orientador Prof. Dr. Rodolfo Ipolito Meneguette pela sua orientação.

"[...] mire as estrelas e salte o mais alto que der, tome distância e faça o melhor que puder. Só não se permita viver na sombra do talvez, aqui só se vive uma vez! Vença seus medos, você é capaz de voar por cima das vozes que gritam pra você parar. Não há nessa vida algo que não se possa alcançar, você só precisa ir buscar. E encontrar na persistência seu valor. E apesar de seu cansaço, sua dor, nunca se entregar!"

(FAGLIONI, 2020)

RESUMO

A evolução da tecnologia de Quinta Geração e dos dispositivos IoT traz novos desafios aos pesquisadores para aperfeiçoar os processos existentes ou criar novos mecanismos que proporcionem qualidade dos serviços e boa experiência aos usuários. Nesse sentido, uma das dificuldades encontradas é utilizar os serviços disponíveis de modo eficiente. Entre estes serviços, há os serviços de computação de borda que são utilizados para descentralizar os serviços de nuvens para mais próximos dos usuários e, assim, diminuir a latência de comunicação da massiva quantidade de dispositivos IoT. No entanto, esta descentralização ocasiona a limitação de recursos dos servidores de borda. Desse modo, com o intuito de melhorar o gerenciamento de serviços de borda e deixar mais eficiente o processo de tomada de decisão para alocação de recursos computacionais, este trabalho propõe dois mecanismos baseados em métodos meta-heurísticos bioinspirados, o LOBO-IoT e o BALE-IoT. O LOBO-IoT é baseado no algoritmo de otimização do lobo cinzento e o BALE-IoT no algoritmo de otimização da baleia. O LOBO-IoT faz uma analogia onde os dispositivos, ou os serviços de bordas, são considerados os lobos e buscam identificar os três melhores (alpha, beta e delta) para atender as demandas. Enquanto o BALE-IoT faz a analogia de que as baleias são os dispositivos, ou serviços de bordas, que usam os movimentos de caça da baleia jubarte para encontrar a melhor opção. Para simulação, validação e análise da eficiência dos mecanismos considerou-se cenários urbanos com serviços de borda e serviços de borda móvel, que necessitam de um processo de tomada de decisão eficiente para alocar recursos de dispositivos IoT. No processo com serviço de borda foi utilizada a linguagem de programação Python e a distribuição de Pearson III, objetivando a criação de um ambiente urbano com serviços de computação de borda limitados, instalados em RSUs, nos quais os dados da simulação foram gerados sinteticamente. Enquanto no processo para serviços de borda móvel foi considerado um ambiente com serviços de computação de borda móvel que compartilham seus recursos formando um pool de recursos computacionais. Para isso, utilizou-se um simulador de mobilidade urbana SUMO para simulação de um ambiente real e a linguagem de programação Python para implantar os mecanismos e simular as tomadas de decisões para alocação dos dispositivos. Os mecanismos foram comparados com técnicas tradicionais encontradas na literatura e ambos demonstraram ser mais eficientes e facilmente customizados para os dois processos. Com isso, o LOBO-IoT e BALE-IoT conseguiram maximizar o atendimento dos dispositivos e minimizar a utilização dos recursos disponibilizados.

Palavras-chave: Alocação de recursos. Computação de Borda. Dispositivos IoT. Meta-heurísticos Bioinspirados.

ABSTRACT

The evolution of Fifth Generation technology and IoT devices brings new challenges for researchers to improve existing processes or create new mechanisms that provide quality of services and a good user experience. In this sense, one of the difficulties found is to use the available services efficiently. Among these services, there are the edge computing services that are used to decentralize the cloud services closer to the users and, thus, decrease the communication latency of the massive amount of IoT devices. However, this decentralization limits the resources of the edge servers. Thus, to improve the management of edge services and make the decision-making process for the allocation of computational resources more efficient, this work proposes two mechanisms based on bioinspired meta-heuristic methods, the LOBO-IoT and the BALE-IoT. LOBO-IoT is based on the grey wolf optimization algorithm and BALE-IoT on the whale optimization algorithm. The LOBO-IoT makes an analogy where devices, or edge services, are considered the wolfs and seek to identify the best three (alpha, beta and delta) to meet the demands. While BALE-IoT makes the analogy that whales are the devices, or edge services, that use the humpback whale hunting movements to find the best option. For simulation, validation, and analysis of the efficiency of the mechanisms, urban scenarios with edge services and mobile edge services were considered, which require an efficient decision-making process to allocate resources from IoT devices. In the process with edge service, Python programming language and Pearson III distribution were used, aiming to create an urban environment with limited edge computing services, installed in RSUs, in which simulation data were generated synthetically. While in the process for mobile edge services an environment with mobile edge computing services that share their resources forming a pool of computing resources was considered. For this, a SUMO urban mobility simulator was used to simulate a real environment and the Python programming language to implement the mechanisms and simulate decision-making for device allocation. The mechanisms were compared with traditional techniques found in the literature and both proved to be more efficient and easily customized for both processes. With this, LOBO-IoT and BALE-IoT were able to maximize the service provided by the devices and minimize the use of available resources.

Keywords: Resource allocation. Edge Computing. IoT devices. Bioinspired Meta-heuristics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

4G	*Fourth Generation*
5G	*Fifth Generation*
CC	*Cloud Computing*
DBSCAN	*Density-Based Spatial Clustering of Application with Noise*
DP	*Dynamic Programming*
EC	*Edge Computing*
GWO	*Grey Wolf Optimizer*
IIoT	*Industrial Internet of Things*
ILP	*Integer Linear Program*
IoT	*Internet of Things*
LuST	*Luxembourg SUMO Traffic*
MEC	*Mobile Edge Computing*
MV	Máquina virtual
NFV	*Network Functions Virtualization*
NOMA	*Non-Orthogonal Multiple Access*
NPS	*Net Promote Score*
PSO	*Particle Swarm Optimization*
RAN	*Radio Access Network*
RSU	*Roadside Unit*
SMDP	*Semi-Markov Decision Process*
SUMO	*Simulation of Urban Mobility*
TraCI	*Traffic Control Interface*
VCC	*Vehicular Cloud Computing*
VEC	*Vehicular Edge Computing*

VNF *Virtual Network Function*

WOA *Whale Optimization Algorithm*

SUMÁRIO

1 INTRODUÇÃO

Há um aumento exponencial de dispositivos móveis no mundo (ZHAO et al., 2019). Tais dispositivos fazem cada vez mais parte do dia a dia das pessoas. Estima-se que até 2025, 75% da população mundial acessem os conteúdos da web apenas por dispositivos móveis (GSMA, 2020). Acompanhada da evolução dos dispositivos móveis, também evolui a tecnologia dos serviços destes dispositivos, em que se busca proporcionar maior qualidade aos usuários finais. A tecnologia de Quinta Geração (*Fifth Generation* - 5G) é um exemplo. Ela surgiu no mercado prometendo velocidade até 10 vezes maior que as gerações anteriores, aumentando além da velocidade, a cobertura de uma área e a capacidade de respostas das redes sem fio (CHENG, 2020).

A internet para dispositivos móveis evolui constantemente (ALRIKABI et al., 2019). O conceito de 5G junto com a tecnologia de Internet das Coisas (*Internet of Things* - IoT) aparecem como a ferramenta que possibilita a transmissão e comunicação do grande fluxo de dados e sinais desses dispositivos, dos quais alcançam desde locais próximos aos mais distantes (ALRIKABI et al., 2019). A tecnologia 5G-IoT promete aprimorar os serviços de disseminação de dados utilizando sensores que geram dados intermitentemente (MENG et al., 2020). Portanto, um desafio desta evolução de tecnologia é encontrar arquiteturas de rede que sejam capazes de acompanhar a modernização que a tecnologia 5G-IoT traz, para alcançar eficiência na troca de informações pela rede (MENG et al., 2020).

Com o avanço destas tecnologias, é necessário que as estruturas dos serviços sejam adequadas para fornecer o serviço com qualidade aos usuários (ZHAO et al., 2019). Assim, passou-se a utilizar o conceito de Computação de Borda (*Edge Computing* - EC), que descentraliza os serviços de uma computação em nuvem (*Cloud Computing* - CC) para mais próximo dos usuários, com o intuito de minimizar o atraso durante a troca de dados. A EC trabalha com servidores de borda em mini nuvens para estender os recursos de uma nuvem para a borda e deixar próximo do dispositivo do usuário (HASSAN; YAU; WU, 2019). No entanto, a descentralização destes serviços, torna mais complexo o desafio de alocação dos recursos computacionais e comunicação dos dispositivos móveis nas bordas, uma vez que a mini nuvem possui menor poder de recursos computacionais (LI et al., 2019).

Estendendo os serviços de computação móvel à computação de borda, surgiu a tecnologia de computação de borda móvel (*Mobile Edge Computing* - MEC). A MEC é uma plataforma que fornece serviços de computação em nuvem e tecnologia da informação em redes 5G, mais próximos dos usuários de dispositivos móveis (YOUSEFPOUR et al.,

2019). MEC fornece, através da rede de acesso de rádio (*Radio Access Netowork* - RAN), armazenamento e recursos computacionais na borda da rede, diminuindo a latência para uma ampla quantidade de usuários finais de dispositivos móveis (TALEB et al., 2017).

Algoritmos são usados para encontrar melhores soluções para o desafio de alocação de recursos em serviços de borda e serviços de borda móvel. Alguns desses algoritmos têm o objetivo de otimizar e maximizar a utilização dos recursos disponibilizados pelos serviços (LI et al., 2018b; COSTA et al., 2020; PEREIRA et al., 2020; QUESSADA et al., 2021). Entre os vários métodos de alocação encontrados na literatura, sendo eles (ZHENG et al., 2015; KETYKó et al., 2016; ZHANG et al., 2016; KUMAR; ZEADALLY; RODRIGUES, 2016; LI et al., 2019; PEREIRA et al., 2019; AAZAM; HARRAS; ZEADALLY, 2019; GOUDOS et al., 2019; JIANG et al., 2019; MOSES; KAARTHICK, 2019; MENEGUETTE; BOUKERCHE; PIMENTA, 2019; SHAO et al., 2019; HOSSEINIOUN et al., 2020; COSTA et al., 2020; WEERASINGHE et al., 2020; SCHNEIDER et al., 2020; HUYNH et al., 2020), alguns necessitam de diversas modificações no algoritmo para se adequarem a cenários dinâmicos do mundo real com quantidade massiva de dispositivos para alocação de recursos em computação de borda.

Os algoritmos meta-heurísticos bioinspirados na natureza são conhecidos por otimizar problemas complexos e dinâmicos do mundo real. Eles simulam o comportamento de algum elemento da natureza e, por isso, são considerados algoritmos inspirados na natureza (JHA; JAIN; THENMALAR, 2018). Tal inspiração considera diversos comportamentos possíveis, como o comportamento de animais durante uma caça ou seu comportamento em grupo, conceitos evolutivos e até fenômenos físicos (MIRJALILI; MIRJALILI; LEWIS, 2014). Além de permitir simular diferentes e propor novos conceitos naturais, os algoritmos bioinspirados são fáceis de manipular para unir dois ou mais comportamentos de elementos de natureza diferentes, o que permite a fácil adaptação dos algoritmos para cenários complexos e dinâmicos (MIRJALILI; MIRJALILI; LEWIS, 2014), como os de computação de borda e borda móvel considerados neste trabalho.

Nesta direção, são propostos dois mecanismos para alocação de recursos IoT para computação de borda e computação de borda móvel: o *i*) mecanismo de otimização do **LOBO** cinzento para alocação de recursos de dispositivos **IoT**, chamado LOBO-IoT, e o *ii*) mecanismo baseado no algoritmo de otimização da **BALE**ia para alocação de recursos **IoT**, chamado BALE-IoT. Desse modo, optou-se por estes dois métodos por considerar que o algoritmo do lobo cinzento seleciona três opções a cada execução (*alpha*, *beta* e *delta*), o que agiliza o processo de alocação dos recursos de um dispositivo, caso a primeira opção não consiga alocar os recursos, e o algoritmo de otimização da baleia apresenta um algoritmo simples e de fácil adaptação aos métodos de simulações propostos.

1.1 Motivação

A massiva quantidade de dispositivos IoT, simultaneamente na rede em cidades inteligentes, necessita de serviços eficientes para atendê-los, considerando a limitação de recursos computacionais dos serviços de borda e serviços de borda móvel. Alocar tais serviços de modo eficiente é um dos desafios citados na literatura.

Na literatura, os trabalhos de Zheng et al. (2015), Ketykó et al. (2016), Zhang et al. (2016), Kumar, Zeadally e Rodrigues (2016), Li et al. (2019), Pereira et al. (2019), Aazam, Harras e Zeadally (2019), Goudos et al. (2019), Jiang et al. (2019), Moses e Kaarthick (2019), Meneguette, Boukerche e Pimenta (2019), Shao et al. (2019), Hosseinioun et al. (2020), Costa et al. (2020), Weerasinghe et al. (2020), Schneider et al. (2020) e Huynh et al. (2020) tratam o problema de alocação de recursos, entretanto, enquanto alguns possuem estruturas complexas para tomadas de decisões em ambientes diferentes, outros não são focados na maximização do atendimento e minimização dos recursos utilizados, com a otimização do compartilhamento de recursos computacionais dos dispositivos, tais como capacidade de processamento, memória, largura de banda, entre outros.

Neste trabalho, busca-se enfrentar o problema de limitações dos recursos computacionais de serviços de borda e serviços de borda móvel, propondo o desenvolvimento dos mecanismos LOBO-IoT e BALE-IoT, baseados em algoritmos meta-heurísticos bioinspirados, com tomadas de decisões eficientes no momento de alocar recursos computacionais e que sejam facilmente adaptados a diversos cenários, a fim de melhorar a utilização de recursos e o atendimento de usuários.

1.2 Objetivos

Este trabalho tem como objetivo geral, propor, desenvolver e avaliar mecanismos para gerenciamento de alocação de recursos de dispositivos IoT em computação de borda e computação de borda móvel, utilizando método meta-heurísticos bioinspirados, que se adaptem melhor ao cenário urbano. Com isso, obtém-se um considerável número de dispositivos, com o intuito de otimizar os recursos disponíveis nas redes, maximizando a utilização de recursos disponíveis e o atendimento dos usuários.

Os objetivos específicos deste trabalho são:

- Desenvolver os mecanismos LOBO-IoT e BALE-IoT para tomada de decisão no processo alocação de recursos de dispositivos IoT, em computação de borda e computação de borda móvel, utilizando algoritmos meta-heurísticos bioinspirados;
- Modelar cenários urbanos com quantidade variada de dispositivos para simulação;

- Realizar comparações com outros trabalhos da literatura para verificar a eficiência dos mecanismos;
- Implementar os algoritmos meta-heurísticos bioinspirados LOBO-IoT e BALE-IoT, utilizando a linguagem de programação Python[1] e o simulador de mobilidade urbana SUMO[2] para realizar as simulações;
- Alcançar eficiência e maximizar o ganho de recompensados dispositivos atendidos;

1.3 Organização do trabalho

A dissertação está organizada do seguinte modo:

No **Capítulo 2**, é apresentada a fundamentação teórica. São apresentadas as tecnologias 5G, computação de borda e computação de borda móvel, além de técnicas de gerenciamento de recursos.

No **Capítulo 3**, é apresentado o estado da arte com recentes trabalhos em alocação de recursos computacionais e suas diferenças para as soluções propostas neste trabalho.

No **Capítulo 4**, é apresentado um cenário urbano com serviço de borda e aplicadas as soluções desenvolvidas para este trabalho, nomeadas como LOBO-IoT e BALE-IoT, criado o cenário para simulação e comparação das métricas para validação deste trabalho e analisados os resultados obtidos.

No **Capítulo 5**, é apresentado um cenário urbano com serviço de borda móvel e aplicadas as soluções desenvolvidas para este trabalho, nomeadas como LOBO-IoT e BALE-IoT, criado o cenário para simulação e comparação das métricas para validação deste trabalho e analisados os resultados obtidos.

No **Capítulo 6**, são apresentadas as conclusões desta dissertação, as contribuições, as produções científicas e sugestão de novas ideias para o desenvolvimento de trabalhos futuros.

[1] https://www.python.org/
[2] https://www.eclipse.org/sumo/

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, é apresentada a fundamentação teórica, em que se descrevem os ambientes utilizados para gerenciamento de recursos de dispositivos IoT. Inicialmente, são apresentadas as tecnologias 5G, o ambiente de computação de borda, que é utilizado na melhoria de comunicação, troca e compartilhamento de recursos em dispositivos IoT e também o conceito de computação de borda móvel, que estende serviços de computação móvel para a computação de borda. Subsequente, é descrito o gerenciamento de recursos e apresentadas as técnicas utilizadas para este serviço.

2.1 Tecnologias e Gerenciamento de Recursos Computacionais

Esta seção abrange as tecnologias 5G, computação de borda e computação de borda móvel, sobre os quais são abordados os seus conceitos e os desafios científicos e tecnológicos para alocação de recursos.

2.1.1 Quinta Geração e Internet das Coisas

Com a evolução da tecnologia nos últimos anos, percebe-se em crescimento que abrange vários ramos tecnológicos, entre eles: a rede de quinta geração (5G) e o serviços de borda. (EJAZ et al., 2016; CHETTRI; BERA, 2020). Os dois conceitos, quando atuam em conjunto, em que procuram interligar diversas espécies de equipamentos, recebem a denominação de Internet das Coisas(IoT). A IoT faz parte do nosso cotidiano, conectando à internet os mais diversos equipamentos físicos, modificando o modo como as pessoas se comunicam, interagem e compartilham conteúdos. (IBM, 2020).

A tecnologia 5G é caracterizada por alguns recursos que são considerados limitados nas redes convencionais (Quarta Geração e anteriores), tais como comunicação massiva de dados, conectividade universal, utilização do espectro, atraso zero e rapidez na troca de informações (PANWAR; SHARMA; SINGH, 2016). Com o 5G, busca-se fornecer à sociedade um novo tipo de conexão, que pode ultrapassar as barreiras de tempo e espaço da atualidade, para conectar diferentes tipos de equipamentos de usuários, sendo o modo de conexão de bilhões de dispositivos com a rede de internet (ZHANG et al., 2016).

Aproximadamente 100 bilhões de equipamentos de usuários estarão interligados através da tenologia 5G, os quais estarão presentes em várias circunstâncias da convivência social das pessoas, como: recreação, profissões, escola, saúde, meios de comunicação sociais, entre outros(FU; CHEN, 2020). Com a alta procura por estes serviços, que tendem a crescer a cada ano, outras dificuldades vem aparecendo, como: fornecer serviços de

qualidade, de modo rápido, atendendo o alto número de dispositivos simultaneamente na rede (LIU et al., 2014).

Nessa perspectiva, pesquisadores cada vez mais buscam melhorar as soluções existentes para acompanhar a emergente evolução tecnológica e enfrentar seus desafios (EJAZ et al., 2016). Segurança, quantidade massiva de dispositivos simultaneamente na rede, baixa latência, gerenciamento e alocação de recursos computacionais, qualidade de serviço e experiência do usuário, são alguns destes desafios (AKPAKWU et al., 2018; CHETTRI; BERA, 2020; XU et al., 2021). O desafio de alocação de recursos abrange uma gama de necessidades importantes para serem gerenciadas, tais como consumo de energia, capacidade de processamento, serviços, largura de banda, espectro, entre outros (LI; XU, 2021). A busca, por otimização destes serviços, abrange o compartilhamento entre os recursos limitados de dispositivos IoT, tais como capacidade de armazenamento e processamento (LI; XU, 2021; XU et al., 2021).

2.1.2 Computação de borda

A computação em nuvem é uma importante descoberta tecnológica da última década que revolucionou os serviços de computação e de rede (DONNO; TANGE; DRAGONI, 2019). Com o aumento expressivo do número de dispositivos que utilizam computação em nuvem, é preciso conseguir atender toda a demanda de serviços de modo eficiente e com baixa latência (DONNO; TANGE; DRAGONI, 2019). Com isso, surge a necessidade dos serviços serem deslocados para mais próximo dos usuários, o paradigma computação de ponta, em que se utilizam as técnicas de serviços de borda (YOUSEFPOUR et al., 2019).

Em busca de solucionar parcialmente esses problemas, foi introduzido um conceito denominado computação de borda (EC) para trabalhar com os equipamentos de usuários de uso constante de recursos computacionais (LI et al., 2018b). Os serviços de borda fornecem uma estrutura a mais de infraestrutura que distribui serviços dos servidores principais às bordas da rede. Desse modo, a capacidade computacional é colocada, em partes, nas extremidades da rede, possibilitando que os recursos computacionais sejam utilizados em locais mais perto dos equipamentos dos usuários (LI et al., 2018b).

Segundo Hassan, Yau e Wu (2019), os principais objetivos da implementação dos serviços de borda são:

- **Aprimorar o administração de dados** para enfrentar as variadas quantidades de dados sensíveis a latência gerados por dispositivos de usuários, que precisam ser executados simultaneamente.
- **Aprimorar a qualidade de serviço** de modo que atenda diversas espécies de

exigências de qualidade de serviço que procuram aperfeiçoar a experiência do usuário na rede (HU M. PATEL; YOUNG, 2015).

- **Prever a demanda de rede** para avaliar os recursos essenciais para servir a demanda de rede em uma ponto próximo do usuário, possibilitando uma alocação de serviços melhor para preencher esta necessidade, ajudando na decisão de onde a demanda será atendida, na extremidade ou na rede central.
- **Gerenciar e ter conhecimento das localizações das bordas** para que os servidores das extremidades consigam rastrear equipamentos de usuários, possibilitando mapear informações sobre locais de interesse próximos.
- **Aperfeiçoar o gerenciamento de recursos computacionais** para otimizar o compartilhamento de recursos da rede, devido à limitação de recursos que os servidores de borda possuem.

Embora esta metodologia apresente diversos ganhos quanto à latência, ela deu origem a um novo desafio, que é lidar com a alocação dos recursos computacionais, disponíveis nas bordas da rede (LI et al., 2019). As bordas devem ser equipadas com mecanismos capazes de prover um gerenciamento e compartilhamento eficiente de recursos para alcançar uma alocação de recursos globalmente eficaz (XU et al., 2017).

2.1.3 Computação de borda móvel

A tecnologia de computação de borda móvel (*Mobile Edge Computing* - MEC) é considerada por pesquisadores como um auxílio à computação de borda que ajuda a minimizar ainda mais a latência e melhorar a conexão da largura de banda durante processos de comunicação e descarregamento de dados (MACH; BECVAR, 2017). A MEC disponibiliza os recursos computacionais em um ambiente na borda do serviço de nuvem e dentro da rede de acesso de rádio, permitindo que os dispositivos troquem informações em tempo real dentro da rede de rádio (SABELLA et al., 2016). A MEC é implementada de modo virtualizado, utilizando virtualização de funções de rede, redes centradas com informações e redes definidas por softwares (MAO et al., 2017).

Os benefícios das MECs vão além do fornecimento de serviço de qualidade ao usuário final. Elas permitem que as operadoras disponibilizem seu serviço de borda de RAN para terceiros devidamente autorizados, possibilitando que se implementem aplicativos e novos serviços para oferecer aos usuários de dispositivos móveis, empresas e outros interessados, de forma fácil e dinâmica (SABELLA et al., 2016). Além disso, no cenário para dispositivos móveis e IoT, os quais possuem capacidade limitada de recursos computacionais, como bateria, memória, processamento, entre outros, as MECs são importantes

para receber o descarregamento de dados dos equipamentos dos usuários, liberando recursos de hardwares e computacionais aos dispositivos (SABELLA et al., 2016).

No entanto, pesquisadores buscam alternativas para resolver alguns desafios encontrados com a evolução da MEC, tais como: *i*) conseguir gerenciar quando e para onde os dados descarregados em uma MEC serão migrados, para que fiquem mais próximo dos usuários, mantendo a qualidade da experiência do usuário na rede. Tal problema é frequente devido a alta mobilidade dos usuários (ZHANG; ZHENG, 2019); *ii*) alocar recursos de dispositivos IoT de modo eficiente e satisfatório com as dificuldades encontradas em um ambiente mutável e complexo de MECs (WANG et al., 2019; HUI et al., 2019); *iii*) limitação de recursos computacionais dos dispositivos IoT que necessitam executar tarefas com computação intensa e baixa latência, em tempo real (WANG; ZHOU, 2019; WANG et al., 2020; ZHAO et al., 2020); *iv*) encontrar políticas eficientes de otimização para alocação de recursos em ambientes dinâmicos (LI et al., 2018a); *v*) entre outros.

2.2 Métodos para gerenciamento de recursos

As redes de celulares 5G mantém conectividade de forma contínua e precisam transmitir uma alta quantidade de dados para atender os requisitos de qualidade de serviço para a diversificada gama de dispositivos que se conectam simultaneamente (HUSSAIN et al., 2020). Além disso, necessita de dar suporte a vários aplicativos de IoT disponibilizados/compartilhados na rede e, assim, é preciso gerenciar os recursos utilizados pelos dispositivos (HUSSAIN et al., 2020). O gerenciamento de recursos computacionais tem o objetivo de organizar os processos dos serviços utilizados, buscando melhorar a qualidade do serviço e a qualidade de experiência do usuário (WAN et al., 2020). Os serviços de computação em nuvem são responsáveis por disponibilizar recursos computacionais sob demanda, onde ficam encarregados pelo processamento e compartilhamento de recursos configuráveis como rede, armazenamento, processamento, aplicações, serviços, energia, entre outros (WAN et al., 2020).

Com a tecnologia 5G e os dispositivos IoT, é necessário que os serviços de rede e dispositivos se comuniquem de maneira rápida e eficiente, para garantir a qualidade de serviço e boa experiência do usuário (PEREIRA et al., 2020). A EC é uma alternativa para resolver estes problemas. Entretanto, como seus serviços são disponibilizados em nuvens de redes de acesso de rádio mais próximas dos usuários, por meio de máquinas virtuais, a EC possui poder computacional limitado para atender a quantidade simultânea e massiva de dados complexos de última geração (WAN et al., 2020). Pesquisadores buscam alternativas de melhoria nos processos de compartilhamento de dados entre dispositivos e EC. Neste capítulo são apresentados métodos tradicionais e meta-heurísticos, utilizados por pesquisadores em processos de alocação de recursos.

