

Euclydes Nasorri Gottsfritz

RAPTOR: Um Algoritmo Meta-heurístico para Alocação de Recursos em Computação de Borda Veicular

São José do Rio Preto

2023

Euclydes Nasorri Gottsfritz

RAPTOR: Um Algoritmo Meta-heurístico para Alocação de Recursos em Computação de Borda Veicular

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.

Financiadora: CAPES.

Orientador: Prof. Dr. Rodolfo Ipolito Meneguette

São José do Rio Preto

2023

G687r

Gottsfritz, Euclydes Nasorri

Raptor: um algoritmo meta-heurístico para alocação de recursos em
computação de borda veicular / Euclydes Nasorri Gottsfritz. -- São José do
Rio Preto, 2023

99 p. : 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. Redes ad hoc veiculares (Redes de
computadores). 3. Alocação de recursos. 4. Algoritmos meta-heurísticos. 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.

EUCLYDES NASORRI GOTTSFRITZ

RAPTOR: Um Algoritmo Meta-heurístico para Alocação de Recursos em Computação de Borda Veicular

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.

Financiadora: CAPES.

Comissão examinadora

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

Prof. Dr. Geraldo Francisco Donega Zafalon
UNESP - Câmpus São José do Rio Preto – SP

Prof. Dr. José Rodrigues Torres Neto
UFPI - Teresina – PI

São José do Rio Preto

11 de agosto 2023

Dedico este trabalho aos meus pais que sempre me apoiaram em minhas decisões e à minha esposa que sempre acreditou e me incentivou a trilhar o caminho do conhecimento.

AGRADECIMENTOS

Gostaria de agradecer a Deus que me iluminou e guiou através dessa jornada cheia de obstáculos e que por vezes pensei que não conseguiria.

Ao IFSP, que por meio da sua política de incentivo à qualificação, fui capaz de me ausentar das atividades laborais para me dedicar exclusivamente ao curso de Mestrado.

Aos meus pais, Paulo e Vera, que sempre me apoiaram nos estudos e me ensinaram a nunca desistir dos meus sonhos.

Ao meu irmão, Paulo, que sempre pude contar nas horas de dificuldades.

Agradeço a minha esposa, que mesmo diante de todos os problemas que enfrentei, sempre acreditou em mim e foi responsável por ministrar diariamente a minha dose de paciência e perseverança.

Aos meus amigos de mestrado e companheiros de pesquisa Douglas e Matheus, que me auxiliaram diretamente no desenvolvimento do meu projeto e varias vezes se reuniram comigo para auxiliar em dificuldades encontradas.

E finalmente, ao meu orientador e amigo, o Prof. Rodolfo Ipólito Meneguette, por ter me incentivado a ingressar no mestrado e por orientar desde a primeira matéria como aluno especial pela UNESP.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

“Existem muitas hipóteses em ciência que estão erradas. Isso é perfeitamente aceitável, elas são a abertura para achar as que estão certas.”

(Carl Sagan, 1980)

RESUMO

A rápida evolução dos dispositivos móveis, seus aplicativos e a quantidade de dados que eles geram causaram um aumento significativo no consumo de largura de banda e também congestionamento no núcleo da rede. A Computação de Borda oferece uma solução parcial para as desvantagens de desempenho, com a extensão da nuvem até a borda da rede com ou sem a utilização de RSUs, que são estruturas com poder computacional, de armazenamento e comunicação. Com essa inspiração, a Computação de Borda Veicular surge para oferecer suporte a aplicativos veiculares. Esse novo paradigma depende de veículos como nós de extremidade para fornecer recursos de memória, armazenamento, processamento e rede para aplicativos de outros veículos com recursos limitados. Porém, essa alocação de recursos gera um problema adicional, os nós têm alta mobilidade pela rede e a capacidade de processamento está dispersa pela mesma. A partir daí, surge a necessidade de métodos eficientes que apoiem a tomada de decisão durante a alocação desses recursos, para que essa alocação seja feita em tempo real, sem atrasos e sem desperdício de poder computacional. Este trabalho propõe o RAPTOR, um algoritmo meta-heurístico para alocação de recursos em Computação de Borda Veicular. O algoritmo é baseado no método da Otimização do Falcão Harris e tem a finalidade de otimizar a escolha do melhor veículo para ter seus recursos alocados à um serviço, com vista para a maximização de serviços atendidos e redução da subutilização de recursos. Também estudou-se os tipos de Computação de Borda e alguns métodos de otimização meta-heurísticos de soluções múltiplas, para tomada de decisão na alocação de recursos. A partir dos resultados, é possível observar que no cenário estabelecido (LuST) com *TraCI* real de mobilidade, o RAPTOR obteve melhores resultados que todos os seus concorrentes, com desempenho de 29% à 36% superior ao segundo colocado em duas das três classes de serviços estabelecidas (“Baixo” e “Médio”), considerando apenas os dados de “Serviços Alocados”, isso nos três diferentes coeficientes de chegada de serviço. Para a última classe de serviços (“Alto”), essa margem foi um pouco mais discreta, de 6% à 14% superior ao segundo colocado.