2.2.1 Métodos tradicionais

Nesta seção, são apresentados três conceitos que tradicionalmente são utilizados para otimização de alguns processos de alocação de recursos computacionais: o problema da mochila, o processo de decisão de Semi-Markov e a árvore de decisão. Optou-se por estes conceitos, porque foram utilizados recentemente por pesquisadores para processos de alocação, conforme trabalhos apresentados no estado da arte.

O problema da mochila é um método que objetiva conseguir carregar os compartimentos de uma mochila com os objetos mais valiosos disponíveis (MOSES; KAARTHICK, 2019). Considera-se que cada objeto tenha um valor de recompensa e deve-se escolher quais serão colocados na mochila obtendo o maior valor de ganho possível. O método tradicional é considerado um problema de mochila 0-1, pois cada objeto é indivisível e não pode ser fracionado para colocar na mochila (MOSES; KAARTHICK, 2019). O objetivo principal do processo é maximizar as recompensas, sem exceder o limite de recursos disponíveis. O problema é considerado uma otimização discreta que pode ser aplicado em diversos cenários (WANG; ZHENG; WANG, 2013).

Alguns trabalhos da literatura (WANG; TAN; LIU, 2018; JIHAD et al., 2019; SUN et al., 2020; COSTA et al., 2020; LIU; LIU; PENG, 2021) utilizam a mochila como base para alocação de recursos computacionais, no entanto, é necessário adaptar o conceito da mochila para cada cenário. Para representar o modelo matemático de uma aplicação do conceito da mochila, considera-se o trabalho de (COSTA et al., 2020), no qual os autores utilizaram o problema da mochila para alocação de tarefas computacionais em nuvens veiculares. Considera-se cada tarefa como um item e cada nuvem veicular como uma mochila e busca-se alocar o máximo de tarefas, com maior recompensa possível, e minimizar a utilização de recursos. A Equação 2.1 apresenta o conceito aplicado:

$$maximizar \sum_{k=1}^{n} g_k t_k$$

$$sujeito\, a \sum_{k=1}^{n} \omega_k t_k \leq \Omega_j, e\, t_k \in \{0,1\} \tag{2.1}$$

Em que g_k representa o ganho de cada tarefa t_k, que aumenta uniformemente com base no seu peso ω_k, e Ω_j a quantidade total de recursos de cada nuvem veicular.

O Processo de Decisão Semi-Markov (*Semi-Markov Decision Process* - SMDP) é um modelo matemático que utiliza uma decisão multicritério para definir seus modelos de tomadas de decisão. Por meio dele se considera tanto valores de recompensas imediatas atribuídas para cada tarefa, quanto o ganho de valor durante a execução do processo para tomar decisões (LOLOS et al., 2017b). Assim, é considerado um algoritmo de aprendizado

por reforço com solução natural, que utiliza dados de ações anteriores para definir os próximos estados (LOLOS et al., 2017a).

O SMDP é composto por 4 estágios (SHI et al., 2015):

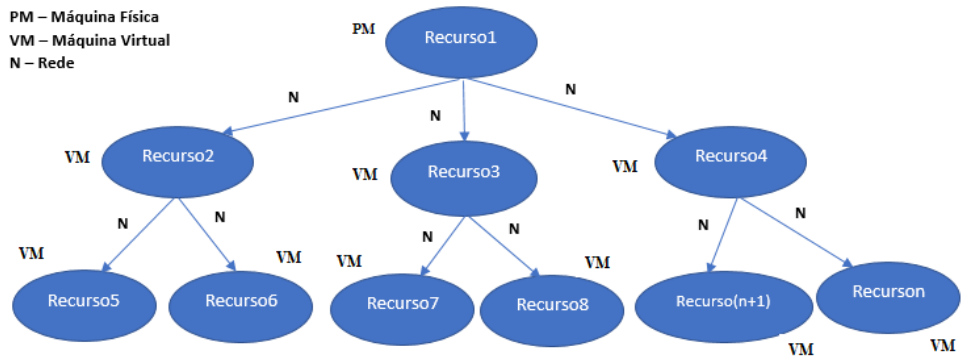
- **Estado:** um estado representa como um processo está atualmente. Seu estado atual pode ser modificado por uma ação;
- **Ação:** uma ação representa as operações possíveis que podem ser realizadas sobre um conjunto de ações;
- **Transição:** representa o processo de mudança que um estado terá durante a execução de uma ação;
- **Recompensa:** é um valor recebido por um estado logo após o término da execução de uma ação.

O processo estabelece uma política de busca a uma estratégia ótima durante a iteração de um algoritmo de programação dinâmica, em que se considera o longo prazo do processo, equilibra os fatores de custo e orienta a política da decisão (SHI et al., 2015; LOLOS et al., 2017b). No entanto, a sobrecarga durante os processos é um dos principais problemas do SMDP (SHI et al., 2015).

Outro conceito utilizado por pesquisadores para gerenciamento de recursos é a Árvore de Decisão, em que se utilizam técnicas de aprendizagem de máquina para tomadas de decisão para alocação de recursos de um dispositivo. Os algoritmos de aprendizagem de máquinas são úteis quando utilizados em um ambiente com massiva quantidade de dados, pois conseguem fornecer resultados de alta precisão (JIANG et al., 2017). Árvores de decisão trabalham com o conceito de regressão, que divide um conjunto de dados em subconjuntos e vai repetindo o processo com base nas características aprendidas com os dados anteriores (SCHNEIDER et al., 2020).

Na Figura 1 apresenta-se um exemplo de uma árvore de decisão com diversos recursos de máquinas virtuais aplicadas no trabalho de Kumar, Zeadally e Rodrigues (2016) para determinar um esquema de agendamento para migração de máquinas virtuais. O tamanho da árvore depende da alocação de máquinas virtuais. O escalonador global proposto é responsável pelas principais decisões de migração. Primeiro, são identificados os *hosts* para migração. Com base neles, uma lista inicial de máquinas com sobrecarga é configurada usando a árvore de decisão. Assim, o gerenciador baseia-se nessa lista para tomar as decisões (KUMAR; ZEADALLY; RODRIGUES, 2016).

Figura 1 – Representação da árvore de decisão.



Fonte: Kumar, Zeadally e Rodrigues (2016), adaptado pelo autor.

2.2.2 Métodos Meta-heurísticos para Decisão de Alocação de Recursos e Funções de aptidão

Conforme citado anteriormente, muitos métodos são utilizados para proporcionar qualidade de serviço aos usuários durante o processo de gerenciamento de recursos em uma nuvem. Entre elas, técnicas de otimização que seguem modelos matemáticos, heurísticos, meta-heurísticos, de aprendizado, entre outros. Nesta seção, é descrito o método meta-heurístico como técnica de decisão para alocação de recursos em EC.

O método meta-heurístico é definido em ciência da computação e otimização matemática como um método de nível superior para encontrar, gerar ou selecionar um algoritmo de pesquisa parcial que pode fornecer um resultado eficientemente otimizado para um problema específico (JHA; JAIN; THENMALAR, 2018). Em ciência da computação e otimização matemática, essas técnicas são definidas como métodos de alto nível para encontrar, gerar ou selecionar uma heurística, que podem proporcionar resultados satisfatórios e otimizados (JHA; JAIN; THENMALAR, 2018).

Os algoritmos meta-heurísticos bioinspirados são algoritmos inspirados na natureza, pois representam o modo como um elemento da natureza se comporta. Estes mecanismos são conhecidos por ter a resposta para otimizar considerando a complexidade e dinamicidade de um ambiente do mundo real (JHA; JAIN; THENMALAR, 2018; POL; PACHGHARE, 2019; MIRJALILI; MIRJALILI; LEWIS, 2014). As inspirações são relacionadas a acontecimentos físicos da natureza, reações de animais ou conceitos evolutivos. A simplicidade dos algoritmos ajudam pesquisadores da área de tecnológica a simular diversificados conceitos naturais e desenvolver novas metodologias meta-heurísticas, propor métodos híbridos com dois ou mais algoritmos, ou aperfeiçoar mecanismos já desenvolvidos (MIRJALILI; MIRJALILI; LEWIS, 2014).

Neste trabalho, são apresentados três métodos meta-heurísticos utilizados para otimização de processos: o algoritmo de inteligência de enxame de Otimização por Enxame de Partículas, os algoritmos bioinspirados de Otimização do Lobo Cinzento e o Algoritmo

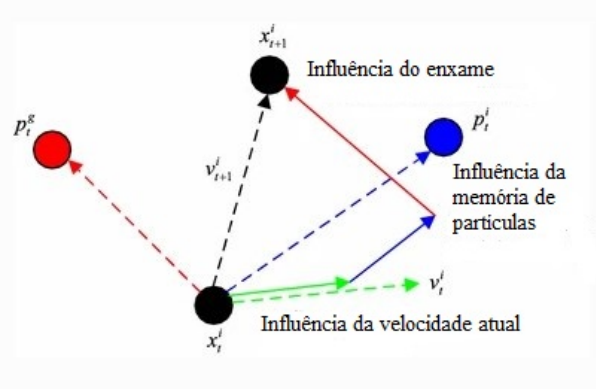
de Otimização da Baleia. Além disso, são apresentadas as funções de aptidão utilizadas neste trabalho.

O algoritmo meta-heurístico de inteligência de enxame, chamado Otimização por Enxame de Partícula (*Particle Swarm Optimization* - PSO), é uma técnica de otimização estocástica que utiliza como base para simulação a migração de animais inteligentes na natureza, como pássaros, rebanhos e peixes, e considera a contínua iteração das partículas para encontrar a solução ótima (CHEN et al., 2020). Cada elemento é considerado uma partícula e representa uma opção candidata como possível solução para o resultado do processo de otimização (WANG; TAN; LIU, 2018).

Na literatura, o PSO é comumente utilizado por ser um algoritmo simples e fácil de ser aplicado em diversos cenários, além de ter uma taxa de convergência baixa e não necessitar de funções otimizadas diferencial, derivada e contínua. O PSO não direciona sua busca diretamente no procedimento de cálculo, mas nas informações dos ótimos do enxame e dos ótimos individuais, obtidas a cada iteração do algoritmo (WANG; TAN; LIU, 2018).

O mecanismo do PSO trabalha com duas versões chamadas de versão global e versão local. Na primeira, o algoritmo rastreia as posições *pbest* e *gbest*, que são os extremos da posição ótima de si mesma e do enxame, respectivamente. Enquanto que na segunda, ele rastreia o *pbest* e a posição ótima de todas as partículas em sua vizinhança, chamado de *nbest*. O mecanismo de iteração de qualquer partícula durante a execução do algoritmo é ilustrado na Figura 2.

Figura 2 – Mecanismo de iteração do PSO.



Fonte: Wang, Tan e Liu (2018), traduzido pelo autor.

A primeira parte considerada pelo mecanismo é a velocidade da partícula anterior, que utiliza seu movimento atual e gera um movimento inercial considerando sua velocidade atual. A segunda é chamada de fator de aceleração cognitiva, pois realiza o movimento, a partir de sua própria experiência, e considera a distância da sua posição atual e sua *pbest*. A terceira parte é o fator de aprendizagem social, que considera o compartilhamento e cooperação entre os elementos para definir seus movimentos (WANG; TAN; LIU, 2018).

O método de Otimização do Lobo Cinzento (*Grey Wolf Optimizer* - GWO) se baseia na técnica de otimização denominada algoritmo do lobo cinzento. Os lobos cinzentos são considerados predadores do topo da cadeia alimentar e preferem viver em bando, em média de 5 a 12 lobos. A hierarquia da alcateia de lobos cinzentos é definida como uma pirâmide de quatro níveis: *alpha*, *beta*, *delta* e *omega* (MIRJALILI; MIRJALILI; LEWIS, 2014).

O *alpha* α está no topo da pirâmide, o que o torna o líder e o responsável por tomar as decisões do grupo, entre elas a decisão da caça. Em seguida, temos o *beta* β, que está um nível abaixo do líder e é responsável por auxiliar o líder nas tomadas de decisões da alcateia. No terceiro nível da pirâmide temos o *delta* δ, que é considerado o responsável pela segurança da alcateia, reporta aos dois primeiros níveis e é o responsável pelos lobos do último nível da alcateia. Os demais lobos da alcateia são considerados os *ômegas* ω e ficam no último nível da pirâmide (MIRJALILI; MIRJALILI; LEWIS, 2014).

Além da hierarquia dos lobos cinzentos, é interessante o comportamento da alcateia durante uma caça. A técnica de caça é agrupada em três etapas, descrito por Mirjalili, Mirjalili e Lewis (2014), como:

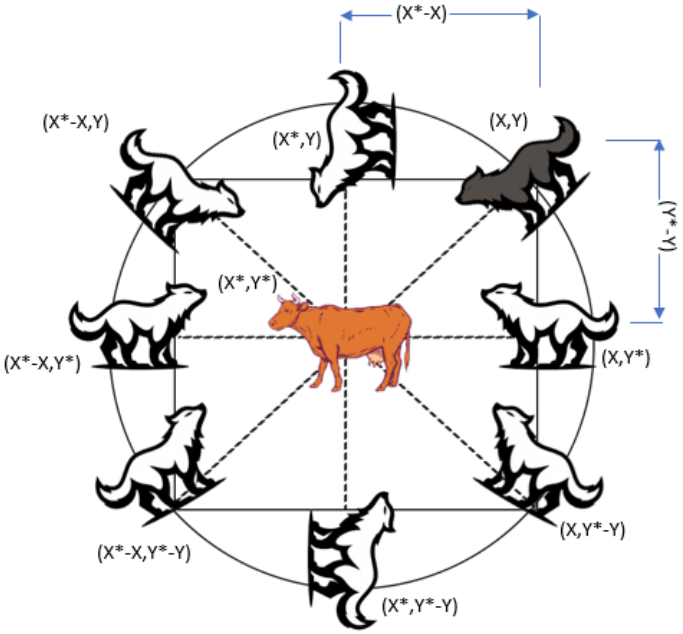
- Rastrear, perseguir e se aproximar da presa;
- Perseguir, cercar e assediar a presa até que ela pare de se movimentar;
- Atacar em direção à presa.

A primeira etapa é identificar quais são as três melhores tarefas do conjunto. Considera-se a melhor solução como o *alpha* (α). E, seguindo a ordem da pirâmide, a segunda e a terceira soluções são definidas como *beta* (β) e *delta* (δ), respectivamente. O restante das soluções são definidos como *ômegas*. Para a implementação do algoritmo de otimização GWO, a caça é guiada pelas soluções α, β e δ.

Pode-se observar o movimento dos lobos durante a realização do cerco na Figura 3, em que se denota a mudança de posicionamento de um lobo (X, Y), utilizando como referência a posição da presa (X* ,Y*). A busca por alcançar a melhor posição em torno do melhor agente é feita com a atualização de coeficientes que são definidos em valores aleatórios de 0 a 1. Na Figura 4, é possível observar como é realizado o cerco e a movimentação dos lobos entorno da presa.

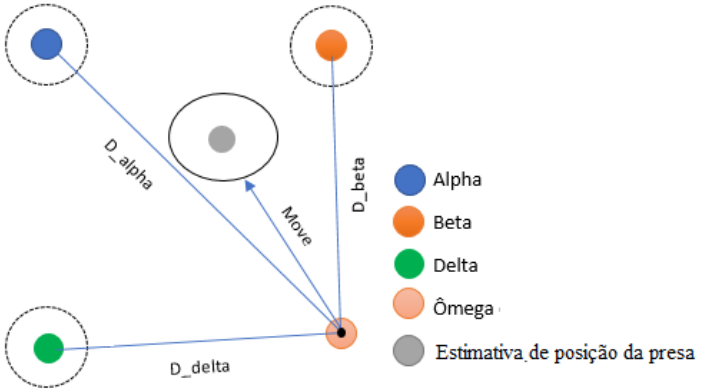
Feito o cerco, a alcateia de lobos cinzentos parte ao ataque, fazendo o reposicionamento dentro do cerco realizado. A caça é guiada pelo lobo *alpha*. O *beta* e o *delta* também participam diretamente do reposicionamento. Como normalmente não se tem o posicionamento ótimo da presa para simular matematicamente o comportamento de caça, considera-se, dessa forma, que os três primeiros tenham conhecimento do posicionamento

Figura 3 – Vetor de posições 2D e suas possibilidades de próximas posições.



Fonte: (MIRJALILI; MIRJALILI; LEWIS, 2014), adaptado pelo autor.

Figura 4 – Cerco feito pelo GWO durante a caça.



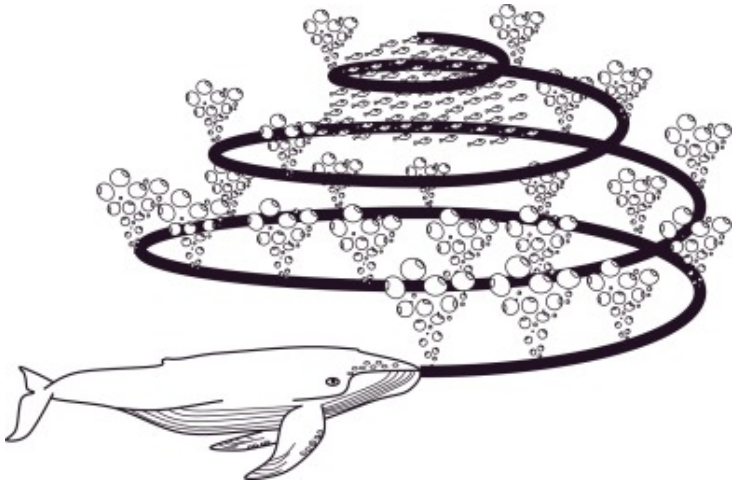
Fonte: (MIRJALILI; MIRJALILI; LEWIS, 2014), adaptado pelo autor.

da presa. Então, são selecionados os três melhores serviços de acordo com sua aptidão e definidos o *alpha*, sendo o de melhor solução. Além dos elementos *beta* e *delta*, sendo a segunda e a terceira melhor solução, respectivamente.

O Algoritmo de Otimização da Baleia (*Whale Optmization Algorithm* - WOA) é outra técnica de otimização meta-heurística bioinspirada, que foi desenvolvida por Mirjalili e Lewis (2016), com base no comportamento da baleia jubarte. Os autores supõem que as baleias possuem alto nível de inteligência sendo capazes de pensar, aprender e julgar como os humanos. Assim, a baleia jubarte foi escolhida para o desenvolvimento do processo de otimização pelo seu interessante comportamento de caça. O comportamento social das baleias pode ser individual ou em grupo, no entanto, é comum encontrar as baleias em grupos (MIRJALILI; LEWIS, 2016; GHAREHCHOPOGH; GHOLIZADEH, 2019).

A baleia jubarte tem por preferência caçar pequenos peixes *krill* que ficam na superfície do mar. Para isso, utilizam uma técnica especial de caça denominada de método para alimentação com rede de bolhas. Esse método de forrageamento é feito quando as baleias nadam ao redor da presa, criando um círculo de bolhas em formato de um '9', denotado na Figura 5. Isso é feito por meio de duas manobras chamadas de "espirais para cima"e "loops duplos". Assim, uma baleia tenta encontrar uma nova posição no espaço de busca, utilizando como referência o melhor elemento do grupo (MIRJALILI; LEWIS, 2016).

Figura 5 – Círculo de bolhas feito pela baleia jubarte no processo de caça.



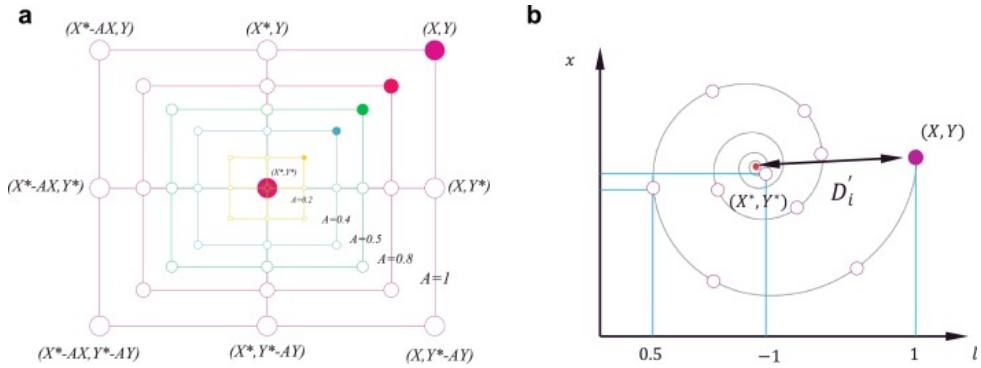
Fonte: (MIRJALILI; LEWIS, 2016).

O processo passa por duas abordagens que foram feitas para desenvolver o modelo matemático: *i*) mecanismo de encurtamento; *ii*) posição de atualização na espiral, conforme Figura 6. A localização da presa, no método de ataque por rede em bolhas, é feita inicialmente pelo encurtamento, em que são definidos coeficientes responsáveis pela aproximação da baleia localizada (X, Y) e a presa alvo (X*, Y*). Na Figura 6 (a)

apresenta-se prováveis posições da movimentação das baleias em um espaço 2D (GHAREHCHOPOGH; GHOLIZADEH, 2019).

No mecanismo de atualização na espiral, denotado pela Figura 6 (b), primeiro é calculada a distância do predador (X, Y) e da presa (X*, Y*), para, então, se movimentar em direção à presa no formato de uma hélice. Assim, os predadores nadam em torno da presa dentro de um círculo que vai se afunilando a cada iteração. Outra alternativa utilizada, além do método de busca por bolha, é a busca por presas de forma aleatória, em que o algoritmo inicia com um conjunto de soluções aleatórias e utilizam as posições em relação a um elemento aleatório ou a melhor solução definida até aquela iteração (MIRJALILI; LEWIS, 2016).

Figura 6 – Cerco e movimentação feitos pela baleia jubarte.



Fonte: (MIRJALILI; LEWIS, 2016).

Pela capacidade de exploração e aproveitamento durante os processos de evolução, o algoritmo WOA pode ser considerado um otimizador global (MIRJALILI; LEWIS, 2016). Além disso, Mirjalili e Lewis (2016) a fim de minimizar a complexidade do algoritmo, optaram por diminuir a quantidade de heurística e de parâmetros internos durante a reprodução do comportamento das baleias jubarte.

Para definição das melhores soluções nos algoritmos meta-heurísticos, utiliza-se funções de aptidão. Para cálculo das aptidões são utilizadas funções *benchmarks*, que são funções matemáticas com o propósito de encontrar uma solução mínima (ótimo global) durante as iterações definidas nos algoritmos (WONG; MING, 2019). Neste trabalho, os valores atribuídos à população de agentes de cada algoritmo são baseados nos recursos computacionais dos dispositivos ou dos serviços de borda, dependendo da simulação, e utilizados para os cálculos das aptidões. Optou-se por analisar três funções *benckmarks* que são consideradas eficientes para utilização em algoritmos meta-heurísticos bioinspirados (GARDEN; ENGELBRECHT, 2014; VALLURU; SINGH, 2018; YANG; HE; FAN, 2020). A seguir, serão apresentadas as funções analisadas:

- **Função de Rosenbrock:** é utilizado para aplicação de teste convencionais em

muitos algoritmos e utiliza a Equação 2.1 (VALLURU; SINGH, 2018);

$$Ros(x) = \sum_{i=1}^{d-1} \left[100 \left(x_{i+1} - x_t^2 \right)^2 + \left(x_i - 1 \right)^2 \right] \tag{2.1}$$

Em que x é o conjunto de recursos da população do algoritmo utilizado, e i corresponde ao recurso específico da iteração que o algoritmo está executando.

- **Função de Rastrigin:** é aplicada em problemas de teste de desempenho em otimização de algoritmos (QIAO; YANG, 2019). A aptidão em Rastrigin é calculada de acordo com a Equação 2.2;

$$Ras(x) = 20 + X_1^2 + X_2^2 - 10(Cos2\pi X_1 + Cos2\pi X_2) \tag{2.2}$$

Em que X corresponde ao conjunto de recursos da população da iteração atual;

- **Função de Ackley:** é considerada uma função de otimização clássica (LIANG; SUGANTHAN; DEB, 2005) e utiliza a Equação: 2.3.

$$\begin{aligned} Ackley = & -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) \\ & - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e \end{aligned} \tag{2.3}$$

Em que n representa os recursos da população do algoritmo, x representa o valor do recurso da iteração atual e e um valor exponencial atribuído ao valor padrão da função.

Tais mecanismos foram utilizados, neste trabalho, para analisar o processo de tomada de decisão nos dois cenários urbanos propostos.

2.3 Considerações parciais

Com a rápida evolução dos dispositivos de telecomunicações, as tecnologias 5G, IoT e a quantidade massiva de usuários que utilizam a rede constantemente, fez-se necessário utilizar os conceitos de computação de borda e computação de borda móvel para deixar as aplicações mais próximas dos usuários e minimizar o tempo de comunicação. No entanto, a descentralização dos serviços para ECs e MECs, limitam a quantidade de recursos computacionais. Assim, faz-se necessário encontrar métodos que auxiliem no gerenciamento eficiente de recursos para ECs e MECs. Pode-se observar que pesquisadores,

constantemente, buscam alternativas eficientes para melhorar a qualidade do serviço e a qualidade de experiência do usuário em processos de gerenciamento de recursos de redes.

Técnicas heurísticas e processos de aprendizado de máquina já são aplicados, no entanto, ainda encontram algumas limitações como complexidade dos algoritmos, tempo de inferência e adaptação a cenários reais, os quais estão se tornando mais complexos devido à quantidade massiva de dados que aumentam anualmente. Em relação aos métodos meta-heurísticos, encontrados na literatura, pode-se observar que tais métodos são utilizados para melhorar processos em diversos ambientes, em que é possível adaptar facilmente as técnicas para o cenário desejado. Portanto, considera-se que os métodos meta-heurísticos podem ser utilizados no cenário urbano para alocação de recursos de dispositivos IoT em computação de borda e computação de borda móvel, adequando os métodos dos conceitos para encontrar a melhor alternativa de alocação e, assim, otimizar a utilização dos recursos computacionais dos serviços disponibilizados.

3 ESTADO DA ARTE

Neste capítulo, são apresentados trabalhos com diversas técnicas para alocação de recursos computacionais. Cada trabalho utiliza ambientes diferentes de estudos, focados na melhoria de recursos específicos para cada cenário aplicado, porém todos têm o propósito de otimizar processos de gerenciamento de recursos computacionais.

3.1 Trabalhos relacionados

Li et al. (2019) apresentam uma proposta de otimização para redes IoT baseada em serviço de Fog com NOMA (*Non-Orthogonal Multiple Access*), com suporte para que os dispositivos descarreguem suas tarefas de forma simultânea. Assim, aplicou-se um algoritmo genético aprimorado para resolver os problemas de alocação de recursos, com o objetivo de minimizar o consumo de energia do sistema durante a decisão de descarregamento, a alocação dos recursos e a potência de transmissão. Entretanto, o foco do trabalho foi analisar a taxa de transferência, latência, probabilidade de interrupção e consumo de energia. No artigo de Aazam, Harras e Zeadally (2019) foi proposto um modelo matemático baseado em qualidade de experiência para analisar o impacto na estimativa de recursos em Fog, considerando aplicações táteis 5G, voltadas para IoT no setor industrial. Assim, foi desenvolvido um algoritmo de estimativa de razão para experiência do usuário, com a linguagem de programação Java, para avaliar a qualidade de experiência do sistema por meio do *Net Promoter Score* (NPS). Porém, a avaliação do algoritmo considerou como base apenas o impacto da estimativa dos recursos. Além disso, o algoritmo utilizado necessita de adequações em caso de mudança de cenário.

Choo, Kim e Pack (2018) utilizaram um algoritmo guloso como base para auxiliar no processo de tomada de decisão. A política decide em que computação de borda irá descarregar as tarefas computacionais as tarefas computacionais com o objetivo de maximizar as tarefas alocadas. O resultado mostrou ser mais eficientes do que técnicas convencionais utilizadas na comparação. O trabalho, porém, não avalia as métricas de número de recursos negados, bloqueados, atendidos, nem o ganho de alocação dos serviços atendidos. Pereira et al. (2020) propuseram uma política de alocação de recursos baseada no processo analítico hierárquico, no contexto de nuvem veicular, em ambientes rodoviários denominado Reliable. Por meio do método matemático de decisão de múltiplos atributos, o Reliable decide se há recursos disponíveis para atender o serviço solicitado pelo veículo. Contudo, o algoritmo do Reliable necessita de várias adequações, caso mude a quantidade de MECs ou dispositivos analisados.

Os algoritmos meta-heurísticos demonstram ser eficientes quando aplicados para

solucionar diversos problemas em redes 5G, conforme trabalhos a seguir. No trabalho de Goudos et al. (2019), utilizou-se o algoritmo meta-heurístico Jaya para fazer um híbrido com o algoritmo de evolução diferencial auto-adaptativa jDE, criando o Jaya-jDE, para reduzir o consumo de energia respondendo à demanda de taxa de bits instantânea pelo usuário. A solução demonstrou ser eficiente quando comparada a outros algoritmos no consumo de energia para configurações de redes 4G (*fourth generation*) e, principalmente, 5G. Porém, o foco do trabalho é no consumo de energia para comparação entre os tipos de redes 4G, 5G *Massive Mimo* e 5G *Massive Mimo* estendido.

Huynh et al. (2020) apresentaram um algoritmo meta-heurístico para reduzir o custo computacional, tempo de conclusão e consumo de energia em ECs, utilizando a otimização por enxame de partículas, denominado JROPSO. O algoritmo é inicializado com soluções candidatas aleatórias (partículas). A cada iteração, a posição das partículas no enxame muda dentro do espaço de busca, a fim de encontrar a melhor solução de acordo com a melhor posição individual e de todas as partículas. As simulações realizadas pelos autores confirmaram a eficiência de métodos meta-heurísticos para problemas de otimização quando comparado a outros métodos. Entretanto, o trabalho visa avaliar a quantidade de sobrecarga de computação, consumo de energia e tamanho dos dados.

Na mesma linha de pesquisa, Jiang et al. (2019) desenvolveram um algoritmo meta-heurístico para escalonamento de tarefas em ECs, baseado no algoritmo de otimização GWO, denominado IBGWO. Os resultados demonstraram que o IBGWO obteve resultados superiores ao algoritmo de morcego binário e a otimização de enxame de partículas, sendo esses dois outros algoritmos meta-heurísticos disponíveis na literatura. Todavia, o trabalho não foca na quantidade de serviço atendido e na maximização da utilização dos recursos disponíveis. Ele analisa a estabilidade, acurácia e rápida convergência do algoritmo proposto.

Hosseinioun et al. (2020) buscaram minimizar o consumo de energia durante o processamento em agendamento de tarefas em serviços de computação em nuvem. Para isso, desenvolveram um algoritmo híbrido de Otimização e Cultura de Ervas Daninhas Invasivas. A solução apresentou resultados melhores quando comparados com algoritmos heurísticos e outros tradicionais. Contudo, os autores focaram na avaliação de consumo de energia e índice de economia de energia durante o agendamento das tarefas.

Em Shao et al. (2019), os autores propuseram um sistema de gerenciamento de replicação dinâmico, um planejador específico de baixo custo para colocação de dados, um sistema de verificação e algumas ferramentas de segurança de dados para computação de borda colaborativa e sistemas de computação em nuvem. Consideraram problemas como dependência de tarefas, confiabilidade no compartilhamento de dados, agendamento de dados para fluxo de trabalho como um problema de computação inteiro. Por fim, propuseram também um algoritmo meta-heurístico mais rápido para solução desse problema.

O algoritmo proposto apresentou uma performance superior a outros considerados tradicionais, sendo capaz de reduzir os custos totais de acesso a dados. Mas, o objetivo do trabalho foi analisar o custo de acesso de dados, volume total de transmissão de dados, números de movimentação de dados e violação de trabalhos.

No trabalho de Ketykó et al. (2016), utilizou-se o problema da mochila para avaliar um modelo de descarregamento multiusuário, em que se considera a latência computacional de ponta a ponta de aplicativos MEC. O problema da mochila foi aplicado considerando dois cenários: compartilhamento de recursos elástico e compartilhamento de recursos dedicado. Na avaliação, utilizaram o problema da mochila com um algoritmo exato (Mulknap) e o algoritmo de otimização de entropia cruzada, cujo algoritmo heurístico apresentou menos tempo de execução. Entretanto, o foco principal do trabalho é minimizar a latência e não foca na maximização de atendimento dos usuários e utilização dos recursos disponíveis.

Costa et al. (2020) aplicaram o problema da mochila para alocação de tarefas em um ambiente de redes veiculares, em que os veículos compartilham seus recursos não utilizados, formando um *cluster*. Assim, são atribuídos um peso e uma recompensa para cada tarefa. Quando um conjunto de tarefas necessita de recursos, o algoritmo CRATOS verifica se há disponibilidade e qual tarefa deverá ser atendida. O CRATOS apresentou ser mais eficiente que o método Greedy e próximo do algoritmo de programação linear inteira, que é um ótimo global. Apesar de buscar maximizar a utilização de recursos e tarefas atendidas, o algoritmo desenvolvido pelos autores precisa de muitas alterações, caso mude o cenário de aplicação.

Na literatura, encontram-se também várias abordagens que utilizam como base o SMDP em processos de alocação de recursos, como no trabalho de Weerasinghe et al. (2020), em que propuseram um algoritmo baseado no SMDP para alocar de forma dinâmica recursos 5G em *slot* de dispositivos sem concessão. Utilizou-se como parâmetros uma confiabilidade específica ou uma prioridade. Assim, o modelo proposto estima o número de dispositivos que chegam e executam uma ação, em cada ciclo, para alocar um número ótimo de recursos. O trabalho tem o objetivo de analisar a quantidade e recompensa de alocação, mas é direcionado para alocação em *slots*, além de ter um algoritmo fixo para o cenário estudado.

Meneguette, Boukerche e Pimenta (2019) aplicaram a técnica do SMDP para realizar alocação dinâmica de recursos em sistemas de transportes inteligentes e desenvolveram o mecanismo denominado AVARAC, com o objetivo de encontrar a solução ótima para agregação e alocação de recursos. O estado de sistema, considerado para o problema proposto, consiste no número de requisições aceitas, número de veículos compartilhando seus recursos na rede e um conjunto de eventos que podem ocorrer no algoritmo. O trabalho apresentou um resultado significativo no ganho de recompensa atribuído e uma boa

tomada de decisão para alocar os recursos. Apesar de analisar o ganho de recompensa e quantidade de serviços atendidos, o AVARAC necessita de várias alterações, caso mude o cenário de aplicação.

Baseado na utilização de árvores de decisão para alocação de recursos, Schneider et al. (2020) propuseram um método para evitar a super e sublocação de recursos em funções de redes virtuais (*Virtual Network Function* - VNF). Eles desenvolveram um método para definir perfis de desempenho úteis, que modelam e preveem com eficiência os requisitos de recursos de VNF. Aplicaram esses perfis criados dentro de algoritmos de posicionamento de VNF que melhoram a alocação dinâmica de recursos. Contudo, o objetivo principal dos autores foi resolver os problemas de requisitos de recursos da VNF, limitações de capacidade da rede e minimizar a latência.

No trabalho de (KUMAR; ZEADALLY; RODRIGUES, 2016), verificou-se o uso de redes veiculares tolerantes a atraso para o problema de disseminação de dados de vários dispositivos, utilizando computação móvel de ponta. Na proposta, fez-se uma abordagem de migração de máquina virtual em tempo real usando previsão de carga, para minimizar o consumo de energia no servidor central. Para utilizar os recursos de modo eficiente em migração de máquina virtual no servidor central, dois aspectos são considerados: carga local e carga global. A carga local é baseada no consumo de processador, memória e recursos de entrada e saída. A carga global considera o status de todas as máquinas físicas da rede. A carga de comunicação utiliza largura de banda, número de canais e interferência de canais vizinhos. Entretanto, o trabalho foi focado na melhoria do atraso de transmissão de mensagens, tempo de resposta e taxa de transferência.

Na tabela 2 resume-se os trabalhos do estado da arte, listados acima, com relação ao método de algoritmo utilizado, tipo de estrutura considerada na aplicação, objetivo principal considerado e problemas, os quais se buscaram resolver. Pode-se observar que os trabalhos de (LI et al., 2019; AAZAM; HARRAS; ZEADALLY, 2019; ZHANG et al., 2016; GOUDOS et al., 2019; HUYNH et al., 2020; JIANG et al., 2019; HOSSEINIOUN et al., 2020; SHAO et al., 2019; KETYKó et al., 2016; SCHNEIDER et al., 2020; KUMAR; ZEADALLY; RODRIGUES, 2016) abordam conceitos de alocação de recursos, mas diferentemente da ideia central deste trabalho. Desse modo, não foram focados na otimização de recursos como capacidade de armazenamento, processamento, memória, largura de banda nos serviços disponibilizados na borda da rede, para otimizar o atendimento dos serviços, e ainda, alguns não consideram um ambiente com a limitação de recursos em serviços de computação de borda. Os trabalhos de (MENEGUETTE; BOUKERCHE; PIMENTA, 2019; PEREIRA et al., 2019; COSTA et al., 2020; WEERASINGHE et al., 2020) são diretamente relacionados à ideia de política de decisão com o objetivo de otimizar os serviços/tarefas atendidos e a utilização dos recursos. No entanto, possuem algoritmos que não se adaptam facilmente a mudanças de cenários para aplicação

dos algoritmos.

3.2 Considerações parciais

Pode-se observar que existem trabalhos que tratam o problema de alocação de recursos, entretanto, os algoritmos requerem modificações para serem aplicados em diferentes cenários. Neste trabalho, são implementados mecanismos de alocação de recursos que busquem otimizar o atendimento de usuários e a utilização de recursos computacionais. Tal proposta visa enfrentar o desafio de limitações dos recursos de serviços de borda, tais como recursos de processamento, armazenamento, memória, largura de banda, entre outros. Os mecanismos LOBO-IoT e BALE-IoT utilizam mecanismos meta-heurísticos bioinspirados que são considerados excelentes métodos de otimização para encontrar, gerar e selecionar um algoritmo de busca (JHA; JAIN; THENMALAR, 2018). Ademais, os mecanismos possuem a vantagem de se adaptarem facilmente a diferentes cenários de computação urbana.

Tabela 1 – Abstração do estado da arte.

Trabalhos	Método	Estrutura	Objetivo Principal	Problemas analisados
Li et al. (2019)	Algoritmo Genético	Fog	Alocação de recursos	Taxa de transferência, latência e consumo de energia
Aazam, Harras e Zeadally (2019)	Prioridade	Fog	Estimativa de recursos	Impacto na estimativa de recursos
Pereira et al. (2019)	AHP	Fog	Alocação de recursos	Atendimento de usuários
Choo, Kim e Pack (2018)	Guloso	VEC	Alocação de tarefas	Quantidade de tarefas e Tempo de atendimento
Goudos et al. (2019)	Jaya	EC	Consumo de energia	Consumo de energia em redes 5G
Huynh et al. (2020)	PSO	EC	Consumo de energia	Sobrecarga, consumo de energia e convergência
Jiang et al. (2019)	GWO	EC	Escalonamento de tarefas	Estabilidade, acurácia e convergência
Hosseinioun et al. (2020)	IWO	EC	Consumo de energia	Utilização de recursos, consumo de energia e índice de economia de energia
Shao et al. (2019)	ITO	EC	Custos de acesso aos dados	Custo de acesso, transmissão e movimentação de dados
Ketykó et al. (2016)	Mochila	MEC	Alocação de recursos	Minimizar a latência
Costa et al. (2020)	Mochila	VCC	Alocação de tarefas	Atendimento de tarefas, utilização de recursos e recompensa
Weerasinghe et al. (2020)	SMDP	Slot	Alocação	Quantidade e recompensa de alocação
Meneguette, Boukerche e Pimenta (2019)	SMDP	VCC	Alocação de recursos	Atendimento, recompensa e tentativas de alocação
Schneider et al. (2020)	Árvore de Decisão	MVs	Alocação de recursos	Limitações de capacidade e latência
Kumar, Zeadally e Rodrigues (2016)	Árvore de Decisão	MEC	Alocação de Recursos	Latência de transmissão e taxa de transferência
LOBO-IoT e BALE-IoT	GWO e BAT	EC e MEC	Alocação de Recursos	Atendimento, tentativas de alocação, recompensa e utilização

Fonte: Produzida pelo autor.

4 MECANISMOS PARA ALOCAÇÃO DE RECURSOS IOT APLICADOS EM COMPUTAÇÃO DE BORDA

Neste capítulo, são apresentados os métodos de pesquisa com base nas informações expostas no trabalho, aplicados a um cenário urbano com serviços de computação de borda. Os mecanismos LOBO-IoT e BALE-IoT foram desenvolvidos com base nos algoritmos meta-heurísticos bioinspirados GWO e WOA, respectivamente. Ambos foram elaborados para tomar as decisões no processo de alocação de recursos computacionais, considerando a massiva quantidade de dispositivos IoT, em um cenário urbano. Assim, busca-se enfrentar a limitação dos recursos computacionais dos serviços de borda, com soluções eficientes para atender a demanda dos dispositivos IoT. Além disso, os mecanismos desenvolvidos foram customizados para atender de modo eficiente os dois tipos de serviços, de borda e borda móvel, sem precisar de alterações. Portanto, neste capítulo é apresentado um cenário urbano com serviço de borda. A seguir, serão apresentados como foram aplicados os mecanismos LOBO-IoT e BALE-IoT para cada cenário configurado, implantados os ambientes de simulação e descritos os resultados obtidos.

4.1 Cenário urbano com serviços de borda

Nesta seção, é denotado um cenário urbano com serviços de borda móvel para atender solicitações de dispositivos IoT, com o objetivo de alocar seus recursos computacionais. Considera-se, desse modo, um cenário em que os serviços de borda são instalados ao redor de um perímetro urbano. Cada EC possui uma quantidade limitada de recursos computacionais, como capacidade de processamento, armazenamento, memória e largura de banda, os quais compartilham com os dispositivos IoT. Os mecanismos têm a função de alocar os serviços dos dispositivos requerentes na melhor EC.

Na Figura 7 ilustra-se a abstração do modelo de sistema para um cenário de computação de borda, em que se considera um perímetro urbano formado por uma topologia heterogênea de dispositivos IoT. Os usuários destes dispositivos se movem por toda a extensão desta topologia de modo independente. Os dispositivos se comunicam com as unidades de beira de estrada (*Roadside Units* - RSUs) pela rede 5G, quando dentro da sua faixa de cobertura. Considera-se que há R unidades de beira de estrada distribuídas pelo local e cada RSU possui um servidor capaz de processar os serviços nele mesmo. Nos serviços de borda, ficam instalados os algoritmos para tomada de decisão.

Figura 7 – Abstração do modelo de sistema com serviços de borda.

Fonte: Produzida pelo autor.

Neste contexto, quando um dispositivo IoT como, por exemplo, um *smartphone*, relógio inteligente, *tablets*, veículos com equipamentos inteligentes embarcados ou outros dispositivos estão dentro da área de cobertura e solicitam recursos para armazenar seus serviços, o serviço de borda mais próximo executa os algoritmos de tomada de decisão para auxiliar na seleção da melhor borda para receber esta requisição.

Na abstração do Algoritmo 1 descreve-se o processo executado pelos serviços de borda ao receber uma requisição. O algoritmo simula os procedimentos de entradas e saídas de dispositivos nos serviços de bordas, utilizando ou liberando, respectivamente, os recursos da ECs. Os dispositivos e os serviços de borda (ECs) possuem os seguintes recursos: *i*) processamento, *ii*) largura de banda, *iii*) memória e *iv*) armazenamento. Inicialmente, o algoritmo verifica se o dispositivo está saindo da EC *Saida_Dispositivo* (linha 1). Em caso positivo, os recursos retornam para o serviço de borda em que o dispositivo estava alocado (linha 2). Depois é verificado se a requisição é para alocação de recursos *Entrada_Dispositivo* (linha 4). Se verdadeiro, é realizada uma estrutura de repetição (linha 5) para passar por todos os dispositivos que estão entrando na rede naquele momento, escalonados por uma fila de ordem de chegada, ou seja, o primeiro que entra será o primeiro a ser atendido. A variável *bloq_dispositivo* é inicializada para contabilizar quantas vezes aquele dispositivo foi bloqueado (linha 6).

Na linha 7, é executado o mecanismo de tomada de decisão (o LOBO-IoT e o BALE-IoT), que retornam aos serviços selecionados (*provavelEC*) pelos mecanismos.

Como pode ter mais de um serviço de borda selecionado, por causa do LOBO-IoT, uma estrutura de repetição é feita para percorrer a lista *provavelEC* (linha 8). O algoritmo passa por todos os recursos do dispositivo selecionado (linha 9), para verificar se os recursos do dispositivo (*Recursos_Dispositivo_i*) não ultrapassa o limite de recursos da EC selecionada (*provavelEC_recursos_i*), na linha 10. Se ultrapassar, são atualizadas as variáveis *bloq_dispositivo* e *bloqueado*. Percorridas todas as possibilidades de ECs e de cada recurso do dispositivo, na linha 16, é verificado se o dispositivo não foi bloqueado em todas as ECs. Se verdadeiro, os recursos do dispositivo (*Recursos_Dispositivo*) são alocados na EC selecionada (*conj_recursos*(*provavelEC*)) (linha 17) e é contabilizado um dispositivo alocado (linha 18). Caso contrário, é contabilizado um dispositivo negado (linha 20).

Algoritmo 1 Abstração do algoritmo com serviço de computação de borda

```
 1: se (Saida_Dispositivo) então
 2:     conj_recursos += Recursos_Dispositivo
 3: fim se
 4: se (Entrada_Dispositivo) então
 5:     enquanto (Dispositivos) faça
 6:         bloq_dispositivo = 0
 7:         provavelEC ← Algoritmo_Decisao(Recursos_Dispositivo,Recursos_EC)
 8:         enquanto (provavelEC_Proximo) faça
 9:             enquanto (Recursos_Dispositivo) faça
10:                 se (provavelEC_recursos_i ≤ Recursos_Dispositivo_i) então
11:                     bloq_dispositivo += 1
12:                     bloqueado += 1
13:                 fim se
14:             fim enquanto
15:         fim enquanto
16:         se (bloq_dispositivo < 4) então
17:             conj_recursos(provavelEC) -= Recursos_Dispositivo
18:             alocado += 1
19:         senão
20:             negado += 1
21:         fim se
22:     fim enquanto
23: fim se
```

4.2 LOBO-IoT: Mecanismo de Otimização do Lobo Cinzento para Alocação de Recursos de Dispositivos IoT em serviços de computação de borda

Esta seção descreve o mecanismo de otimização do **LOBO** cinzento para alocação de recursos de dispositivos **IoT** em serviços de computação de borda, o LOBO-IoT, que se baseou no algoritmo meta-heurístico bioinspirado de otimização do lobo cinzento para

o cenário de alocação de recursos de dispositivos IoT. Este algoritmo foi utilizado por ter uma metodologia semelhante ao cenário considerado para este trabalho. Considera-se um conjunto de serviços de computação de borda, o que se faz necessário selecionar qual é o melhor serviço para alocar o dispositivo que está requisitando recursos das ECs. Faz-se uma analogia em que as ECs representam os lobos. Como na alcateia dos lobos cinzentos, que possui um líder, é necessário selecionar qual é a EC que representará o *alpha* para alocar os recursos do dispositivo requisitante. A seguir, será descrito como foi feito o processo de escolha da melhor EC.

O LOBO-IoT apresenta três alternativas de ECs com base no sistema de hierarquia da alcateia de lobos, em que se tem como líderes os lobos *alpha*, *beta* e *delta*. A melhor solução encontrada pelo mecanismo é considerada como *alpha* (α). Seguindo a ordem da pirâmide, a segunda e a terceira soluções são definidas como *beta* (β) e *delta* (δ), respectivamente. As demais ECs são definidas como *ômegas*. Para a implementação do algoritmo LOBO-IoT, a caça é guiada pelas soluções α, β e δ. Considera-se, na aplicação deste cenário, que os lobos são as ECs e o objetivo é identificar qual a melhor EC para alocar os recursos computacionais do dispositivo requerente.

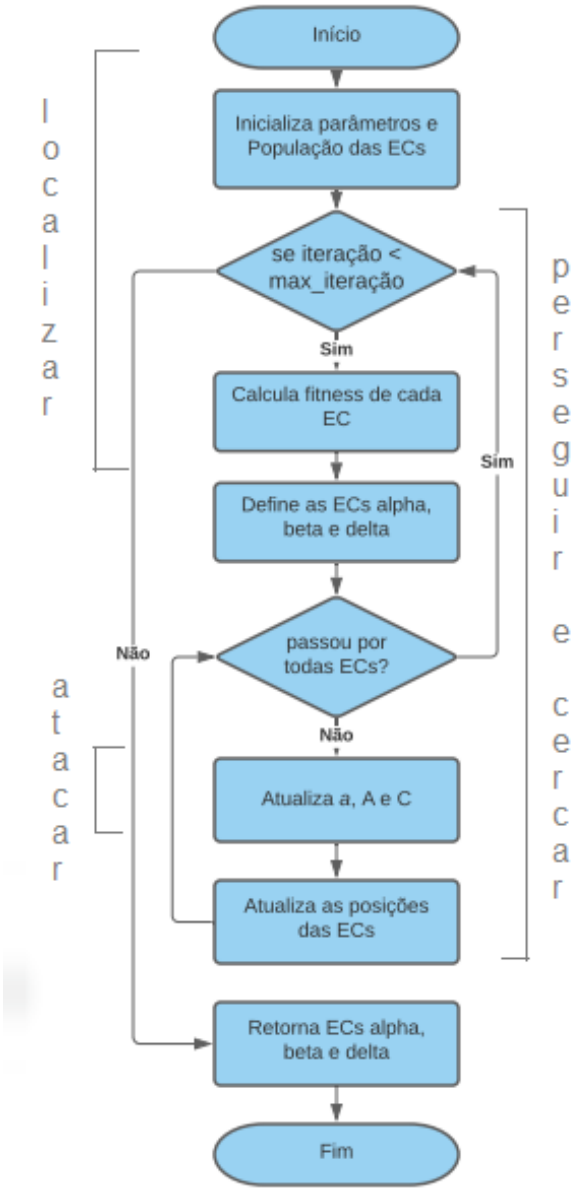
O fluxograma do modelo matemático para otimização do processo de alocação de recursos, baseado no cenário desta seção, é apresentado na Figura 8. Ele é embasado nas seguintes etapas utilizadas pela alcateia dos lobos cinzentos (MIRJALILI; MIRJALILI; LEWIS, 2014):

1. **localizar e perseguir o dispositivo:** abrange as etapas de inicialização das variáveis, preenchimento da população de ECs, baseado na diferença do valor do recurso do dispositivo, e na definição inicial da aptidão;
2. **perseguir e cercar o dispositivo:** consiste nos processos de iterações para atualização dos ECs *alpha*, *beta* e *delta*, e nas atualizações das posições das ECs, com base em suas aptidões, definidas em cada iteração;
3. **atacar o dispositivo:** é a etapa que a variável a é decrementada a cada iteração, aproximando a EC do valor ideal em relação ao dispositivo.

Na primeira etapa, inicializam-se os parâmetros de número de iterações (n_iter), dimensão dos ECs (dim) e a população de ECs (p), em que é considerada a diferença de valores entre os recursos das ECs e o dispositivo requerente. O mecanismo aponta quais são as três melhores ECs para liderar o conjunto durante a caça pelo melhor serviço de borda para atendê-lo. Declara-se a melhor EC como *alpha* (α), a segunda melhor como *beta* (β) e a terceira como *delta* (δ). As três ECs selecionadas são responsáveis por guiar a caça ao dispositivo, conforme definido pela Equação 4.1.

$$f(x+1) = \frac{f_1 + f_2 + f_3}{3} \tag{4.1}$$

Figura 8 – Fluxograma dos processos de caça adaptado ao LOBO-IoT - Método com EC.



Fonte: Produzido pelo autor.

Em que f corresponde à aptidão da ECs, identificadas e atualizadas durante as iterações. Assim, para iniciar a perseguição e circundação, Mirjalili, Mirjalili e Lewis (2014) propuseram as equações 4.13 e 4.14, as quais foram adaptadas para modelar o comportamento das ECs durante a perseguição ao dispositivo.

$$EC = c2 * V_d(x) - V(x) \tag{4.2}$$

$$V(r+1) = V_d(x) - c1 * EC \tag{4.3}$$

No qual x representa a iteração atual, V_d é a variável correspondente ao recurso do dispositivo, V indica os recursos que o serviço de borda EC possui, e $c1$ e $c2$ são os

coeficientes utilizados para movimentação na fórmula. Desta forma, é feita a relação de distância entre os recursos do dispositivo e o serviço de EC, que diminui a cada iteração.

As variáveis $c1$ e $c2$ são calculadas seguindo os modelos das equações 4.15 e 4.16, respectivamente:

$$c1 = 2a * s_1 - a \tag{4.4}$$

$$c2 = 2 * s_2 \tag{4.5}$$

Em que a é decrementado de 2 até 0 durante a iteração e s_1 e s_2 são variáveis aleatórias com valores entre 0 e 1. Estes valores são atribuídos para que outras ECs alcancem diferentes posições ao redor da melhor EC no momento da circundação, como a movimentação apresentada anteriormente na Figura 4. A variável a é responsável por fazer a EC se aproximar do dispositivo em cada iteração.

A próxima etapa é atacar o dispositivo. Considera-se que os três serviços de borda definidos como líderes possuem conhecimento da diferença dos recursos do dispositivo. Para isso, são selecionadas as três melhores ECs de acordo com sua aptidão para atualizar a distância dos recursos das ECs em relação aos recursos dos dispositivos. Mirjalili, Mirjalili e Lewis (2014) propuseram as equações 4.6, 4.7 e 4.8 para representar matematicamente o comportamento da caça.

$$f_1 = f_{-\alpha}(t) - c1_1 * EC\_\alpha \tag{4.6}$$

$$f_2 = f_{-\beta}(t) - c1_2 * EC\_\beta \tag{4.7}$$

$$f_3 = f_{-\delta}(t) - c1_3 * EC\_\delta \tag{4.8}$$

Em que se calcula as aptidões f dos três líderes da alcateia. As variáveis EC_α, EC_β e EC_δ são definidas, respectivamente, pelas equações 4.9, 4.10 e 4.11:

$$EC\_\alpha = c2 * f_{-\alpha}(x) - f(x) \tag{4.9}$$

$$EC_{-\beta} = coef2 * f_{-\beta}(x) - f(x) \tag{4.10}$$

$$EC_{-\delta} = c2 * f_{-\delta}(x) - f(x) \tag{4.11}$$

O coeficiente a é responsável pelo ataque ao dispositivo, ou seja, ele é decrementado a cada iteração do processo de perseguição até o momento que chega a distância mais próxima (na última iteração). Matematicamente, esta etapa é feita com a atualização de a, como apresentado na Equação 4.12:

$$a = 2 - (\frac{a}{n\_iter}) \tag{4.12}$$

Em que a é decrementado de 2 a 0 e n_iter corresponde ao número máximo de iterações definido no algoritmo. Com a atualização de a a cada iteração, também ocorre a flutuação de $c1$. Isto faz com que as ECs se aproximem da melhor posição dos recursos do dispositivo, pois a cada iteração decrementada a próxima posição tende a ser entre a posição atual da EC e a posição do dispositivo.

O Algoritmo 2 é responsável pela seleção das ECs (*Algoritmo_Decisao*) para alocação de recursos no Algoritmo 1. Como parâmetros, o algoritmo tem os dados de recursos do dispositivo (*Recursos_Dispositivo*) e dos serviços de borda (*Recursos_EC*). Na linha 1 são inicializados os parâmetros de número de iterações (n_iter) e dimensão dos ECs (dim). Na linha 2 verifica-se e armazena-se quantos dispositivos solicitaram recursos (num_disp). A população de agentes, correspondente às ECs, é populada com a distância euclidiana do recursos do dispositivo em relação ao recurso da EC (linha 5). Inicia-se, então, o máximo de iterações definidas para o processo de busca pela melhor solução (linha 8). Na linha 9 há uma estrutura de repetição para percorrer todas as ECs, para que sejam calculadas as aptidões (linha 10). Posteriormente, é verificado se a aptidão da EC atual está entre as três melhores, para ser definida como *alpha*, *beta* ou *delta* (linhas 12, 14 e 16, respectivamente).

Após isso, o algoritmo passa para a etapa do ataque ao dispositivo. Primeiro é atualizada a variável responsável por fazer o avanço no ataque a (linha 19). Depois, na estrutura de repetição, são atualizados os coeficientes $c1$ e $c2$ (linha 22), conforme equações 4.15 e 4.16, respectivamente, além de executar as equações que definem as posições de movimentação das ECs (linhas 23, 24 e 25). Por fim, após executar o máximo de iterações (n_iter), definido no algoritmo, é retornado os três melhores ECs: EC_Alpha, EC_Beta e EC_Delta (linha 29).

4.3 BALE-IoT: Mecanismo Baseado no Algoritmo de Otimização da Baleia para Alocação de Recursos IoT em serviços de computação de borda

Nesta seção, descreve-se o mecanismo baseado no algoritmo de otimização da **BALE**ia para alocação de recursos **IoT** em computação de borda, chamado BALE-IoT. O

Algoritmo 2 LOBO-IoT

```
1: n_iter, dim
2: num_disp ← len(D)
3: enquanto D faça
4:    enquanto dim faça
5:       p(x, y) ← euclidiana(D_x, recursosEC)
6:    fim enquanto
7: fim enquanto
8: enquanto (n_iter) faça
9:    enquanto (Numero_EC) faça
10:      fitness ← calculaFitness(Numero_EC, :)
11:      se fitness < EC_Alpha então
12:         atualiza_Alpha()
13:      senão se fitness < EC_Beta então
14:         atualiza_Beta()
15:      senão se fitness < EC_Delta então
16:         atualiza_Delta()
17:      fim se
18:   fim enquanto
19:   atualiza(a)
20:   enquanto (provEC_Proximo ≤ Numero_EC) faça
21:      enquanto (tamanho) faça
22:         atualiza as variáveis (c1 e c2)
23:         executa as equações (6.8, 6.9 e 6.10)
24:         executa as equações (6.5, 6.6 e 6.7)
25:         executa as equações (6.11)
26:      fim enquanto
27:   fim enquanto
28: fim enquanto
29: retorna EC_Alpha, EC_Beta, EC_Delta
```

WOA foi escolhido para modelar este mecanismo por ser considerado um otimizador global devido ao seu modelo de exploração e processo de evolução e ser facilmente customizável ao cenário proposto. Ademais, alguns processos heurísticos e parâmetros do algoritmo foram minimizados para tornar os processos mais simples (MIRJALILI; LEWIS, 2016). Faz-se uma analogia em que os ECs são as presas da baleia que, por sua vez, é representada pelo dispositivo IoT que solicita armazenar recursos nos serviços de borda. Aplica-se, então, o comportamento de caça do dispositivo para encontrar o melhor serviço de borda para atendê-lo. A seguir, será detalhado todo o processo do mecanismo.

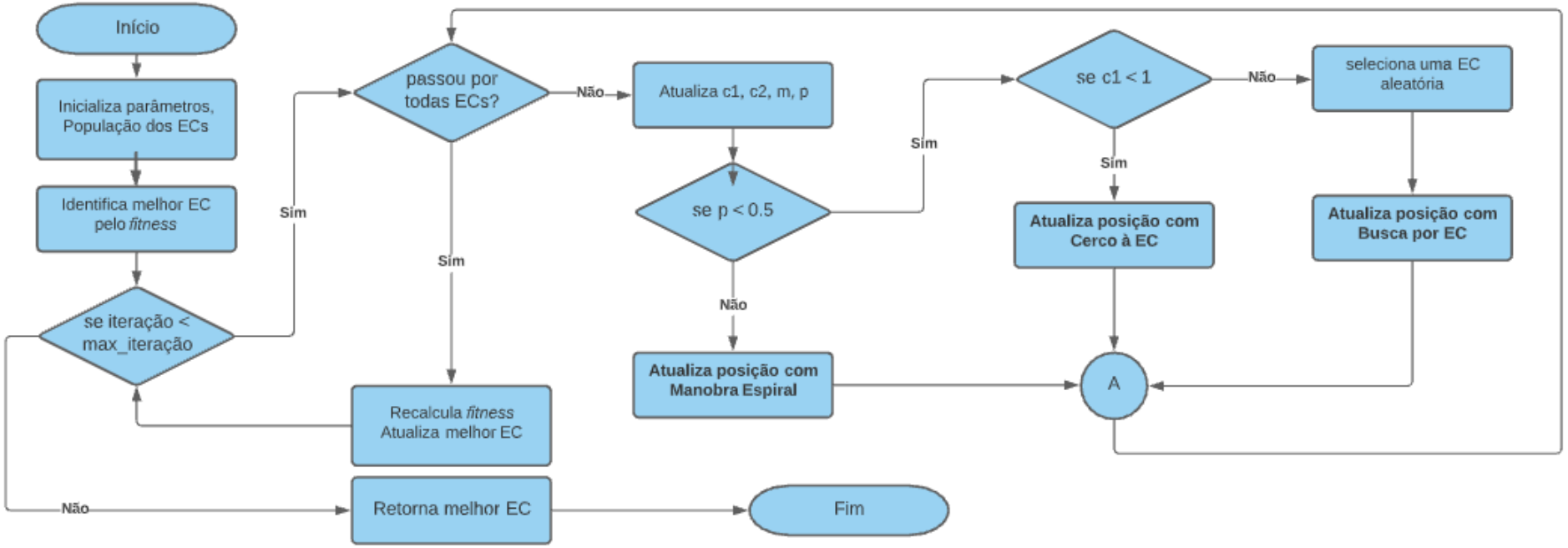
Três manobras principais são realizadas pelo mecanismo do BALE-IoT, das quais são:

- **Presa em círculo:** consiste no movimento de circundação do dispositivo em relação ao EC, no qual se busca a aproximação do melhor resultado;
- **Manobra espiral de alimentação com rede de bolhas:** é o processo realizado pelo mecanismo para cercar e avançar em direção à melhor EC;

- **Busca por presas:** o dispositivo se movimenta baseado em uma EC aleatória, com o objetivo de encontrar uma EC melhor para se aproximar.

No fluxograma, apresentado na Figura 9, denota-se o modelo matemático criado por Mirjalili e Lewis (2016) e utilizado como base para a aplicação das manobras citadas anteriormente. O processo começa com a inicialização dos parâmetros de número de iterações (n_iter), dimensão dos ECs (dim) e a população de ECs (pos), em que é considerada a diferença de valores entre os recursos das ECs e o dispositivo requerente. Em seguida, percorre-se a estrutura de repetição com as iterações definidas e, em cada iteração, analisa todas as ECs. Nesta etapa, o mecanismo tem duas fases: i) mecanismo de encolhimento do círculo e ii) a atualização da espiral. Presume-se que a probabilidade de escolher entre uma fase ou outra é de 50%. Caso seja direcionado para o mecanismo de encolhimento, têm-se mais uma decisão aleatória, para decidir se a atualização da posição será de forma aleatória ou baseada na melhor EC até o momento. Finalizadas as iterações, o mecanismo retorna à melhor EC identificada no final dos procedimentos.

Figura 9 – Fluxograma dos processos de caça adaptado ao BALE-IoT - Método com EC.



Fonte: Produzido pelo autor.

A primeira manobra é de localização e circundação da EC. O mecanismo BALE-IoT inicializa a população dos indivíduos com os dados de recursos computacionais das ECs. Feito isso, é calculada a aptidão das ECs a fim de detectar a melhor solução candidata atual. Então, é utilizada a informação da atual melhor EC para atualizar as posições das outras em relação a ela. Esta etapa é realizada de acordo com as Equações 4.13 e 4.14.

$$EC = c2 * ED_d(x) - ED(x) \tag{4.13}$$

$$ED(x+1) = ED_d(x) - c1 * EC \tag{4.14}$$

Em que x é a iteração atual. A variável ED_d representa o atual serviço de borda com melhor aptidão e pode ser atualizada durante as iterações. A variável EC corresponde ao serviço de borda que está se reposicionando na atual iteração. As variáveis $c1$ e $c2$ são os coeficientes que possibilitam alcançar posições diferentes ao redor do melhor agente e são calculados pelas Equações 4.15 e 4.16.

$$c1 = 2a * r - a \tag{4.15}$$

$$c2 = 2 * r \tag{4.16}$$

Em que a é decrementado de 2 a 0, no decorrer das iterações para realizar o cerco, e r é um valor aleatório entre 0 e 1, que permite chegar a qualquer posição na vizinhança da melhor solução.

A segunda manobra é considerada a fase de exploração, em que se aplica o conceito de redes de bolhas. Duas formas são aplicadas:

- **Mecanismo de encolhimento de redução:** em que o coeficiente m segue a declínio da variável a a cada iteração, para se aproximar da EC;
- **Posição de atualização da espiral:** aplica-se a Equação 4.17 para atualizar as posições com o movimento espiral, baseado nos recursos dos dispositivos.

$$ED(x+1) = EC * e^{bl} * \cos(2\pi l) + ED(x) \tag{4.17}$$

Do qual, EC é o melhor agente até o momento, b é um coeficiente para determinar a espiral logarítmica e l é um valor randômico de -1 a 1 para movimento pela vizinhança.

Para se aproximar das ECs, em cada iteração pode ser aplicado qualquer um dos movimentos da rede de bolhas, baseado na premissa de 50% de probabilidade. Define-se as Equações 4.14 e 4.17 para efetuar os movimentos de encolhimento de redução e atualização da espiral, respectivamente.

Na terceira manobra do mecanismo é feita a busca randômica pela melhor EC. Isto ocorre com a movimentação da atualização de posições sendo realizada com base em uma EC escolhida de modo aleatório. As Equações 4.18 e 4.19 representam a terceira manobra.

$$EC = c2 * ED_{rand} - ED \tag{4.18}$$

$$ED(x+1) = ED_{rand} - c1 * EC \tag{4.19}$$

A abstração do Algoritmo 3 será apresentada a seguir. O mecanismo desenvolvido para seleção da melhor EC, baseado no algoritmo WOA, e adaptado para o cenário de alocação de recursos de dispositivos IoT em serviços de computação de borda. Como parâmetros são enviados os recursos dos dispositivos (D) e os recursos das ECs ($recursosEC$). A princípio, são inicializadas as variáveis n_iter e dim, que correspondem ao número de iterações e dimensão dos recursos das ECs, respectivamente (linha 1). A população de agentes é preenchida com a distância euclidiana dos valores de recursos dos dispositivos (D) em relação EC ($recursosEC$) (linha 4). Na linha 7 é calculada a aptidão da população de agentes (ECs) e na linha 8 é verificada qual a melhor aptidão, neste momento do algoritmo. Aplica-se um laço de repetição com o número de iterações definido para o algoritmo (linha 9). A cada iteração, percorre-se todas as ECs para atualizar suas posições e se aproximar da melhor solução em cada iteração (linha 10). Os parâmetros responsáveis pelas movimentações e decisão são atualizados na linha 11.

A variável p representa a probabilidade de aplicação dos mecanismos de encurtamento ou atualização da espiral. Quando p é menor que 0.5 (linha 12), é aplicado o mecanismo responsável pelo encolhimento. Neste mecanismo, dois processos podem ser realizados: i) a atualização da posição é efetuada baseada no posicionamento da melhor EC atual, se o coeficiente $c1$ for menor que 1 (linha 14). Senão, ii) a atualização é baseada em uma EC escolhida de modo aleatório (linhas 16 e 17). Caso p seja maior ou igual a 0.5, é aplicado o mecanismo de atualização com manobra espiral (linha 20). Após percorrer todas as ECs, no final de cada iteração, são atualizadas as aptidões dos agentes da população, a melhor EC até o momento e a iteração (linhas 23, 24 e 25). Finalizado o processo de iteração, é retornado o melhor serviço de borda (ED) identificado pelo mecanismo (linha 27).

Algoritmo 3 BALE-IoT

```
1: n_iter, dim
2: enquanto D faça
3:    enquanto dim faça
4:       pos(x, y) ← euclidiana(D_x, recursosEC)
5:    fim enquanto
6: fim enquanto
7: calcula a aptidão
8: identifica a melhor EC
9: enquanto (t < n_iter) faça
10:   for cada_EC faça
11:      atualiza a, c1, c2, l, p
12:      se (p < 0.5) então
13:         se (c1 < 1) então
14:            atualiza posição com Eq. 1
15:         senão se (c1 ≥ 1) então
16:            seleciona uma EC aleatória
17:            atualiza posição com Eq. 6
18:         fim se
19:      senão se (p ≥ 0.5) então
20:         atualiza posição com Eq. 5
21:      fim se
22:   fim for
23:   recalcula a aptidão
24:   atualiza ED se tiver algum agente melhor
25:   t = t + 1
26: fim enquanto
27: retorna ED
```

4.4 Simulação

Esta seção apresenta a aplicação dos mecanismos LOBO-IoT e BALE-IoT, no cenário urbano com ECs, e define como os experimentos foram conduzidos. Explica-se a configuração do ambiente, as métricas de avaliação e as técnicas de comparação utilizadas para o cenário urbano com serviços de computação de borda.

4.4.1 Configuração do cenário

Considera-se o ambiente de um perímetro urbano com 4 RSUs interconectadas por rede 5G. Para representar este ambiente, é considerada a cidade de Manhattan, que possui um perímetro em que as 4 RSUs podem cobrir e interconectar os serviços de ECs (PEREIRA et al., 2019). Cada RSU possui um equipamento de serviço de borda, com capacidade para processar e compartilhar recursos computacionais. Os recursos compartilhados por estas ECs são: *i*) consumo de processamento, *ii*) consumo de largura de banda considerando rede 5G, *iii*) consumo de memória e *iv*) tempo de utilização do serviço.

Cada dispositivo possui uma identificação individual $D\_id \in [1, n]$ e um conjunto de recursos computacionais formado por $R = (proc_r, band_r, mem_r, tem_r)$, correspondentes à capacidade de processamento, largura de banda, memória e tempo de processamento, respectivamente. A quantidade de cada recurso é definida de forma randômica com valores de [1,10], gerados sinteticamente durante a simulação, que representam a porcentagem de cada recurso que um dispositivo necessita. As ECs são inicializadas com o valor 100 para cada recurso computacional que compartilha ($EC = [100, 100, 100, 100]$), correspondente a 100% da sua capacidade. O trabalho de Pereira et al. (2020) foi utilizado para validar a implementação deste cenário.

O tempo de simulação foi variado (1200, 2400, 3600, 4800 e 6000 segundos), para gerar uma variação dos números de dispositivos (198, 423, 618, 827 e 1064) que solicitam recursos das ECs. Isso possibilita comparar situações diferentes, simulando casos com poucos e muitos dispositivos. Para realizar a mobilidade da entrada e saída de dispositivo das ECs, utilizou-se a distribuição de Pearson III, que é um padrão gama avançado que pode representar o padrão heterogêneo do tráfico de entrada e saída de dispositivos (BOBEE; ROBITAILLE, 1977). Cada simulação foi executada 33 vezes. Apresenta-se a média dos resultados e um intervalo de confiança de 95% de acordo com a distribuição *T-student*.

Os mecanismos foram desenvolvidos utilizando a linguagem de programação Python (versão 3.7)[1]. Para validar a efetividade da simulação extensiva e utilização de dados sintéticos, foram levantados trabalhos na literatura, tais como o de Pereira et al. (2020) e Nabi et al. (2017). Na Tabela 2 apresenta-se os parâmetros da configuração da simulação deste cenário de modo resumido.

Tabela 2 – Configuração da simulação com serviços de borda.

Parâmetros	**Valores**
Cenário urbano	Cidade de Manhattan
Número de Dispositivos	[198, 423, 618, 827, 1064]
Tempo de Simulação	[1200, 2400, 3600, 4800 e 6000] segundos
Número de RSUs	4 RSUs com ECs que se comunicam por rede 5G
Valores dos recursos	sintético e randômico de [1,10]
Entrada e saída de dispositivos na EC	Distribuição de Pearson III
Número de execuções	33
Intervalo de confiança	95% (distribuição *t-student*)
Linguagem de programação	Python (versão 3.7)

Fonte: Produzida pelo autor.

4.4.2 Métricas de avaliação

A simulação é avaliada considerando as seguintes métricas:

[1] https://www.python.org/

- **Dispositivos atendidos:** é uma métrica que representa o número de dispositivos atendidos durante a simulação;
- **Dispositivos bloqueados:** é uma métrica que contabiliza a quantidade de vezes que uma EC bloqueou a alocação de serviços de um dispositivo. É contabilizado sempre que a EC verificada não tem recurso suficiente para alocar um dispositivo;
- **Dispositivos negados:** é uma métrica que computa quando um dispositivo foi bloqueado em todos os serviços de borda, ou seja, o serviço foi bloqueado na tentativa de alocação em todas as ECs.

4.4.3 Técnicas de comparação

Para validação da eficiência dos mecanismos propostos, os mecanismos LOBO-IoT e BALE-IoT foram comparados com outras técnicas heurísticas tradicionalmente utilizadas por pesquisadores para alocação de recursos, tais como nos trabalhos de Choo, Kim e Pack (2018), (FICKERT, 2020) e (PEREIRA et al., 2020), que utilizaram os métodos Guloso, Best-first e Reliable, respectivamente. Neste trabalho, os conceitos dos métodos foram implementados como descrito a seguir:

- **Guloso:** o método tenta alocar os recursos do dispositivo requerente sempre na primeira EC que encontrar;
- **Best-first:** o mecanismo foi implementado para encontrar a melhor EC baseado na disponibilidade de recursos de largura de banda. Assim, o algoritmo percorre todas as ECs e seleciona qual tiver com o recurso de largura de banda mais próximo dos 100% de utilização;
- **Reliable:** o método é baseado no método AHP. É aplicado um fator de influência para os recursos dos dispositivos, definido graus de importâncias e aplicada uma matriz de decisão multicritério para tomada de decisão.

4.5 Resultados

Após realizar as etapas de simulações, os dados resultantes foram analisados e processados para fins de validação, desempenho e eficiência dos mecanismos LOBO-IoT e BALE-IoT. Para validação das simulações e dos resultados, as comparações foram realizadas com cenários e métodos de alocação de recursos implementados e utilizados por pesquisadores (CHOO; KIM; PACK, 2018; FICKERT, 2020; PEREIRA et al., 2020). Sendo assim, nesta seção, são apresentados os resultados dos testes de aptidão realizados para selecionar a melhor função para os algoritmos meta-heurísticos bioinspirados, propostos neste trabalho, e a comparação dos resultados de alocação de recursos no cenários com serviços de borda.

Os parâmetros utilizados para os mecanismos LOBO-IoT e BALE-IoT são: *i*) o número de iterações *n_iter* foi definido como 30, por ser o número de iterações que os mecanismos se aproximaram do valor de aptidão ideal, e *ii*) a dimensão das matrizes de populações *dim* recebe o valor 4, que corresponde à quantidade de ECs considerada no cenário. A seguir, são apresentadas comparações das funções de aptidão, convergência dos mecanismos LOBO-IoT e BALE-IoT, e realizadas as comparações com as métricas de alocação.

4.5.1 Comparação das funções de aptidão

Esta subseção apresenta o resultado dos testes de aptidão no cenário urbano de computação de borda. Os mecanismos LOBO-IoT e BALE-IoT foram simulados com as funções de aptidão Ackley, Rastrigin e Rosenbrock. Para definir a melhor função para a comparação com as métricas de alocação, optou-se por comparar os resultados dos dispositivos, atendidos por ser a métrica, que representa a maximização de atendimentos dos dispositivos. Na Tabela 3, é apresentada a média dos resultados obtidos nas simulações, considerando a quantidade média de dispositivos [198, 423, 618, 827, 1064] com as funções de aptidão Ackley, Rastrigin e Rosenbrock.

Tabela 3 – Média de dispositivos atendidos por cada função de aptidão - Configuração com ECs.

	LOBO-IoT			**BALE-IoT**		
Dispositivos/Função	**Ackley**	**Rastrigin**	**Rosenbrock**	**Ackley**	**Rastrigin**	**Rosenbrock**
198	163.70	162.00	160.48	161.88	159.73	158.54
423	279.42	280.45	279.24	274.30	276.85	273.36
618	385.57	393.94	390.61	381.64	387.88	382.24
827	500.61	507.64	505.42	492.75	498.21	496.42
1064	618.33	621.94	621.88	608.00	610.94	610.18

Fonte: Produzida pelo autor

Nesta simulação, é possível observar que na configuração com 198 dispositivos, a função Ackley obteve melhor resultado em comparação às Funções Rastrigin e Rosenbrock, nos dois mecanismos. Com 423 dispositivos, a função Rastrigin teve uma leve vantagem em relação às funções Ackley e Rosenbrock, atendendo em média **208.45** e **276.85** dispositivos, com os mecanismos LOBO-IoT e BALE-IoT. Na simulação com 618 dispositivos, Rastrigin alocou em média **507.64** dispositivos com o mecanismo LOBO-IoT, enquanto que Rosenbrock e Ackley alocaram 505.42 e 500.61, respectivamente. Enquanto no mecanismo BALE-IoT, Rastrigin alocou **387.88** dispositivos, contra 382.24 e 381.64, de Ackley e Rosenbrock, respectivamente.

Com 827 dispositivos, Rastrigin também apresentou melhor performance nos mecanismos LOBO-IoT e BALE-IoT atendendo, respectivamente, **507.64** e **498.21** dispositivos. Enquanto Rosenbrock atendeu 505.42 e 496.42, e Ackley atendeu 500.61 e 492.75. Rastrigin também teve melhor desempenho na simulação com maior quantidade

de dispositivos, atendendo **621.94** e **610.94** para os algoritmos LOBO-IoTe BALE-IoT, respectivamente. Seguido de perto por Rosenbrock, com 621.88 e 610.18, enquanto Ackley atendeu 618.33 e 608 dispositivos. Os resultados demonstram que a função Rastrigin apresentou melhor desempenho e será considerada para a comparação dos resultados em relação às métricas do cenário urbano com serviços de borda, para ambos os mecanismos.

4.5.2 Convergência dos mecanismos

Nesta seção, são apresentadas as melhores convergências obtidas para o cenário de computação de borda, com base na melhor função de aptidão selecionada para o cenário apresentado.

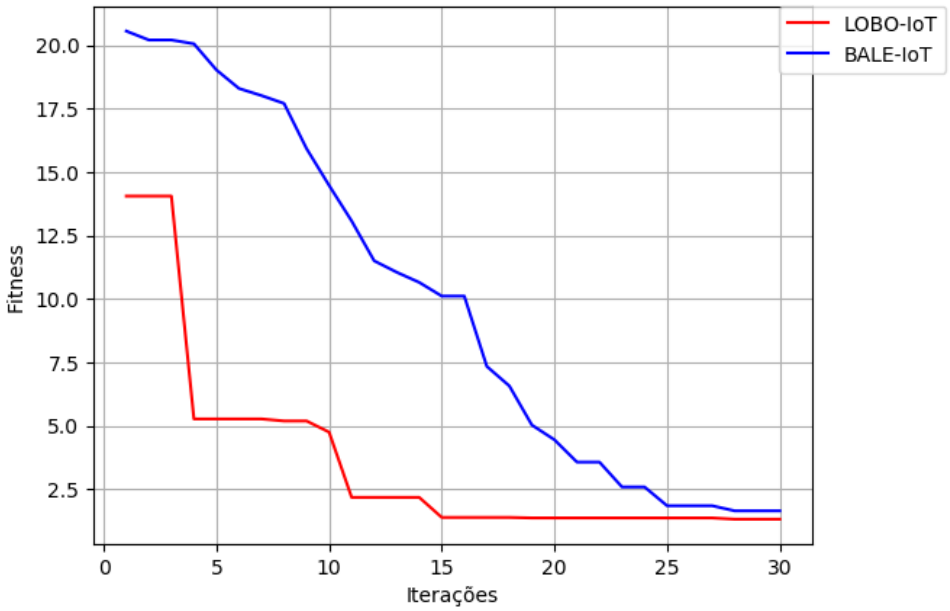
Na Figura 10, é possível visualizar a convergência dos mecanismos LOBO-IoT e BALE-IoT para a simulação com serviços de borda, na qual utilizaram a função de Rastrigin. O LOBO-IoT apresenta uma curva de convergência mais rápida. Inicia-se com a função de aptidão no valor de 14.05. Na terceira iteração, alcança o valor de 5.27. Na décima primeira iteração, o valor de aptidão cai para 2.18 e, na décima quinta, novamente tem um avanço, em que o valor chega a 1.39. Mantém-se com pouca variação até a iteração final, em que terminou com o valor de **1.37**.

O BALE-IoT demora mais para convergir ao valor considerado ideal, em relação ao LOBO-IoT. O mecanismo inicia a iteração com o valor de aptidão 20.55. O valor de aptidão vai decaindo sem grandes quedas. Na décima iteração atinge o valor de 14.47. Na metade das iterações, o BALE-IoT possui 10.11 de valor de aptidão. Com 7.11, na décima sétima iteração, começa sua maior queda para se aproximar da sua aptidão final. Somente na vigésima quinta iteração, o mecanismo chega a 1.85 e mantém a curva próxima do valor final, que foi de **1.65** na trigésima iteração. Pode-se observar que o LOBO-IoT converge rapidamente e, na décima quinta iteração, chegou próximo ao valor ideal. Enquanto o BALE-IoT só se aproximou do valor de aptidão final na vigésima quinta iteração. Apesar disso, o BALE-IoT conseguiu se aproximar, no final das iterações, da aptidão do LOBO-IoT.

4.5.3 Comparação entre as técnicas de alocação

Os resultados obtidos durante as simulações realizadas para o cenário com serviços de borda foram comparados com técnicas de alocação da literatura. Os mecanismos LOBO-IoT e BALE-IoT utilizam a função Rastrigin para cálculo de aptidão das populações. Foram comparadas as métricas de dispositivos atendidos, bloqueados e negados. Os números de dispositivos variam [198, 423, 618, 827 e 1064], de acordo com a distribuição de Pearson III, para representar desde o cenário menos desafiador (menor número de dispositivos), até chegar a um mais desafiador (maior número de dispositivos).

Figura 10 – Gráfico de convergência - Configuração com ECs.

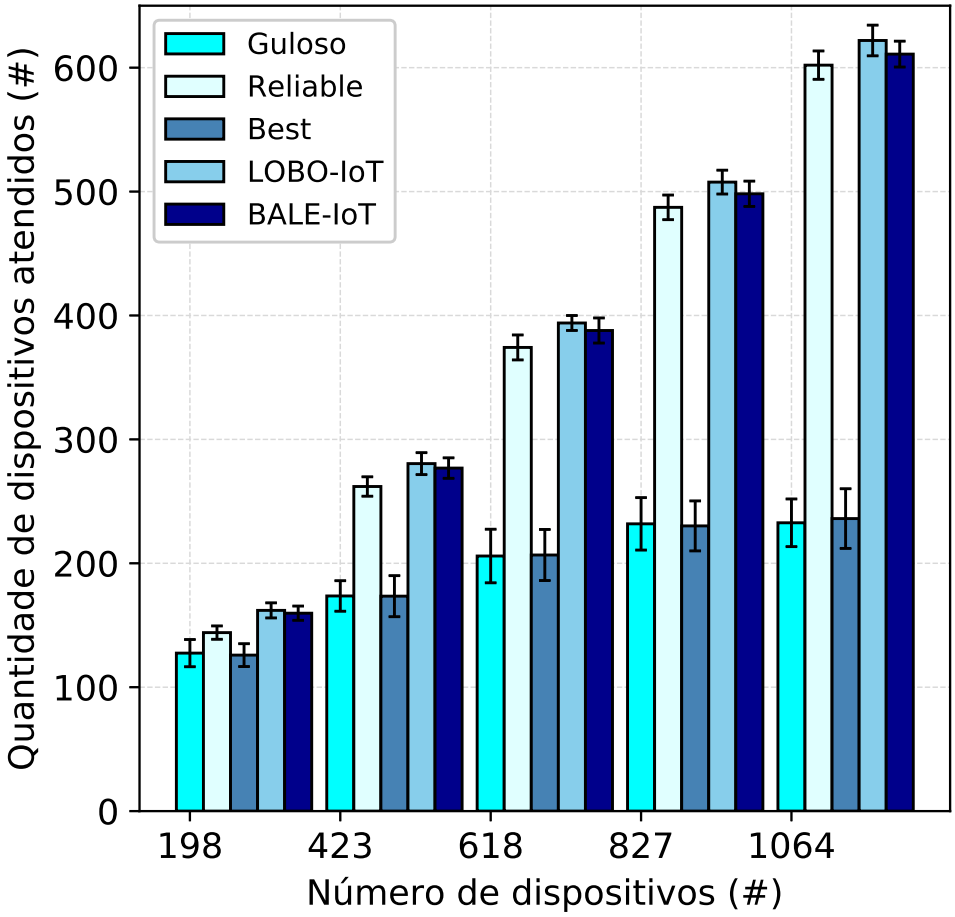


Fonte: Produzido pelo autor.

Os dispositivos atendidos correspondem àqueles que solicitaram recursos das ECs e, mesmo que tenham sido bloqueados em alguma tentativa, no final conseguiram ser atendidos por alguma das ECs. Na Figura 11 denota-se a média de dispositivos atendidos, comparando os mecanismos LOBO-IoT e BALE-IoT com outros utilizados na literatura, que são o Guloso, Best e Reliable. Na simulação com 198 dispositivos, o LOBO-IoT atendeu em média mais dispositivos, sendo **162.00** solicitações atendidas, seguido de perto pelo BALE-IoT que atendeu **159.73**. Os métodos tradicionais Reliable, Guloso e Best atenderam, respectivamente, 144.06, 127.51 e 125.91. Quando 423 dispositivos solicitaram recursos dos serviços de bordas, o mecanismo LOBO-IoT obteve melhor aproveitamento também, atendendo **280.45**. O BALE-IoT atendeu **276.85**, ainda próximo da performance do LOBO-IoT. Reliable, Guloso e Best mantiveram a sequência, atendendo em média 261.97, 173.64 e 173.48 dispositivos, respectivamente.

Na terceira simulação, com 618 dispositivos, o LOBO-IoT e o BALE-IoT novamente foram mais eficientes. Atenderam em média **393.94** e **387.88** dispositivos, respectivamente. Enquanto o Reliable atendeu 374.21 e os métodos Best e Guloso foram menos eficientes, atendendo 206.70 e 205.88, respectivamente. O LOBO-IoT também foi mais eficiente na simulação com 827 dispositivos, atendendo **521.94**. Seguido do BALE-IoT que atendeu **498.21** dispositivos. O Reliable atendeu 487.27 dispositivos. Enquanto o Guloso e o Best atenderam 231.85 e 230.18, respectivamente. Na simulação com mais dispositivo, novamente o LOBO-IoT foi mais eficiente, atendendo em média **621.94** dispositivos, enquanto que o BALE-IoT, segundo melhor, atendeu **610.94**. O Reliable, o Best e o Guloso completam a sequência, atendendo 602.30, 236.09 e 232.70, respectivamente.

Figura 11 – Gráfico de dispositivos atendidos - Configuração com ECs.



Fonte: Produzido pelo autor.

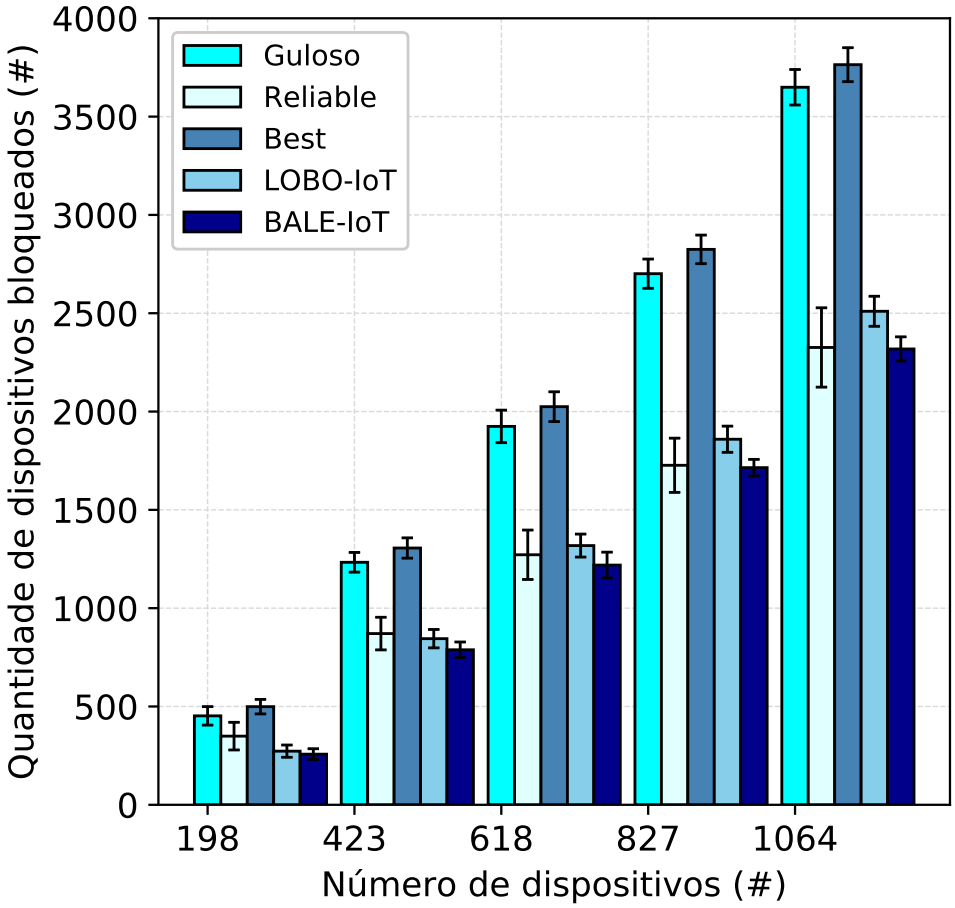
Com os resultados apresentados, os mecanismos LOBO-IoT e BALE-IoT foram mais eficientes no atendimento de dispositivos do que os métodos da literatura, Reliable, Guloso e Best. O LOBO-IoT demostrou melhor performance por apresentar três alternativas de ECs para atendimento em cada solicitação de tomada de decisão do mecanismo. O BALE-IoT também demonstrou ser eficiente, superando os outros métodos da literatura. Isso se deu por ter o mecanismo de inteligência da baleia para se aproximar da melhor solução. Além disso, ambos utilizam a aptidão como modo de seleção das ECs e em cada iteração, os mecanismos se aproximam da melhor solução baseando-se na diferença dos recursos das ECs em relação ao dispositivo, conforme foi determinado no momento de gerar a população.

Dispositivos bloqueados correspondem à quantidade de tentativas que um recurso tenta ser alocado nas ECs, mas não pode ser atendido. Na Figura 12 apresenta-se a média de dispositivos bloqueados, comparando os mecanismos propostos com outros utilizados na literatura. Na simulação com menor número de dispositivos, o BALE-IoT teve apenas **257.51** bloqueios durante a simulação, seguido do LOBO-IoT que teve **272.82** tentativas de alocar seus recursos. Em terceiro, o Reliable teve 349.18, seguido dos mecanismos

Guloso e Best que tiveram 452.33 e 499.12 bloqueios, respectivamente. Na simulação com 423 dispositivos, os recursos do BALE-IoT e do LOBO-IoT foram bloqueados em média **788.39** e **844.94** vezes, respectivamente, seguidos do Reliable, com 870.82 bloqueios.

Os métodos Guloso e Best já começaram apresentar maior dificuldade na busca por alocar seus recursos, sendo bloqueados 1233.33 e 1305.97 vezes, respectivamente. Na terceira simulação, o BALE-IoT novamente teve menos recursos bloqueados, **1219.15**. O Reliable foi o segundo menos bloqueado, com a média de 1271.64 bloqueios, seguido do LOBO-IoT que teve seus recursos bloqueados 1318.30 vezes. O Guloso teve 1924.48 bloqueios, enquanto que o Best teve 2024.79. Nas simulações com mais dispositivos, o BALE-IoT também foi o mais efetivo, com a média de **1714.70** e **2318.30** bloqueios nas simulações de 827 e 1064 dispositivos, respectivamente. Seguido de perto pelo Reliable que teve 1726.67 e 2325.91 bloqueios. O LOBO-IoT apresentou a terceira melhor média nas duas últimas simulações, com 1859.18 e 2509.91 bloqueios de recursos. Enquanto o Guloso foi bloqueado em média 2701.09 e 3649.27 vezes, e Best apresentou 2824.76 e 3764.09 bloqueios nas simulações, duas simulações com mais dispositivos solicitando recursos das ECs.

Figura 12 – Gráfico de dispositivos bloqueados - Configuração com ECs.



Fonte: Produzido pelo autor.

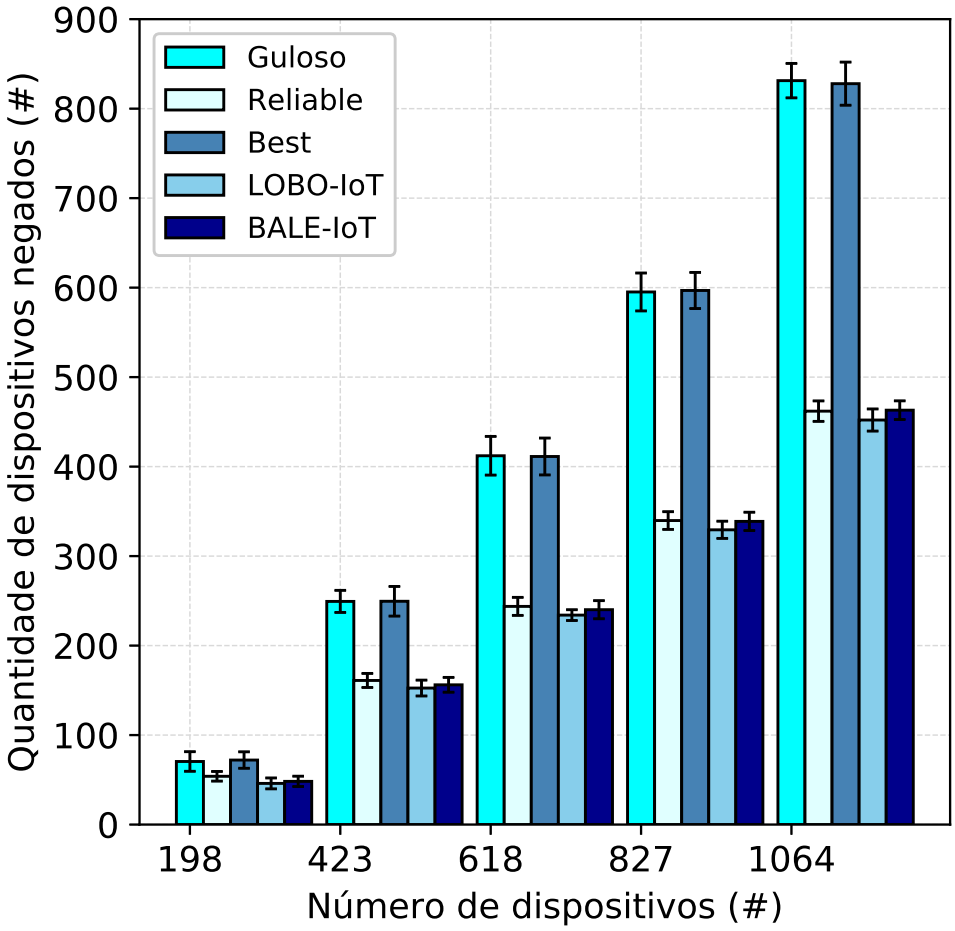
O método de inteligência e encurtamento para encontrar a melhor solução do BALE-IoT, mostrou-se o mais efetivo na distribuição dos recursos pelas ECs. Ele atendeu aos dispositivos, distribuindo de um modo mais igualitário seus recursos pelas ECs, assim, foi menos vezes bloqueado durante as simulações. O LOBO-IoT mostrou-se mais efetivo na distribuição dos recursos nos cenários com menos dispositivos. Isso se deve ao fato de ter conseguido encontrar a solução já na opção da EC_Alpha, sem tentar as outras opções.

Dispositivos negados são contabilizados quando os dispositivos não conseguem ser alocados em nenhuma EC, ou seja, eles passaram por todos os dispositivos, mas nenhuma EC tinha recursos suficientes para atendê-los. As comparações dos mecanismos LOBO-IoT e BALE-IoT, com métodos utilizados na literatura, são apresentadas na Figura 13. Na simulação com 198 dispositivos, o LOBO-IoT foi o que menos negou dispositivos, apenas **18.19%**. Seguido de perto pelo BALE-IoTque negou apenas **19.33%** dos dispositivos. O Reliable, o Guloso e o Best negaram 27.27%, 35.60% e 36.41% dos dispositivos na primeira simulação. Dos 423 dispositivos, requerentes na segunda simulação, o LOBO-IoT conseguiu atender apenas **33.70%**, seguido de perto pelo BALE-IoT e Reliable, que negaram **34.55%** e 38.07% dispositivos, respectivamente. O Best e o Guloso negaram aproximadamente 59.00% na segunda simulação.

Na terceira simulação, com 618 dispositivos, novamente o LOBO-IoT se mostrou ser o mais eficiente negando **36.26%**. O BALE-IoT ficou próximo, negando **37.24%** de dispositivos, enquanto o Reliable negou 39.45%. Novamente os dois últimos foram o Best e o Guloso, com aproximadamente 66.65% de dispositivos negados. Na quarta simulação com um cenário mais desafiador, o LOBO-IoT novamente negou menos dispositivos, apenas **38.62%**. Na sequência, vem o BALE-IoT, que negou **39.76%**. O Reliable, o Guloso e o Best negaram em média 41.08%, 71.97% e 72.17% de dispositivos, respectivamente. Na quinta configuração com 1064 dispositivos solicitando atendimento, o cenário foi mais desafiador. O LOBO-IoTnegou apenas **41.55%** dos dispositivos, enquanto o BALE-IoTnegou **42.58%**. O Reliable manteve sua média próxima dos mecanismos propostos, com 43.42% de negações. Enquanto o Best e o Guloso tiveram um desempenho ineficiente, negando em média e na sequência, 77.82% e 78.13% de dispositivos.

O mecanismo de caça do LOBO-IoT, que retorna as opções *EC_Alpha*, *EC_Beta* e *EC_Delta*, proporcionou maior eficiência no momento da tomada de decisão. O BALE-IoT também se beneficiou do seu processo de caça, para conseguir maior efetividade, com o objetivo de minimizar a quantidade de dispositivos que não conseguem alocar seus recursos nos serviços de borda. Novamente, a busca pela melhor opção, através da aptidão das ECs, apresenta ser eficiente por analisar diretamente as quantidade de recursos das ECs em relação aos dispositivos.

Figura 13 – Gráfico de dispositivos negados - Configuração com ECs.



Fonte: Produzido pelo autor.

4.6 Considerações parciais

Os mecanismos LOBO-IoT e BALE-IoT foram comparados com métodos de alocação de recursos tradicionalmente utilizados por pesquisadores. Ambos demonstraram ser eficientes, obtendo os melhores resultados nas avaliações das três métricas comparadas. O LOBO-IoT apresentou maior efetividade na maximização de atendimentos de usuários, mas foi seguido de perto pelo BALE-IoT, que também demonstrou ser efetivo para maximizar o atendimento de dispositivos e de um modo mais eficaz, bloqueando menos dispositivos. Os resultados demonstram que os processos dos mecanismos metaheurísticos bioinspirados, propostos neste trabalho, são eficientes por processar as buscas pelas melhores soluções, comparando os recursos das ECs e dos dispositivos ao preencher o vetor de populações. Desse modo, a função de aptidão é baseada diretamente nesta relação. Além disso, os mecanismos são facilmente customizados, caso modifique a quantidade de ECs e/ou dispositivos na simulação.

5 MECANISMOS PARA ALOCAÇÃO DE RECURSOS IOT APLICADOS EM COMPUTAÇÃO DE BORDA MÓVEL

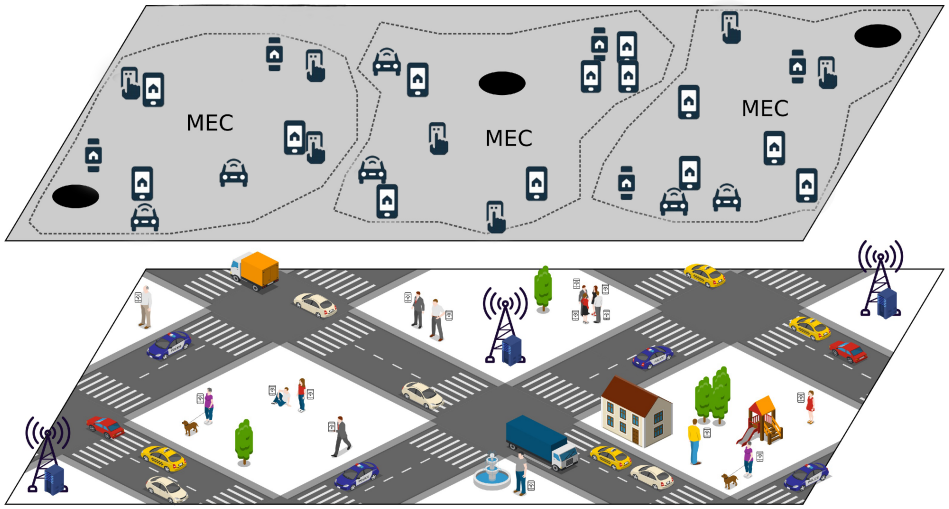
Neste capítulo, são apresentados os mecanismos LOBO-IoT e BALE-IoT para tomada de decisão em um cenário urbano com serviços de computação de borda móvel. Os mecanismos possuem fácil adaptação a cenários diferentes e conseguem atender a quantidade massiva de dispositivos IoT, com o objetivo de distribuir de modo eficiente os dispositivos nas MECs, que possuem limitação de recursos computacionais. Com base nessas limitações, as MECs são geradas com recursos ociosos dos dispositivos IoT e precisam de mecanismos que atenda o máximo de dispositivos e gerenciem os recursos das MECs de forma eficiente. Sendo assim, primeiramente é apresentado o cenário urbano com serviços de borda móvel e o modelo de sistema. Em seguida, é relatado como os mecanismos LOBO-IoT e BALE-IoT foram customizados para o cenário, as configurações das simulações e os resultados obtidos.

5.1 Cenário urbano com serviços de borda móvel

Nesta seção, são aplicados os mecanismos LOBO-IoT e BALE-IoT para um cenário de computação urbana que utiliza serviços de computação de borda móvel para simular os processos de alocação de recursos computacionais. É considerado um ambiente de mobilidade urbana, em que os veículos possuem dispositivos IoT embarcados, que compartilham seus recursos computacionais ociosos, formando um *pool* de recursos que são disponibilizados para auxiliar aqueles dispositivos que circulam pelo perímetro e necessitam de recursos computacionais para processar alguns serviços.

Na Figura 14 denota-se um cenário de computação urbana de cidade inteligente com veículos que possuem dispositivos IoT embarcados, tais como *smartphone*, relógios inteligentes, *tablets*, unidades *on-board*, entre outros, se movimentando ao redor do perímetro urbano. O ambiente é composto por N dispositivos IoT embarcados, em veículos que se comunicam entre si, e com os serviços de borda por uma rede 5G. Em algum momento, um dado dispositivo que não faz parte da MEC precisa processar alguns dados, mas tem limitações em seus recursos computacionais. Com isso, ele envia uma solicitação ao nó responsável pelo controle da MEC, que executa a política de otimização responsável por gerenciar a alocação de recursos.

Figura 14 – Abstração do modelo de sistema com serviço de borda móvel.



Fonte: Produzida pelo autor.

Considera-se cada MEC como $d_j \in D = \{d_1, \ldots, d_n\}$, que acata um subconjunto de dispositivos $D \subset U$, capaz de compartilhar até 3 de seus recursos computacionais, tais como i) processamento, ii) memória e iii) armazenamento. Em cada borda da rede, é implantada uma MEC com dispositivo controlador, conectado a diferentes estações de base para coordenar as múltiplas MECs. Neste contexto, quando determinado dispositivo u_i solicitar recursos para alocar seus serviços, o nó controlador fica responsável por executar o algoritmo de tomada de decisão para verificar se a MEC tem recursos suficientes para alocar o serviço. O objetivo é atender o máximo de dispositivos possíveis e minimizar o desperdício de recursos oferecidos.

O modelo de sistema é representado por um conjunto de dispositivos D com $\{s = 1, ..., n\}$ em que cada dispositivo D possui seus dados específicos e são representados como (id_s, $processamento_s$, $valor_s$). Em que id_s é a identificação de cada dispositivo, $processamento_s$ o valor considerado para a quantidade de processamento necessário para os recursos do dispositivo. E $valor_s$ é o valor de recompensa atribuído ao dispositivo, que corresponde ao valor do serviço e à quantidade de recursos computacionais alocados.

O propósito é maximizar a recompensa dos dispositivos alocados nas MECs. Assim, é definido que os dispositivos em D devem ser alocados em serviços de computação móveis $MECs$. O agrupamento da quantidade de processamento de cada MEC M representa a soma de processamento m_i ociosos compartilhados por cada dispositivo, representado por $d_i(i = 1, ..., z)$, que faz parte do MEC, sendo $\sum_{i=1}^{z} m_i \leq \forall d_i \in V$. O valor $valor_s$ de um dispositivo id_s é atribuído de acordo com sua necessidade de processamento $processamento_s$, então é contabilizado como o dobro do processamento de cada dispositivo. Assim, quanto maior a necessidade para processar os serviços de um dispositivo, maior será o valor de recompensa atribuído àquele dispositivo.

A abstração do mecanismo de operação para alocação de recursos computacionais para o cenário descrito nesta seção, é apresentada no Algoritmo 4. O mecanismo tem como parâmetro inicial a soma dos recursos de processamento disponíveis no conjunto de MECs ($recursosMEC$), linha 1. Verifica-se a lista de serviços de dispositivos (D) para serem armazenados (linha 2). Uma estrutura de repetição é utilizada para percorrer o conjunto de recursos ($recursosMEC$), enquanto tiver disponibilidade (linha3). Aplica-se, então, o mecanismo de decisão ($Algoritmo\_Decisao$) que recebe como parâmetro $recursosMEC$ e D, e retorna à lista s os dispositivos selecionados para ser atendidos (linha 4). Percorre-se a lista s, verificando se há recursos suficientes de processamento na MEC ($recursosMEC$), para atender o dispositivo selecionado (linhas 5 e 6).

Caso tenha disponibilidade, são adicionados o valor de recompensa ($D\_valor(s_i)$) na variável que soma o montante de recompensa ($soma\_ganho$) e a quantidade de processamento do dispositivo ($D\_processamento(s_i)$) na variável que soma o montante de processamento atendido ($soma\_processamento$), contabilizado mais um dispositivo atendido na variável qtd, retirado dos recursos disponíveis na MEC ($recursosMEC$) a quantidade de processamento do dispositivo atendido ($D\_processamento(s_i)$) e retirado da lista de dispositivos o dispositivo atendido $D.pop(s_i)$ (linhas 7 a 11). Após isso, verifica se a lista de dispositivo (D) já foi totalmente atendida para interromper a estrutura de repetição (linhas 13 e 14). Por fim, a função retorna os parâmetros de soma recompensa ($soma\_ganho$), quantidade de dispositivos atendidos (qtd) e quantidade de processamento ($soma\_processamento$) (linha 19).

Algorithm 4 Abstração do algoritmo com serviço de borda móvel.

```
1: recursosMEC ← sum(M_i)
2: if (D ≥ 0) then
3:    while (recursosMEC ≥ 0) do
4:       s ← Algoritmo_Decisao(recursosMEC, D)
5:       while (s) do
6:          if (recursosMEC − D_p(s_i) ≥ 0) then
7:             soma_ganho+ = D_valor(s_i)
8:             soma_processamento+ = D_processamento(s_i)
9:             qtd+ = 1
10:            recursosMEC− = D_processamento(s_i)
11:            D.pop(s_i)
12:         end if
13:         if (D == 0) then
14:            break
15:         end if
16:      end while
17:   end while
18: end if
19: return soma_ganho, qtd, soma_processamento
```

5.2 LOBO-IoT: Mecanismo de Otimização do Lobo Cinzento para Alocação de Recursos de Dispositivos IoT em serviços de borda móvel

Nesta seção, é descrita a operacionalização do mecanismo de otimização do **LOBO** cinzento para alocação de recursos de dispositivos **IoT**, em serviços de computação de borda móvel, chamado LOBO-IoT. O mecanismo para tomada de decisão é ajustado para otimizar a utilização de recursos das MECs e atender os serviços computacionais de dispositivos IoT. Por exemplo, o LOBO-IoT considera um conjunto de serviços de dispositivos a serem atendidos por uma MEC e é necessário escolher o melhor dispositivo, no caso, o líder do conjunto.

Para a simulação com serviços de borda móvel, considerado na seção 5.1 o LOBO-IoT, considera que os lobos são os dispositivos IoT e o objetivo é encontrar os líderes para armazenar na MEC. Os dispositivos consideram a MEC, sua presa, e baseiam sua caça na quantidade de recursos das MECs.

O processo de otimização, modelado por Mirjalili, Mirjalili e Lewis (2014), foi adaptado para atender as necessidades do LOBO-IoT no cenário com MECs, conforme o fluxograma, apresentado na Figura 15. As etapas ilustradas no fluxograma consideram três etapas do processo de caça do GWO, que são descritas abaixo:

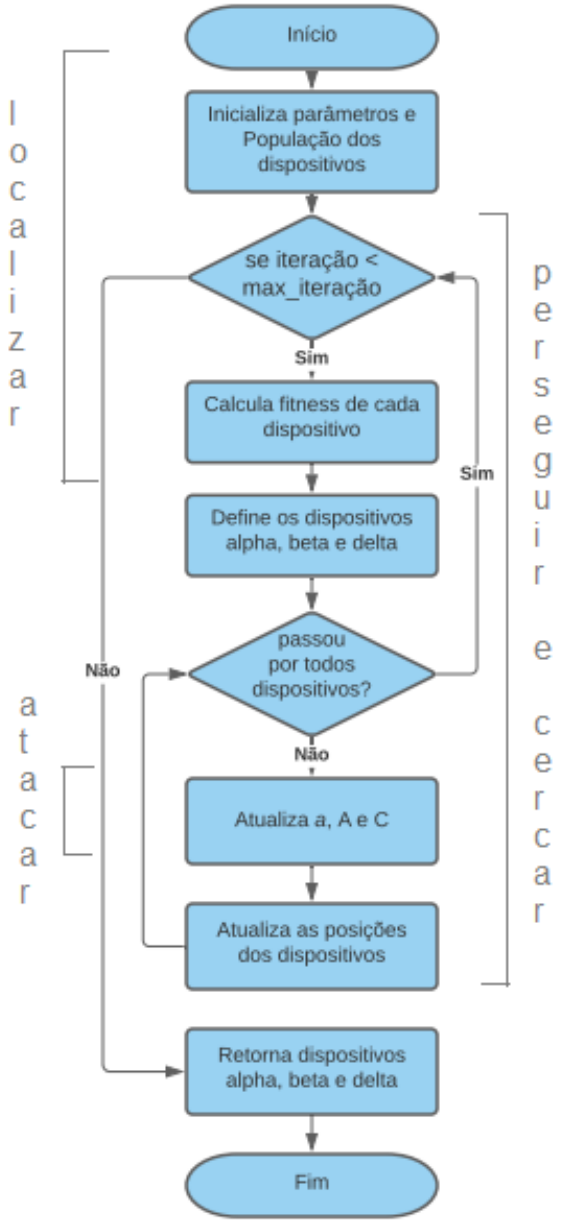
1. **localizar e perseguir a MEC:** abrange as etapas de inicialização das variáveis, preenchimento da população de dispositivos, baseado na diferença do valor do recurso da MEC, e na definição inicial da aptidão;

2. **perseguir e cercar a MEC:** consiste nos processos de iterações para atualização dos dispositivos *alpha*, *beta* e *delta* e atualizações das posições dos demais dispositivos, considerando suas aptidões, em cada iteração;

3. **atacar a MEC:** é a etapa que a variável a é decrementada a cada iteração, aproximando o dispositivo do valor ideal em relação a MEC.

O mecanismo, em primeiro lugar, identifica os três melhores dispositivos no conjunto D. O melhor dispositivo é definido como o *alpha* (α) inicial. Também são definidos o *beta* (β) e o *delta* (δ), que são, respectivamente, o segundo e o terceiro dispositivo com a melhor solução. Os demais dispositivos são considerados os *ômegas*. Define-se, assim, os guias responsáveis para o processo de caça à MEC do LOBO-IoT. O melhor dispositivo é atualizado com a média das aptidões dos três melhores, de acordo com a Equação 5.1 e

Figura 15 – Fluxograma dos processos de caça adaptado ao LOBO-IoT - Método com MEC.



Fonte: Produzido pelo autor.

será o responsável por guiar o reposicionamento dos demais dispositivos.

$$f(t+1) = \frac{f_1 + f_2 + f_3}{3} \tag{5.1}$$

Em que f corresponde à aptidão dos dispositivos identificadas e atualizadas durante as iterações.

O modelo matemático para o processo de caça, proposto por Mirjalili, Mirjalili e Lewis (2014), foi adaptado ao cenário desta seção. Portanto, para identificar a distância de uma MEC é calculada a distância entre os recursos do dispositivo e MEC. Isso permite o avanço em relação à melhor MEC, em cada iteração, sendo representado nas Equações 5.2 e 5.3, mostradas a seguir:

$$Disp = c2 * V_{mec}(i) - V(i) \tag{5.2}$$

$$V(i+1) = V_{mec}(i) - c1 * Disp \tag{5.3}$$

Sendo, i a iteração atual, V_{mec} o valor do recursos da MEC, V é o valor do serviço do dispositivo e $c1$ and $c2$ são coeficientes responsáveis pela movimentação e aproximação dos dispositivos em relação à MEC. Tais coeficientes são calculados de acordo com as Equações 5.4 e 5.5:

$$c1 = 2a * \tilde{s_1} - a \tag{5.4}$$

$$c2 = 2 * s_2 \tag{5.5}$$

Em que a é linearmente decrementado de 2 a 0, ao decorrer das iterações, para aproximar o dispositivo da MEC durante o cerco. As variáveis s_1 e s_2 são atualizadas com valores aleatórios entre 1 e 0, para fazer a movimentação dos dispositivos ao redor do melhor dispositivo, identificado no momento da iteração.

Durante a perseguição, os dispositivos se movimentam com a aplicação das Equações 5.2 e 5.3. A busca pela posição do melhor agente é feita com a atualização dos coeficientes $c1$ e $c2$. Após o cerco, os dispositivos iniciam o ataque guiados pelos líderes α, β e δ. As aptidões dos líderes ($f1$, $f2$ e $f3$) são atualizadas a cada iteração e, assim, podem se reposicionar em relação aos recursos da MEC. Para isso, utiliza-se as Equações 5.6, 5.7 e 5.8. Os demais dispositivos, definidos como ω, atualizam suas aptidões baseadas nos líderes.

$$f_1 = f\_\alpha(t) - c1_1 * Disp\_\alpha \tag{5.6}$$

$$f_2 = f_{-\beta}(t) - c1_2 * Disp\_\beta \tag{5.7}$$

$$f_3 = f_{-\delta}(t) - c1_3 * Disp\_\delta \tag{5.8}$$

Sendo que as variáveis $Disp\_\alpha$, $Disp\_\beta$ e $Disp\_\delta$ são definidas pelas Equações 5.9, 5.10 e 5.11, respectivamente.

$$Disp\_\alpha = c2 * f_{-\alpha}(t) - f(t) \tag{5.9}$$

$$Disp_{-\beta} = c2 * f_{-\beta}(t) - f(t) \tag{5.10}$$

$$Disp_{-\delta} = c2 * f_{-\delta}(t) - f(t) \tag{5.11}$$

Após o encurtamento dos recursos dos dispositivos em relação à MEC, o processo de ataque é realizado. A variável a é a responsável por isso. A cada iteração ela é decrementada, atualizando os coeficientes $c1$ e $c2$ para o reposicionamento. A Equação 5.12 apresenta o procedimento de encurtamento da variável a:

$$a = 2 - \left(\frac{a}{n\_iter}\right) \tag{5.12}$$

Em que a é decrementado de 2 a 0 e n_iter representa a quantidade de iterações definidas no algoritmo. A cada iteração, a é atualizada e permite a movimentação de $c1$. Cada decremento de a aproxima os dispositivos da melhor solução, uma vez que a próxima posição tende a ser entre a distância do melhor dispositivo atual e a MEC.

A abstração do Algoritmo 5 ,apresentada nesta seção, é responsável pela seleção dos dispositivos (*Algoritmo_Decisao*) para alocação de recursos no cenário do Algoritmo 4. O mecanismo possui como parâmetros os $recursosMEC$ e D que correspondem aos recursos disponíveis na MEC e a lista de dispositivos. São inicializados os parâmetros de número de iterações (n_iter) e a dimensão (dim) do vetor de população p (linha 1). Verifica-se, desse modo, o número de serviços de dispositivos (num_disp) solicitando recursos da MEC (linha 2). Faz-se um laço com duas estruturas de repetição (linhas 3 e 4) para preencher a população de agentes (p) com a distância euclidiana entre os recursos dos dispositivos (D_x) e os recursos da MEC ($recursosMEC$), na linha 5. Em seguida, inicia-se o processo de iterações em busca da melhor aptidão (linhas 8 e 9). Na linha 10 é calculada a aptidão e na linha 11 é feita a atualização dos agentes líderes Dis_alpha, Dis_beta e Dis_delta. O coeficiente a é atualizado para fazer o movimento de avanço

em relação aos recursos da MEC (linha 14). Subsequente, o processo de atualização das posições é realizado com duas estruturas de repetições para percorrer todos os dispositivos, passando por todos os recursos (linhas 15 e 16). Dentro das estruturas são atualizados os coeficientes $c1$ e $c2$, atualizadas as posições de *Disp_alpha*, *Disp_beta* e *Disp_delta*, e atualizadas as aptidões e as posições dos demais dispositivos (linhas 17, 18, 19 e 20, respectivamente). Terminadas as iterações, são retornados os três dispositivos selecionados *Disp_alpha*, *Disp_beta* e *Disp_delta* para alocar na MEC (linha 23).

Algorithm 5 LOBO-IoT

```
1: n_iter, dim
2: num_disp ← len(D)
3: while D do
4:    while dim do
5:       p(x, y) ← euclidiana(D_x, recursosMEC)
6:    end while
7: end while
8: while n_iter do
9:    while num_disp do
10:      calculaaaptidão
11:      atualiza(Disp_alpha, Disp_beta, Disp_delta)
12:   end while
13: end while
14: atualiza(a)
15: while num_disp do
16:   while dim do
17:      atualiza(c1, c2)
18:      atualiza(Disp_alpha, Disp_beta, Disp_delta)
19:      atualiza(f1, f2, f3)
20:      atualiza a posição do melhor dispositivo
21:   end while
22: end while
23: return Disp_alpha, Disp_beta, Disp_delta
```

5.3 BALE-IoT: Mecanismo Baseado no Algoritmo de Otimização da Baleia para Alocação de Recursos IoT em serviços de computação de borda móvel

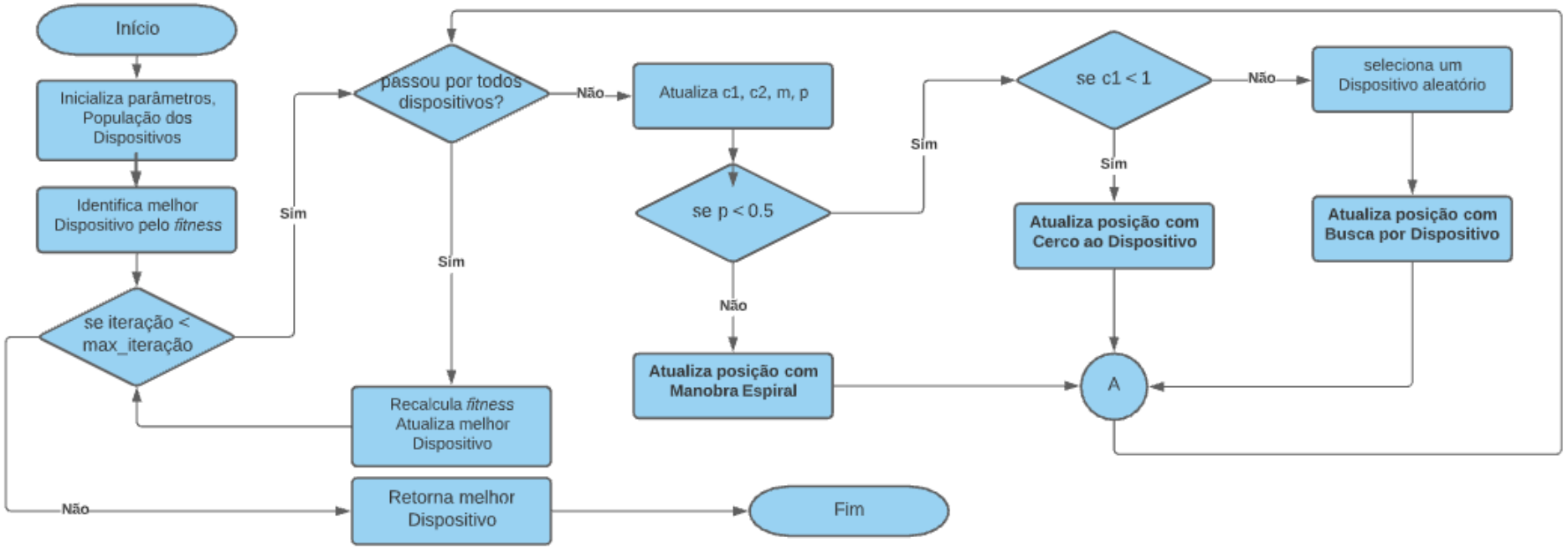
Nesta seção, o mecanismo baseado no algoritmo de otimização da **BALE**ia para alocação de recursos **IoT**, em computação de borda móvel chamado BALE-IoT, utiliza a base do WOA para tomar a decisão de alocação de recursos. Considera-se que as presas são as MECs e os dispositivos IoT embarcados em veículos que requerem recursos computacionais das MECs, são as baleias. O comportamento de caça à MEC é realizado para definir no final das iterações qual o melhor dispositivo para "atacar". A seguir, é detalhado todo o processo do mecanismo adaptado ao cenário descrito nesta seção.

As três principais manobras são realizadas pelo mecanismo do BALE-IoT, com base no dispositivo em busca da MEC. Os movimentos são:

- **Presa em círculo:** o dispositivo faz o movimento de circundação da MEC, buscando se aproximar do melhor dispositivo;
- **Manobra espiral de alimentação com rede de bolhas:** o mecanismo faz com os recursos do dispositivo os movimentos para cercar e avançar em direção MEC;
- **Busca por presa:** o dispositivo se posiciona com base num dispositivo selecionado de modo aleatório, com o objetivo de encontrar a MEC para se aproximar.

Os processos do BALE-IoT, para encontrar o melhor dispositivo para ser alocado em uma MEC, é apresentado no fluxograma da Figura 16. Inicialmente, os parâmetros são inicializados e a população de dispositivos (*pos*) preenchida. Posteriormente, as iterações são realizadas. Em cada iteração, todos os dispositivos são percorridos. Duas etapas são possíveis: *i*) mecanismo de encolhimento do círculo e *ii*) a atualização da espiral. Assume-se que a probabilidade de escolher entre uma etapa ou outra é de 50%. O mecanismo de encolhimento utiliza uma escolha aleatória para decidir se a atualização da posição será de forma aleatória ou baseada no melhor dispositivo. Finalizadas as iterações, o mecanismo retorna ao melhor dispositivo para ser alocado.

Figura 16 – Fluxograma dos processos de caça adaptado ao BALE-IoT - Método com MEC.



Fonte: Produzido pelo autor.

A primeira manobra é de localização e circundação da MEC. O mecanismo BALE-IoT inicializa a população dos indivíduos com os dados de recursos computacionais dos dispositivos. É calculada a aptidão dos dispositivos, a fim de detectar a melhor solução para o momento. Então, é utilizada a informação do melhor dispositivo para atualizar as posições dos outros. Esta etapa é realizada de acordo com as Equações 5.13 e 5.14.

$$D = c2 * Disp_d(x) - Disp(x) \tag{5.13}$$

$$Disp(x+1) = Disp_d(x) - c1 * D \tag{5.14}$$

Em que x é a iteração atual. A variável $Disp_d$ representa o atual dispositivo com melhor aptidão e pode ser atualizada durante as iterações. A variável D corresponde ao dispositivo que está se reposicionando na atual iteração. As variáveis $c1$ e $c2$ são os coeficientes que possibilitam alcançar posições diferentes ao redor do melhor agente e são calculadas pelas Equações 5.15 e 5.16.

$$c1 = 2a * r - a \tag{5.15}$$

$$c2 = 2 * r \tag{5.16}$$

Em que a é decrementado de 2 a 0, no decorrer das iterações, para realizar o cerco e r é um valor aleatório entre 0 e 1, que permite chegar a qualquer posição na vizinhança da melhor solução.

A segunda manobra é considerada a fase de exploração, em que se aplica o conceito de redes de bolhas. Duas formas são aplicadas:

- **Mecanismo de encolhimento de redução:** em que o coeficiente m segue a declínio da variável a a cada iteração, para se aproximar do dispositivo.
- **Posição de atualização da espiral:** aplica-se a Equação 5.17 para atualizar as posições com o movimento espiral, baseado nos recursos da MEC.

$$Disp(x+1) = D * e^{bl} * \cos(2\pi l) + Disp(x) \tag{5.17}$$

Em que D é o melhor dispositivo até o momento, b é um coeficiente para determinar a espiral logarítmica e l é um valor randômico de -1 a 1 para movimento pela vizinhança.

Para se aproximar da MEC, em cada iteração pode ser aplicado qualquer um dos movimentos da rede de bolhas, baseado na premissa de 50% de probabilidade. Define-se as Equações 5.14 e 5.17 para efetuar os movimentos de encolhimento de redução e atualização da espiral, respectivamente.

A terceira manobra do mecanismo é feita a busca randômica pelo melhor dispositivo. Isso ocorre com a movimentação da atualização de posições sendo realizada com base em um dispositivo escolhido de modo aleatório. As Equações 5.18 e 5.19 representam a terceira manobra.

$$D = n * Disp_{rand} - Disp \tag{5.18}$$

$$Disp(x+1) = Disp_{rand} - c1 * D \tag{5.19}$$

O algoritmo 6 foi desenvolvido para seleção do melhor dispositivo para alocar na MEC. Ele é utilizado para atender a função *Algoritmo_Decisao* do Algoritmo 4. O mecanismo têm como parâmetros a lista de dispositivo (D) que requisitam recursos da MEC e os recursos disponíveis na MEC ($recursosMEC$). Na linha 1, são inicializadas as variáveis n_iter e dim, que correspondem ao número de iterações e a dimensão dos recursos dos dispositivos, respectivamente. As estruturas de repetições das linhas 2 e 3 são responsáveis por percorrer as posições do vetor de população de agentes (pos) e preencher com a distância euclidiana entre os recursos do dispositivo e da MEC (linha 4). Depois, calcula-se a aptidão (linha 7) e verifica qual o $Dispositivo\_melhor$ para iniciar a perseguição (linha 8).

Em seguida, é inicializada a estrutura de decisão para o máximo de iterações definidas (n_iter), na linha 9. Em cada iteração, todos os dispositivos são percorridos (linha 10). Na linha 12, as variáveis responsáveis pelas movimentações (a, $c1$ e $c2$) e as variáveis que decidem o caminho do mecanismo (l e p) são atualizadas. A variável p representa a probabilidade para o algoritmo fazer o encurtamento ou atualizar a espiral. Se p menor que 0.5, o processo de encurtamento é realizado (linha 12). Dois processos são possíveis: i) a atualização da posição é efetuada baseada no posicionamento do $Dispositivo\_melhor$ atual, se o coeficiente $c1$ for menor que 1 (linha 14). Senão, ii) um dispositivo aleatório é escolhido para basear a próxima posição (linhas 16 e 17). Caso p seja maior ou igual a 0.5 (linha 19), o dispositivo se reposiciona com base no mecanismo de atualização com manobra espiral (linha 20). Após percorrer todas os dispositivos, no final de cada iteração, são atualizadas as posições dos agentes da população (aptidões), o $Dispositivo\_melhor$ até o momento e a iteração (linhas 23, 24 e 25). Finalizado o processo de iteração, é retornado o melhor dispositivo ($Dispositivo\_melhor$) identificado pelo mecanismo (linha 27).

Algoritmo 6 BALE-IoT

```
1: n_iter, dim
2: enquanto D faça
3:    enquanto dim faça
4:       p(x, y) ← euclidiana(D_x, recursosMEC)
5:    fim enquanto
6: fim enquanto
7: calcula a aptidão
8: identifica o Dispositivo_melhor
9: enquanto (t < n_iter) faça
10:   for cada_Dispositivo faça
11:      atualiza a, A, C, l, p
12:      se (p < 0.5) então
13:         se (A < 1) então
14:            atualiza posição com Eq. 1
15:         senão se (A ≥ 1) então
16:            seleciona um Dispositivo aleatório
17:            atualiza posição com Eq. 6
18:         fim se
19:      senão se (p ≥ 0.5) então
20:         atualiza posição com Eq. 5
21:      fim se
22:   fim for
23:   recalcula a aptidão
24:   atualiza Dispositivo_melhor (se tiver algum agente melhor)
25:   t = t + 1
26: fim enquanto
27: retorna Dispositivo_melhor
```

5.4 Simulação

Esta seção apresenta a aplicação dos mecanismos LOBO-IoT e BALE-IoT no cenário de computação de borda móvel, descrito na seção 5.1 e descreve como os experimentos foram conduzidos. Explica-se a configuração do ambiente, as métricas utilizadas para avaliação e as técnicas de comparação da literatura que foram utilizadas para o cenário com serviços de borda móvel. A seguir, descreve-se o ambiente configurado para simulação do processo de alocação de recursos nos serviços de computação de borda móvel e as técnicas de comparação utilizadas.

5.4.1 Configuração do cenário

Considera-se um cenário urbano, em que veículos possuem dispositivos IoT embarcados que compartilham seus recursos ociosos para formação de MECs. Para representar este cenário, é utilizado o mapa de Luxemburgo, apresentado na Figura 17. Para isso, utilizou-se o *trace* de mobilidade LuST (Luxembourg SUMO Traffic), que contém 24 horas de simulação em um perímetro urbano de 155.95km (CODECA; FRANK; ENGEL, 2015).

Para simular a mobilidade dos dispositivos IoT, utiliza-se o Simulador de Mobilidade Urbana (*Simulation of Urban Mobility* - SUMO)[1], versão 0.32.0. O SUMO utiliza a interface de controle de tráfego (TraCI - Traffic Control Interface), para integrá-lo com a linguagem de programação Python e desenvolver os mecanismos. Com isso, durante a execução de simulação de tráfico real, é possível acessar e manipular as informações (COSTA et al., 2020).

Figura 17 – Cenário LuST.

Fonte: Retirada do simulador de mobilidade urbana - SUMO.

Para formação da MEC, é realizada a clusterização com os recursos dos dispositivos móveis, em que um grupo de dispositivos compartilham seus próprios recursos ociosos, tais como capacidade de processamento, largura de banda e memória (ASLAM et al., 2021). Como a formação da MEC, neste cenário, só necessita ser identificada e disponibilizada, isto é feito com a técnica tradicional de clusterização espacial, baseada em densidade de aplicação com ruído (Density-Based Spatial Clustering of Application with Noise - DBSCAN) (COSTA et al., 2020).

O DBSCAN identifica os *clusters*, baseado na densidade espacial dos dispositivos, utilizando suas coordenadas e considera que a vizinhança de dispositivos deve estabelecer um limite para cada ponto em um *cluster*. Ele abrange um certo número de pontos para um raio estabelecido (ESTER et al., 1996). Foi determinado um período de clusterização de 60 segundos para medir o impacto da dinâmica nas mudanças da MECs e ter um efeito

[1] ⟨https://sumo.dlr.de/docs/index.html⟩

mais significativo na criação do *cluster* (COSTA et al., 2020). O raio considerado para formação do *cluster* foi de 100 metros.

A mobilidade dos dispositivos IoT (embarcados em veículos) foi utilizada a Distribuição de Poisson com uma média de dispositivos $\lambda = 25$, uma vez que suas ocorrências são independentes (KADHIM; SENO, 2019).

Um conjunto de dispositivos D possui serviços com valores de pesos específicos $peso = [1, 25, 50, 100, 200, 300]$. O ganho de alocação é atribuído, sendo o dobro do valor de peso $2 \times peso$. Também há uma variação na quantidade de serviços compartilhados por cada dispositivo com uma MEC [1,2,3]. Cada variação é considerada nas configuração 1, 2 e 3, respectivamente. Para validar as configurações desta simulação, foram levantados trabalhos da literatura, tal como o trabalho de Costa et al. (2020). Na Tabela 4 apresenta-se os parâmetros da configuração da simulação de modo resumido.

Tabela 4 – Configuração da simulação com serviços de borda móvel.

Parâmetros	**Valores**
Tempo de clusterização	60 segundos
Raio de abrangência	100 metros
Média do número de serviços λ	25
Capacidade média de processamento dos serviços	1, 25, 50, 100, 200, 300
Ganho médio dos serviços	2 x {1, 25, 50, 100, 200, 300}
Quantidade de serviços por dispositivos	1, 2, 3
Tempo de simulação	1h do cenário LuST
Perímetro	155,95km^2
Intervalo de confiança	95% (distribuição *t-student*)
Linguagem de programação	Python (versão 3.7)

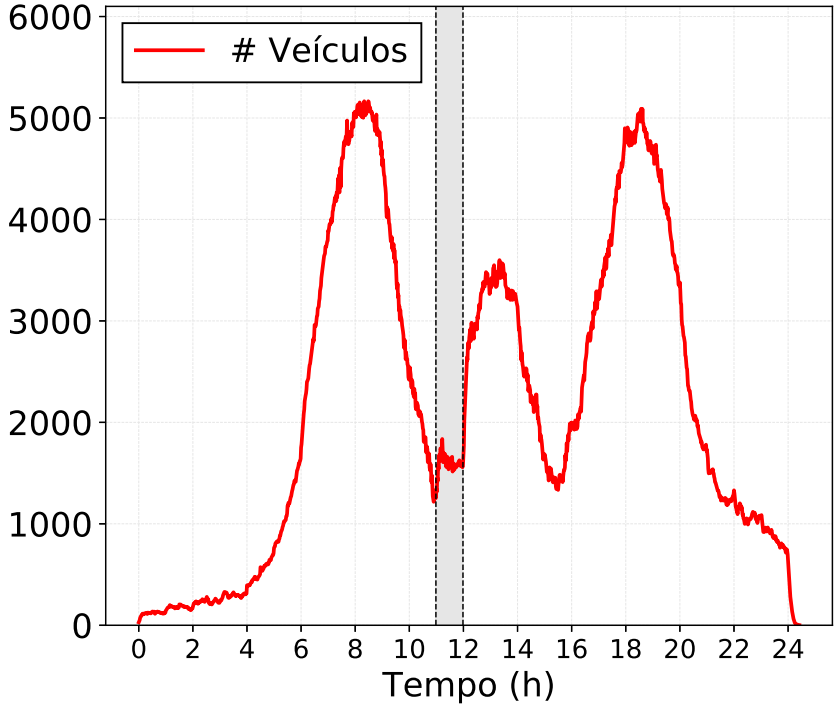
Fonte: Produzida pelo autor.

Neste trabalho, optou-se por considerar o período das 11am até às 12pm, período em que teve baixa quantidade de veículos circulando no cenário LuST, como indicado na Figura 18. Nesse sentido, é possível demonstrar que os mecanismos LOBO-IoT e BALE-IoT operam de modo satisfatório em um cenário mais desafiador, com menor mobilidade e, consequentemente, menos recursos disponíveis. Considerou-se que é necessário pelo menos 2 dispositivos para formar uma MEC. Baseado nisso, na Figura 19 mostra-se o número de MECs identificadas no trace e, com isso, pode-se observar que é inversamente proporcional à densidade veicular. Isso ocorre, porque o aumento da densidade veicular faz com que os veículos se juntem em um número menor de $MECs$, por estarem mais próximos.

5.4.2 Métricas de avaliação

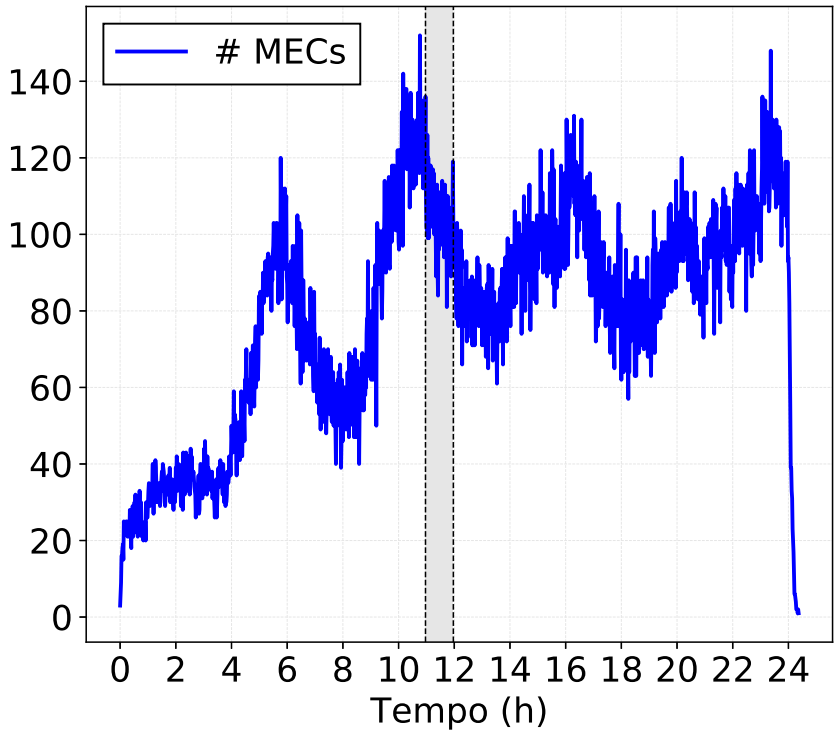
Neste cenário, a simulação é avaliada considerando as seguintes métricas:

Figura 18 – Densidade de veículos no cenário LuST.



Fonte: Produzido pelo autor.

Figura 19 – Quantidade de MECs identificadas.



Fonte: Produzido pelo autor.

- **Quantidade de serviços atendidos:** é uma métrica que representa o número de dispositivos atendidos durante a simulação. O propósito é maximizar a quantidade de dispositivos atendidos;
- **Utilização de recursos:** é uma métrica que verifica a quantidade de recursos utilizados de acordo com a disponibilidade fornecida pelas MECs. O objetivo é minimizar a utilização de recursos das MECs;
- **Valor de recompensa:** é uma métrica que contabiliza os valores de ganho de recompensa dos dispositivos atendidos. A meta é maximizar o valor de recompensa.

5.4.3 Técnicas de comparação

Foram utilizados métodos tradicionais, encontrados na literatura, para validar a simulação dos mecanismos LOBO-IoT e BALE-IoT, tais como os métodos de Programação Dinâmica e Guloso, dos respectivos trabalhos de (SACHDEVA; GOEL, 2014) e (CHOO; KIM; PACK, 2018). Os conceitos dos métodos são apresentados a seguir:

- **DP (*Dynamic Programming*):** o método DP ou Programação Dinâmica encontra a melhor solução com base na quantidade atribuída de processamento e valor dos serviços. No entanto, é direcionado para a MEC de maior capacidade;
- **Guloso:** este método considera o valor atribuído para a quantidade de processamento do serviço para se basear na escolha de modo guloso em todas as MECs disponíveis no momento.

5.5 Resultados

Após realizar as etapas de simulações, os dados resultantes foram analisados e processados para fins de validação, desempenho e eficiência dos mecanismos LOBO-IoT e BALE-IoT. Para validação das simulações e dos resultados, as comparações foram realizadas com cenário e métodos de alocação de recursos implementados e utilizados por pesquisadores ((SACHDEVA; GOEL, 2014), (CHOO; KIM; PACK, 2018), (COSTA et al., 2020)). Sendo assim, nesta seção são apresentados os resultados dos testes de aptidão realizados para selecionar a melhor função para os algoritmos meta-heurísticos bioinspirados, propostos neste trabalho, e a comparação dos resultados de alocação de recursos para a simulação com serviços de borda móvel.

Os parâmetros utilizados para os mecanismos LOBO-IoT e BALE-IoT são: *i*) O número de iterações n_iter foi definido como 30, por ser o número de iterações que os mecanismos se aproximaram do valor de aptidão ideal, e *ii*) a dimensão das matrizes

de populações *dim*, que recebe o tamanho do vetor de dispositivos e varia durante a simulação.

5.5.1 Comparação das funções de aptidão

Esta seção apresenta os resultados dos testes de aptidão. Para definir qual a melhor função de aptidão para os mecanismos propostos, foram realizadas simulações com as funções Ackley, Rastrigin e Rosenbrock, no cenário de computação de borda móvel. Para definir a melhor função de aptidão para a comparação com as métricas de alocação, optou-se por considerar o ganho de alocação, que representa a média do valor de recompensa obtido de acordo com o peso dos recursos alocados. Considerando ser essa a métrica que concilia a quantidade de dispositivos atendidos com o valor dos recursos alocados.

Na Tabela 5, é apresentada a média dos resultados obtidos nas simulações, considerando a quantidade de recursos compartilhados por dispositivos embarcados nos veículos [1, 2 e 3] durante a mobilidade e as funções de aptidão Ackley, Rastrigin e Rosenbrock.

Tabela 5 – Média de ganho de alocação por cada função de aptidão - Configuração com MECs.

	LOBO-IoT			**BALE-IoT**		
Recursos/Função	**Ackley**	**Rastrigin**	**Rosenbrock**	**Ackley**	**Rastrigin**	**Rosenbrock**
1	684.71	685.03	686.10	720.05	718.11	725.88
2	1493.51	1492.80	1497.91	1533.92	1532.06	1537.67
3	2294.18	2295.94	2303.36	2302.63	2301.45	2308.09

Fonte: Produzida pelo autor

Para o cenário de computação de borda móvel, é possível observar que, na simulação com 1 recurso compartilhado por dispositivos, a função Rosenbrock obteve uma leve vantagem em relação ao ganho médio de alocação em comparação com as funções Rastrigin e Rosenbrock no mecanismo LOBO-IoT, sendo **686.10** para o Rosenbrock, 685.03 para a Rastrigin e 684.71 para a Ackley. No BALE-IoT, a Rosenbrock também foi mais eficiente, com ganho médio de **752.88**, contra 720.05 e 718.11 da Ackley e Rastrigin, respectivamente. Para as simulações com 2 recursos compartilhados por dispositivos, no LOBO-IoT a Rosenbrock teve melhor performance com ganho de **1497.91**, seguido da Ackley com 1493.51 e Rastrigin com 1492.80.

A Rosenbrock também foi a melhor função no BALE-IoT, com ganho médio de **1537.67**, seguido de perto da Ackley e Rastrigin, com 1533.92 e 1532.06, respectivamente. Na terceira configuração, com 3 recursos, a Rosenbrock teve recompensa média de **2303.36** para o LOBO-IoT, sendo novamente maior que a Rastrigin e Ackley, com 2295.94 e 2294.18, respectivamente. Em relação ao BALE-IoT, a Rosenbrock também foi o melhor com ganho de **2308.09**, contra 2302.63 para a Ackley e 2301.45 para a Rastring. Os resultados demonstram que, neste cenário, com mais dispositivos e a MEC, a

função Rosenbrock apresentou melhor ganho de alocação médio e será considerada para a comparação dos resultados, em relação às métricas do cenário de computação de borda móvel para ambos os mecanismos.

5.5.2 Convergência dos mecanismos

Os algoritmos meta-heurísticos permitem analisar a convergência da função de aptidão, que consiste na evolução da função de aptidão até chegar no valor ideal na iteração final. Quanto menor o valor de aptidão, mais próximo o mecanismo está do valor considerado ideal. Neste sentido, são apresentadas as melhores convergências obtidas para o cenário da simulação com serviço de borda móvel, com base na melhor função de aptidão identificada nesta simulação.

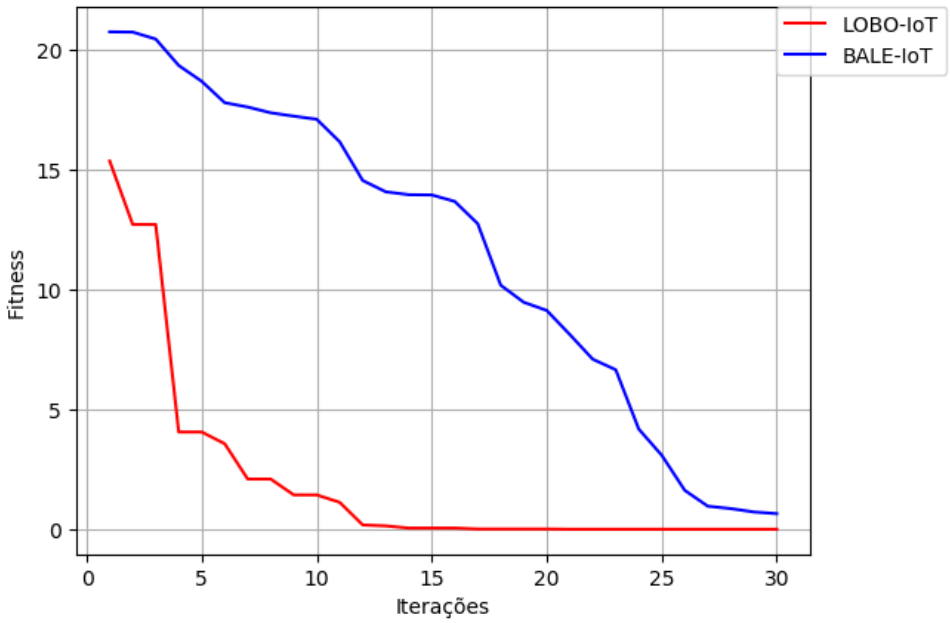
Na Figura 20 demonstra-se a evolução da curva de convergência dos mecanismos LOBO-IoT e BALE-IoT. Busca-se a melhor aptidão da população de dispositivos na simulação com MECs com a função Rosenbrock. Para isso, o objetivo é aproximar o valor de aptidão de 0. Neste caminho, o LOBO-IoT demonstrou conseguir rápida convergência com a população na simulação do cenário com serviço de móvel. Iniciou sua iteração com o valor de aptidão 15.36. Em pouco tempo, na quarta iteração, chegou no valor 4.08 e, na décima segunda iteração, o valor de aptidão chegou no valor de 0.21. A partir disso, a curva se manteve estável e chegou na trigésima iteração com o valor de **0.08**. Enquanto isso, o BALE-IoT, novamente, apresentou uma curva menos acentuada. Iniciou sua trajetória com o valor de aptidão definido com 20.74. Foi evoluindo em média pouco mais de 1.0 a cada três iterações, chegando na décima quinta iteração com o valor de 13.95. O ritmo do mecanismo se manteve até a vigésima terceira iteração, em que teve uma queda mais acentuada no valor, alcançando o valor de aptidão 4.2. Apenas na vigésima sexta iteração, chegou abaixo de 0, com o valor de 0.99. Terminou as 30 iterações com o valor de **0.68**.

O mecanismo LOBO-IoT apresentou boa convergência com a função de aptidão Rosenbrock, conseguindo se aproximar do valor ideal, obtido no final das iterações, antes da sua metade. Enquanto que o BALE-IoT demonstrou necessitar de mais iterações para se aproximar do valor ideal. Entretanto, isso não impediu a eficiência do mecanismo. As 30 iterações possibilitaram que o BALE-IoT se aproximasse da aptidão média ideal do LOBO-IoT.

5.5.3 Comparação entre as técnicas de alocação

Esta seção apresenta os resultados obtidos durante as simulações realizadas para a simulação com MECs. Neste cenário, os mecanismos LOBO-IoT e BALE-IoT utilizam a função Rosenbrock para cálculo de aptidão das populações. Foram comparadas as métricas de dispositivos atendidos, ganho de recompensa por alocação e utilização dos

Figura 20 – Gráfico de convergência - Configuração com MECs.



Fonte: Produzido pelo autor.

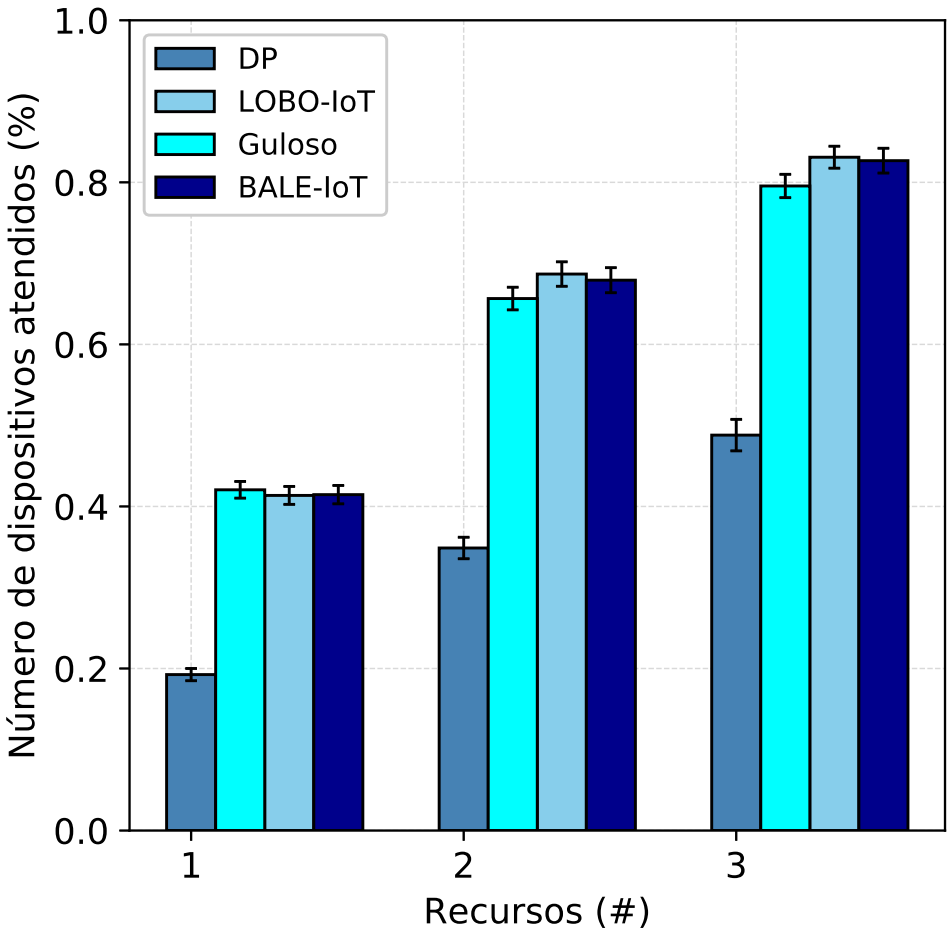
recursos disponíveis. Para as simulações, foram considerados que os dispositivos compartilham 1, 2 e 3 recursos com as MECs, denotadas aqui como configurações 1, 2 e 3, respectivamente.

Os dispositivos atendidos representam a eficiência dos métodos de tomadas de decisão para maximizar o atendimentos aos usuários dos dispositivos. Na Figura 21 mostra-se os resultados das comparações realizadas entre os mecanismos propostos e métricas utilizada na literatura. Na simulação com 1 recurso compartilhado por dispositivo, para formação das MECs, os mecanismos LOBO-IoT e BALE-IoT atenderam 41.36% e 41.46% dos dispositivos, respectivamente, bem próximos do Guloso que atendeu **42.05%** e melhores que o DP que atendeu somente 19.24%. No entanto, quando considerada a configuração com 2, o LOBO-IoT alocou **68.68%** e teve o melhor desempenho, seguido do BALE-IoT com **67.93%**, Guloso com 65.66% e DP com 34.87% de atendimento. Na configuração 3, os mecanismos LOBO-IoT e BALE-IoT novamente foram melhores com **83.10%** e **82.67%** de atendimentos, respectivamente, seguidos do Guloso com 79.55% e do DP com 48.80%.

O LOBO-IoT mostrou-se mais eficiente nas alocações das configurações 2 e 3 e ficou bem próximo do método Guloso e do BALE-IoT na configuração 1. Tal eficiência se dá pelo mecanismo propor sempre as 3 melhores opções de dispositivos para atendimento. O BALE-IoT acompanhou de perto a eficiência do LOBO-IoT, não atendendo menos que 1% do que o melhor mecanismo. Isso se dá pelo processo de busca pela melhor aptidão no momento de selecionar o dispositivo a ser alocado. Além disso, a definição das populações de ambos mecanismos com os valores da distância entre os recursos e as MECs, influencia

positivamente na seleção dos dispositivos durante as simulações.

Figura 21 – Gráfico de dispositivos atendidos - Configuração com MECs.



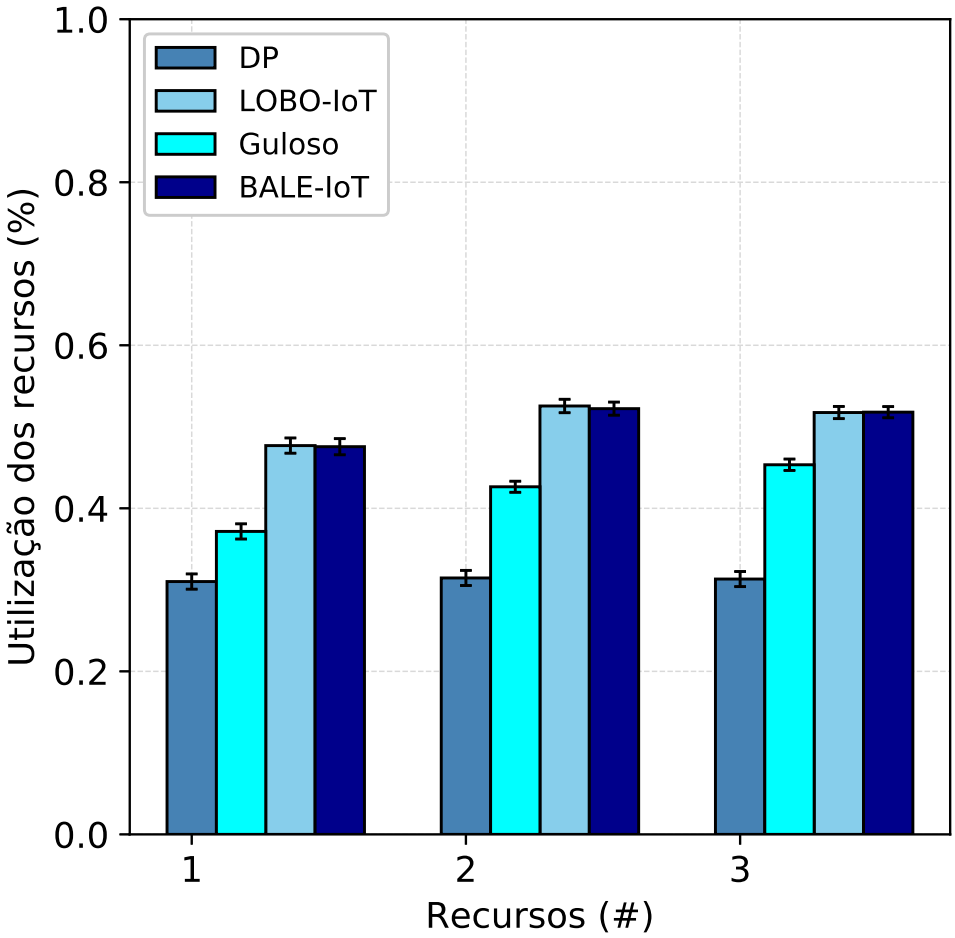
Fonte: Produzido pelo autor.

A métrica de recursos utilizados refere-se à quantidade de recursos que as MECs disponibilizaram e qual a média percentual foi utilizada. Neste caminho, na Figura 22 mostra-se os percentuais de utilização das métricas avaliadas. Na configuração 1, o DP utilizou **31.02%** de recursos, seguido do Guloso com 37.17% e dos mecanismos BALE-IoT e LOBO-IoT com 47.56 e 47.69, respectivamente. Na configuração 2, o DP utilizou **31.45%**, o Guloso 42.64%, o BALE-IoT 52.22% e o LOBO-IoT 52.55%. Por fim, na configuração 3, o DP utilizou **31.32%**, o Guloso 45.34%, o LOBO-IoT 51.75% e o BALE-IoT 51.79%.

A maior utilização dos recursos das MECs pelos mecanismos LOBO-IoT e BALE-IoT tem relação direta com a maior quantidade de dispositivos atendidos. Comparando-os ao método DP, pode-se observar que apesar de ter a menor utilização de recursos, foi o método que menos atendeu dispositivos, ou seja, os recursos das MECs ficaram ociosos. Isso mostra que os mecanismos propostos conseguem fazer uma boa utilização dos recursos, atendendo o máximo número de dispositivos com a menor quantidade de recursos que conseguir. Isso ocorre, porque a população dos mecanismos LOBO-IoT e

BALE-IoT é definida pela diferença dos recursos dos dispositivos em relação às MECs, sendo assim, a aptidão calculada é baseada nesta diferença e influencia diretamente na seleção dos dispositivos.

Figura 22 – Gráfico de recursos utilizados - Configuração com MECs.



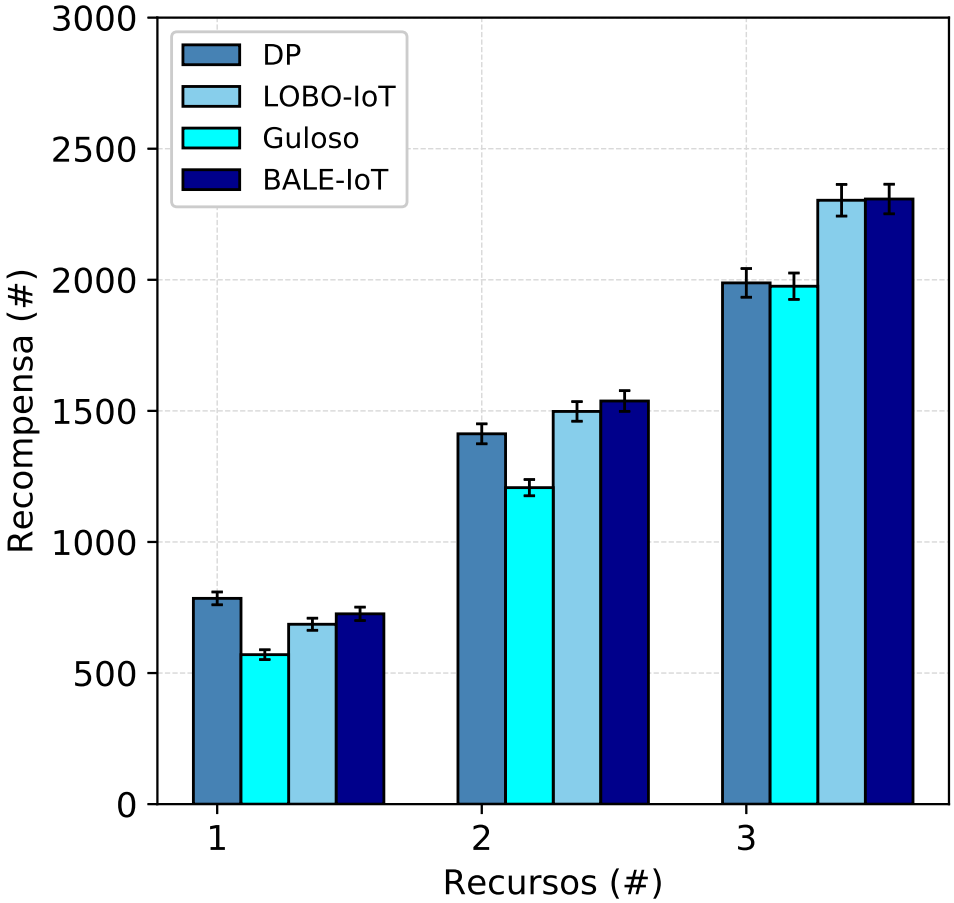
Fonte: Produzido pelo autor.

O ganho de alocação corresponde à recompensa atribuída aos valores dos recursos dos dispositivo. Na Figura 23, pode-se observar o valor médio de recompensa obtidos pelos métodos durante as simulações. O BALE-IoT obteve recompensa de **725.88**, na configuração 1, seguido doLOBO-IoT com **686.10**, DP com 584.86 e Guloso com 570.20. Na configuração 2, o BALE-IoT e LOBO-IoT tiveram maior recompensa de alocação com **1537.67** e **1497.91**, respectivamente, seguidos do DP com ganho de 1312.57 e Guloso com 1207.23. Por fim, na configuração 3 os mecanismos BALE-IoT e LOBO-IoT também tiveram melhor recompensa por alocação, com ganho respectivo de **2308.09** e **2303.36**. O DP obteve ganho de 1988.20 e o Guloso de 1975.46.

Os resultados mostram que os mecanismos BALE-IoT e LOBO-IoT conseguiram obter maior valor de recompensa, ou seja, alocaram os dispositivos com maior valor de processamento. Isso ocorre, porque ambos os mecanismos tiveram os seus cálculos de aptidão, baseados na relação do valor de processamento com os recursos disponíveis

nas MECs. Então, a busca pela melhor solução influencia diretamente na busca pelos dispositivos mais valiosos.

Figura 23 – Gráfico de ganho de alocação - Configuração com MECs.



Fonte: Produzido pelo autor.

5.6 Considerações parciais

Neste capítulo, os mecanismos LOBO-IoT e BALE-IoT foram comparados com métodos de alocação de recursos, tradicionalmente utilizados por pesquisadores, em serviços de borda móvel. Ambos demonstraram ser eficientes, obtendo os melhores resultados nas métricas de avaliação. Os mecanismos demonstraram conciliar bem o objetivo de maximizar o atendimento de dispositivos e minimizar a utilização de recursos das MECs. Isso se deve ao fato dos mecanismos relacionarem a diferença de recursos dos dispositivos entre os disponibilizados pelas MECs, no momento da população de agentes. Além disso, ambos apresentaram melhor ganho de recompensa quando os dispositivos compartilham mais recursos com as MECs. Por fim, o LOBO-IoT e BALE-IoT podem ser facilmente customizados para mudanças no cenário de simulação.

6 CONCLUSÃO

Cidades inteligentes tornam-se uma realidade cada vez mais presente no cotidiano das pessoas,que é proporcionado pelo significativo avanço tecnológico que ocorre no mundo. A tecnologia 5G e os dispositivos IoT contribuem significativamente para este crescimento tecnológico. No entanto, surgem desafios importantes para serem estudados e propostas soluções. Entre os vários desafios, há a necessidade do gerenciamento eficiente da massiva quantidade de dados gerados pelos dispositivos IoT nas redes 5G.

Este trabalho foi focado em encontrar soluções eficientes para gerenciar o processo de alocação de recursos computacionais de dispositivos IoT em cenários urbanos. Nesse sentido, foram apresentados dois mecanismos para alocação de recursos computacionais em serviços de computação de borda e borda móvel. Os mecanismos, denominados LOBO-IoT e BALE-IoT, são baseados nos métodos meta-heurísticos bioinspirados na natureza de otimização do lobo cinzento e do algoritmo de otimização da baleia, respectivamente.

Duas simulações foram utilizadas para validar a eficiência dos mecanismos LOBO-IoT e BALE-IoT. Na simulação com serviços de borda, os mecanismos mostraram-se eficientes na maximização do atendimento de dispositivos em comparação com técnicas tradicionais da literatura. Na simulação com serviços de borda móvel, foram analisadas a capacidade dos mecanismos em maximizar a utilização com o maior ganho de recompensa possível e minimizar a utilização dos recursos disponibilizados. Novamente, ambos se mostraram mais eficientes quando comparados à técnicas da literatura.

Além disso, os mecanismos LOBO-IoT e BALE-IoT demonstraram ser facilmente adaptáveis a diferentes cenários de simulações. Os métodos meta-heurísticos permitem esta simples adequação. Nesse sentido, podem ser utilizados em outros cenários, considerar diversas métricas e até ser implantados para tomadas de decisões em outras áreas.

6.1 Principais contribuições

Com esta pesquisa foi possível contribuir com a literatura. Dessa forma, nesta seção são listadas as contribuições dos mecanismos LOBO-IoT e BALE-IoT:

- Proposta da utilização de algoritmos meta-heurísticos para alocação de recursos de dispositivos IoT em cenários urbanos;
- Desenvolvimento dos mecanismos LOBO-IoT e BALE-IoT na linguagem de programação Python, com o objetivo de serem facilmente customizáveis;

- Implantação de um ambiente de simulação com dados gerados sinteticamente na simulação com serviços de borda;
- Implantação de um simulador de mobilidade urbana para criação de um ambiente realista na simulação com serviços de borda móvel;
- Maximização do atendimento de dispositivos IoT em serviços de borda com maior eficiência;
- Maximização no atendimento dos dispositivos IoT e no ganho de recompensa na simulação para serviço de borda móvel;
- Minimização da utilização de recursos disponibilizados conciliado com a maximização de atendimentos na simulação com serviços de borda móvel.

6.2 Produção Científica

- Lieira et al. (2021a). "Algorithm for 5G Resource Management Optimization in Edge Computing,"in IEEE Latin America Transactions, vol. 19, no. 10, pp. 1772-1780, Oct. 2021, doi: 10.1109/TLA.2021.9477278;
- Lieira et al. (2021). "TOVEC: Task Optimization Mechanism for Vehicular Clouds using Meta-heuristic Technique,"In: 2021 17th Int. Wireless Communications Mobile Computing Conference (IWCMC), 2021, pp. 1-6;
- Lieira et al. (2021b). "TRIAD: Whale Optimization Algorithm for 5G-IoT Resource Allocation Decision in Edge Computing,"2021 16th Iberian Conference on Information Systems and Technologies (CISTI), 2021, pp. 1-6, doi: 10.23919/CISTI52073.2021.9476599;
- Lieira et al. (2020). "Resource Allocation Technique for Edge Computing Using Grey Wolf Optimization Algorithm,"2020 IEEE Latin-American Conference on Communications (LATINCOM), 2020, pp. 1-6, doi: 10.1109/LATINCOM50620.2020.9282316.

6.3 Trabalhos Futuros

Com o desenvolvimento dos mecanismos, propostos neste trabalho, novas ideias são apresentadas para avançar as pesquisas de alocação de recursos computacionais com algoritmos meta-heurísticos, que serão listadas a seguir:

- Utilizar outros meta-heurísticos para buscar melhorar a eficiência de atendimento de dispositivos e minimizar a utilização de recursos computacionais;

- Implantar os algoritmos em um simulador de redes para analisar outros fatores como: tempo de resposta, interferências, oscilações de rede, entre outros;
- Desenvolver outros ambientes e considerar outras métricas da literatura para analisar a eficiência dos mecanismos desenvolvidos;
- Utilizar outros mecanismos para definição das aptidões, com o intuito de melhorar a convergência dos mecanismos;
- Implementar novos mecanismos utilizando métodos meta-heurísticos bioinspirados de modo híbrido com outros algoritmos meta-heurísticos ou utilizando aprendizado de máquina, como a biblioteca Hyperopt.

REFERÊNCIAS

AAZAM, M.; HARRAS, K. A.; ZEADALLY, S. Fog computing for 5g tactile industrial internet of things: Qoe-aware resource allocation model. *IEEE Transactions on Industrial Informatics*, v. 15, n. 5, p. 3085–3092, 2019.

AKPAKWU, G. A. et al. A survey on 5g networks for the internet of things: Communication technologies and challenges. *IEEE Access*, v. 6, p. 3619–3647, 2018.

ALRIKABI, H. et al. Analysis the efficient energy prediction for 5g wireless communication technologies. *International Journal of Emerging Technologies in Learning*, v. 14, 2019. ISSN 1863-0383.

ASLAM, S. et al. A machine learning approach to enhance the performance of d2d-enabled clustered networks. *IEEE Access*, v. 9, p. 16114–16132, 2021.

BOBEE, B. B.; ROBITAILLE, R. The use of the pearson type 3 and log pearson type 3 distributions revisited. *Water Resources Research*, Wiley Online Library, v. 13, n. 2, p. 427–443, 1977.

CHEN, Z. et al. Pso-ga-based resource allocation strategy for cloud-based software services with workload-time windows. *IEEE Access*, v. 8, p. 151500–151510, 2020.

CHENG, R. *5G: Everything you need to know about the wireless revolution*. 2020. Disponível em: ⟨https://www.cnet.com/news/5g-everything-you-need-to-know-about-the-wireless-revolution/⟩.

CHETTRI, L.; BERA, R. A comprehensive survey on internet of things (iot) toward 5g wireless systems. *IEEE Internet of Things Journal*, v. 7, n. 1, p. 16–32, 2020.

CHOO, S.; KIM, J.; PACK, S. Optimal task offloading and resource allocation in software-defined vehicular edge computing. In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. [S.l.: s.n.], 2018. p. 251–256.

CODECA, L.; FRANK, R.; ENGEL, T. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In: *VNC*. [S.l.: s.n.], 2015. p. 1–8.

COSTA, J. B. D. da et al. Combinatorial optimization-based task allocation mechanism for vehicular clouds. In: IEEE. *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. [S.l.], 2020. p. 1–5.

DONNO, M. D.; TANGE, K.; DRAGONI, N. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *IEEE Access*, v. 7, p. 150936–150948, 2019.

EJAZ, W. et al. Internet of things (iot) in 5g wireless communications. *IEEE Access*, v. 4, p. 10310–10314, 2016.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. *In Kdd*, p. 226–231, 1996.

FAGLIONI, B. *Mire as Estrelas*. 2020. Rio de Janeiro: Fábrika Produções. 1 CD (47min).

FICKERT, M. A novel lookahead strategy fordelete relaxation heuristics in greedy best-first search. In: *Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)*. [S.l.: s.n.], 2020.

FU, J.; CHEN, C. L. Y. The contribution and prospect of 5g technology to china's economic development. *Journal of Economic Science Research— Volume*, v. 3, n. 03, 2020.

GARDEN, R. W.; ENGELBRECHT, A. P. Analysis and classification of optimisation benchmark functions and benchmark suites. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. [S.l.: s.n.], 2014. p. 1641–1649.

GHAREHCHOPOGH, F. S.; GHOLIZADEH, H. A comprehensive survey: Whale optimization algorithm and its applications. *Swarm and Evolutionary Computation*, v. 48, p. 1 – 24, 2019. ISSN 2210-6502. Disponível em: ⟨http: //www.sciencedirect.com/science/article/pii/S2210650218309350⟩.

GOUDOS, S. K. et al. A novel design approach for 5g massive mimo and nb-iot green networks using a hybrid jaya-differential evolution algorithm. *IEEE Access*, v. 7, p. 105687–105700, 2019.

GSMA. *Stat: Mobile devices to lead internet use in 2025*. 2020. Disponível em: ⟨https://www.lsnglobal.com/news/article/23539/ stat-mobile-devices-to-lead-internet-use-in-2025⟩. Acesso em: 31.10.2020.

HASSAN, N.; YAU, K. A.; WU, C. Edge computing in 5g: A review. *IEEE Access*, v. 7, p. 127276–127289, 2019.

HOSSEINIOUN, P. et al. A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. *Journal of Parallel and Distributed Computing*, v. 143, p. 88 – 96, 2020. ISSN 0743-7315. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S074373152030023X⟩.

HU M. PATEL, D. S. N. S. Y. C.; YOUNG, V. Mobile edge computing-a key technology towards 5g. In: *ETSI white paper*. [S.l.: s.n.], 2015. v. 11, n. 11, p. 1–16.

HUI, H. et al. A new resource allocation mechanism for security of mobile edge computing system. *IEEE Access*, v. 7, p. 116886–116899, 2019.

HUSSAIN, F. et al. Machine learning for resource management in cellular and iot networks: Potentials, current solutions, and open challenges. *IEEE Communications Surveys Tutorials*, v. 22, n. 2, p. 1251–1275, 2020.

HUYNH, L. N. et al. Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 10, n. 1, p. 203, 2020.

IBM. *Telecom's 5G future - Creating new revenue streams and services with 5G, edge computing, and AI*. 2020. Disponível em: ⟨https://www.ibm.com/industries/ telecom-media-entertainment/resources/5g-revolution/res/Telecom$_$s$_$5G$\ _$future$_$RESE\ARCH$_$INSIGHTS$_$v2.pdf⟩. Acesso em: 31.10.2020.

JHA, A. A.; JAIN, S.; THENMALAR, S. Survey on grey wolf algorithm in resource allocation. *International Journal of Pure and Applied Mathematics - Special Issue*, v. 118, p. 201 – 206, 2018.

JIANG, C. et al. Machine learning paradigms for next-generation wireless networks. *IEEE Wireless Communications*, v. 24, n. 2, p. 98–105, 2017.

JIANG, K. et al. An improved binary grey wolf optimizer for dependent task scheduling in edge computing. In: *2019 21st International Conference on Advanced Communication Technology (ICACT)*. [S.l.: s.n.], 2019. p. 182–186.

JIHAD, S. et al. Multidimensional knapsack problem for resource allocation in a distributed competitive environment based on genetic algorithm. In: *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*. [S.l.: s.n.], 2019. p. 1–5.

KADHIM, A. J.; SENO, S. A. H. Maximizing the utilization of fog computing in internet of vehicle using sdn. *IEEE Communications Letters*, IEEE, v. 23, n. 1, p. 140–143, 2019.

KETYKó, I. et al. Multi-user computation offloading as multiple knapsack problem for 5g mobile edge computing. In: *2016 European Conference on Networks and Communications (EuCNC)*. [S.l.: s.n.], 2016. p. 225–229.

KUMAR, N.; ZEADALLY, S.; RODRIGUES, J. J. P. C. Vehicular delay-tolerant networks for smart grid data management using mobile edge computing. *IEEE Communications Magazine*, v. 54, n. 10, p. 60–66, 2016.

LI, J. et al. Deep reinforcement learning based computation offloading and resource allocation for mec. In: *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. [S.l.: s.n.], 2018. p. 1–6.

LI, S. et al. Joint admission control and resource allocation in edge computing for internet of things. *IEEE Network*, v. 32, n. 1, p. 72–79, 2018.

LI, X. et al. Optimizing resources allocation for fog computing-based internet of things networks. *IEEE Access*, v. 7, p. 64907–64922, 2019.

LI, X.; XU, L. D. A review of internet of things—resource allocation. *IEEE Internet of Things Journal*, v. 8, n. 11, p. 8657–8666, 2021.

LIANG, J.; SUGANTHAN, P.; DEB, K. Novel composition test functions for numerical global optimization. In: *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* [S.l.: s.n.], 2005. p. 68–75.

LIEIRA, D. D. et al. Tovec: Task optimization mechanism for vehicular clouds using meta-heuristic technique. In: *2021 17th Int. Wireless Communications Mobile Computing Conference (IWCMC)*. [S.l.: s.n.], 2021. p. 1–6.

LIEIRA, D. D. et al. Resource allocation technique for edge computing using grey wolf optimization algorithm. In: *2020 IEEE Latin-American Conference on Communications (LATINCOM)*. [S.l.: s.n.], 2020. p. 1–6.

LIEIRA, D. D. et al. Algorithm for 5g resource management optimization in edge computing. *IEEE Latin America Transactions*, v. 19, n. 10, p. 1772–1780, 2021.

LIEIRA, D. D. et al. Triad: Whale optimization algorithm for 5g-iot resource allocation decision in edge computing. In: *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2021. p. 1–6.

LIU, B.; LIU, C.; PENG, M. Resource allocation for energy-efficient mec in noma-enabled massive iot networks. *IEEE Journal on Selected Areas in Communications*, v. 39, n. 4, p. 1015–1027, 2021.

LIU, J. et al. Device-to-device communications achieve efficient load balancing in lte-advanced networks. *IEEE Wireless Communications*, v. 21, n. 2, p. 57–65, 2014.

LOLOS, K. et al. Adaptive state space partitioning of markov decision processes for elastic resource management. In: *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. [S.l.: s.n.], 2017. p. 191–194.

LOLOS, K. et al. Elastic resource management with adaptive state space partitioning of markov decision processes. *CoRR*, abs/1702.02978, 2017. Disponível em: ⟨http://arxiv.org/abs/1702.02978⟩.

MACH, P.; BECVAR, Z. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys Tutorials*, v. 19, n. 3, p. 1628–1656, 2017.

MAO, Y. et al. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, v. 19, n. 4, p. 2322–2358, 2017.

MENEGUETTE, R. I.; BOUKERCHE, A.; PIMENTA, A. H. M. Avarac: An availability-based resource allocation scheme for vehicular cloud. *IEEE Transactions on Intelligent Transportation Systems*, v. 20, n. 10, p. 3688–3699, 2019.

MENG, Y. et al. Advancing the state of the fog computing to enable 5g network technologies. *Sensors*, v. 20, 2020.

MIRJALILI, S.; LEWIS, A. The whale optimization algorithm. *Advances in Engineering Software*, v. 95, p. 51 – 67, 2016. ISSN 0965-9978. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0965997816300163⟩.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey wolf optimizer. *Advances in Engineering Software*, v. 69, p. 46 – 61, 2014. ISSN 0965-9978. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0965997813001853⟩.

MOSES, M. L.; KAARTHICK, B. Multiobjective cooperative swarm intelligence algorithm for uplink resource allocation in lte-a networks. *Transactions on Emerging Telecommunications Technologies*, v. 30, n. 12, p. e3748, 2019. E3748 ETT-19-0231.R1. Disponível em: ⟨https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3748⟩.

NABI, M. et al. Resource assignment in vehicular clouds. In: *2017 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2017. p. 1–6.

PANWAR, N.; SHARMA, S.; SINGH, A. K. A survey on 5g: The next generation of mobile communication. *Physical Communication*, v. 18, p. 64 – 84, 2016. ISSN 1874-4907. Special Issue on Radio Access Network Architectures and Resource Management for 5G. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1874490715000531⟩.

PEREIRA, R. S. et al. Reliable: Resource allocation mechanism for 5g network using mobile edge computing. *Sensors*, v. 20, n. 19, 2020.

PEREIRA, R. S. et al. A novel fog-based resource allocation policy for vehicular clouds in the highway environment. In: *proceeding of the 11th Latin-American Conference on Communications (LATINCOM)*. [S.l.: s.n.], 2019. p. 1–6.

POL, P.; PACHGHARE, V. K. of meta-heuristic optimization approaches: in virtue of grey wolf optimization. In: *2019 Global Conference for Advancement in Technology (GCAT)*. [S.l.: s.n.], 2019. p. 1–7.

QIAO, W.; YANG, Z. Modified dolphin swarm algorithm based on chaotic maps for solving high-dimensional function optimization problems. *IEEE Access*, v. 7, p. 110472–110486, 2019.

QUESSADA, M. S. et al. A bat bio-inspired mechanism for resource allocation in vehicular clouds. p. 1–6, 2021.

SABELLA, D. et al. Mobile-edge computing architecture: The role of mec in the internet of things. *IEEE Consumer Electronics Magazine*, v. 5, n. 4, p. 84–91, 2016.

SACHDEVA, C.; GOEL, S. An improved approach for solving 0/1 knapsack problem in polynomial time using genetic algorithms. In: *ICRAIE-2014*. [S.l.: s.n.], 2014. p. 1–4.

SCHNEIDER, S. et al. Machine learning for dynamic resource allocation in network function virtualization. In: *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. [S.l.: s.n.], 2020. p. 122–130.

SHAO, Y. et al. Cost-effective replication management and scheduling in edge computing. *Journal of Network and Computer Applications*, v. 129, p. 46 – 61, 2019. ISSN 1084-8045. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1084804519300013⟩.

SHI, R. et al. Mdp and machine learning-based cost-optimization of dynamic resource allocation for network function virtualization. In: *2015 IEEE International Conference on Services Computing*. [S.l.: s.n.], 2015. p. 65–73.

SUN, J. et al. A mab-based knapsack algorithm for coexistence of lte/wifi in resource allocation optimization. In: *2020 IEEE 20th International Conference on Communication Technology (ICCT)*. [S.l.: s.n.], 2020. p. 421–424.

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

VALLURU, S. K.; SINGH, M. Multi-objective genetic and adaptive particle swarm optimization algorithms: A performance analysis with benchmark functions. In: *2nd IEEE ICPEICES*. [S.l.: s.n.], 2018. p. 847–851.

WAN, S. et al. Efficient computation offloading for internet of vehicles in edge computing-assisted 5g networks. In: *The Journal of Supercomputing*. [S.l.: s.n.], 2020. p. 2518–2547.

WANG, B. et al. A reliable iot edge computing trust management mechanism for smart cities. *IEEE Access*, v. 8, p. 46373–46399, 2020.

WANG, D.; TAN, D.; LIU, L. *Particle swarm optimization algorithm: an overview*. 2018. Disponível em: ⟨https://doi.org/10.1007/s00500-016-2474-6⟩.

WANG, J. et al. Smart resource allocation for mobile edge computing: A deep reinforcement learning approach. *IEEE Transactions on Emerging Topics in Computing*, 2019.

WANG, L.; ZHENG, X. long; WANG, S. yao. A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. *Knowledge-Based Systems*, v. 48, p. 17 – 23, 2013. ISSN 0950-7051. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0950705113001159⟩.

WANG, Q.; ZHOU, F. Fair resource allocation in an mec-enabled ultra-dense iot network with noma. In: *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. [S.l.: s.n.], 2019. p. 1–6.

WEERASINGHE, T. N. et al. Mdp-based resource allocation for uplink grant-free transmissions in 5g new radio. In: *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. [S.l.: s.n.], 2020. p. 1–6.

WONG, W.; MING, C. I. A review on metaheuristic algorithms: Recent trends, benchmarking and applications. In: *2019 7th International Conference on Smart Computing Communications (ICSCC)*. [S.l.: s.n.], 2019. p. 1–5.

XU, J. et al. Zenith: Utility-aware resource allocation for edge computing. In: *2017 IEEE International Conference on Edge Computing (EDGE)*. [S.l.: s.n.], 2017. p. 47–54.

XU, Y. et al. A survey on resource allocation for 5g heterogeneous networks: Current research, future trends, and challenges. *IEEE Communications Surveys Tutorials*, v. 23, n. 2, p. 668–695, 2021.

YANG, X.-S.; HE, X.-S.; FAN, Q.-W. Chapter 7 - mathematical framework for algorithm analysis. In: YANG, X.-S. (Ed.). *Nature-Inspired Computation and Swarm Intelligence*. Academic Press, 2020. p. 89–108. ISBN 978-0-12-819714-1. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/B9780128197141000178⟩.

YOUSEFPOUR, A. et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, v. 98, p. 289 – 330, 2019. ISSN 1383-7621. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S1383762118306349⟩.

ZHANG, C.; ZHENG, Z. Task migration for mobile edge computing using deep reinforcement learning. *Future Generation Computer Systems*, v. 96, p. 111 – 118, 2019. ISSN 0167-739X. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0167739X18329674⟩.

ZHANG, K. et al. Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. *IEEE Access*, v. 4, p. 5896–5907, 2016.

ZHAO, L. et al. Optimal edge resource allocation in iot-based smart cities. *IEEE Network*, v. 33, n. 2, p. 30–35, 2019.

ZHAO, W. et al. Secure energy-saving resource allocation on massive mimo-mec system. *IEEE Access*, v. 8, p. 137244–137253, 2020.

ZHENG, K. et al. An smdp-based resource allocation in vehicular cloud computing systems. *IEEE Transactions on Industrial Electronics*, p. 7920–7928, 2015.