Palavras-chave: Alocação de recursos. Redes veiculares. Computação de borda veicular. Algoritmos meta-heurísticos. Otimização do falcão Harris.

ABSTRACT

The rapid evolution of mobile devices, their applications, and the amount of data they generate has caused a significant increase in bandwidth consumption and congestion at the core of the network. Edge Computing offers a partial solution to the performance disadvantages, with the extension of the cloud to the edge of the network with or without the use of RSUs, which are structures with computational, storage, and communication power. With that inspiration, Vehicular Edge Computing emerged to support vehicular applications. This new paradigm relies on vehicles as endpoints to provide memory, storage, processing, and network resources to resource-constrained applications on other vehicles. However, this allocation of resources generates an additional problem, the nodes have high mobility through the network and the processing capacity is dispersed throughout it. From there, the need arises for efficient methods that support decision-making during the allocation of these resources, so that this allocation is done in real time, without delays, and without wasting computational power. This work proposes the RAPTOR, a meta-heuristic algorithm for resource allocation in Vehicular Edge Computing. The algorithm is based on the Harris Hawk Optimization method and aims to optimize the choice of the best vehicle to have its resources allocated to a service, with a view to maximizing services served and reducing the underutilization of resources. We also studied the types of Edge Computing and some meta-heuristic optimization methods of multiple solutions, for decision making in resource allocation. From the results, it is possible to observe that in the established scenario (LuST) with real mobility *TraCI*, the RAPTOR obtained better results than all its competitors, with a performance of 29% to 36% superior to second place in two of the three classes of services established (“Low” and “Medium”), considering only the data of “Allocated Services”, this in the three different service arrival coefficients. For the last class of services (“High”), this margin was a little more discreet, from 6% to 14% higher than the second place.

Keywords: Resource allocation. Vehicular networks. Vehicular Edge Computing. Meta-heuristic algorithms. Harris Hawk Optimization

LISTA DE FIGURAS

Figura 1 – Conectividade da Edge Computing.	20
Figura 2 – Funcionamento de uma MEC.	21
Figura 3 – Arquitetura de uma VEC.	23
Figura 4 – Exemplo de mobilidade de uma VEC.	24
Figura 5 – Classificação dos métodos meta-heurísticos de soluções múltiplas. . . .	30
Figura 6 – Execução da Rede de Bolhas da baleia Jubarte.	33
Figura 7 – Coeficientes e projeção de movimentação da baleia Jubarte.	33
Figura 8 – Exemplo de Orientação Transversal.	35
Figura 9 – Voo em espiral com fonte luminosa muito próxima.	35
Figura 10 – Movimento da mariposa em espiral logarítmica.	36
Figura 11 – Ecolocalização dos morcegos.	38
Figura 12 – Convergência e passeio aleatório dos vaga-lumes.	41
Figura 13 – Fluxograma do FA.	43
Figura 14 – Fluxograma do Algoritmo CS.	45
Figura 15 – Fluxograma do Algoritmo de Otimização do Cuco.	46
Figura 16 – Exemplo de imigração de um grupo de Cucos.	47
Figura 17 – Raio de postura de ovos de um Cuco.	47
Figura 18 – Iteração do esquemas de partículas	49
Figura 19 – Descrição das fases de caça do algoritmo HHO.	52
Figura 20 – Hierarquia social dos Lobos Cinzentos.	53
Figura 21 – Movimentação durante a perseguição à presa.	54
Figura 22 – Cerco e assédio à presa.	55
Figura 23 – Exemplo de convergência de resultados.	55
Figura 24 – Abstração do cenário de <i>Edge</i> com representação de comunicação. . . .	66
Figura 25 – Descrição das fases de execução do RAPTOR.	68
Figura 26 – Fluxograma do algoritmo RAPTOR.	76
Figura 27 – Cenário utilizado na simulação, mapa da cidade de Luxemburgo - LU. .	77
Figura 28 – Gráfico de Serviços Alocados com Classe de Serviços Baixo	81
Figura 29 – Gráfico de Serviços Não Alocados com Classe de Serviços Baixo	81
Figura 30 – Gráfico de Quantidade de Recursos com Classe de Serviços Baixo . . .	82
Figura 31 – Gráfico de Serviços Alocados com Classe de Serviços Médio	83
Figura 32 – Gráfico de Serviços Não Alocados com Classe de Serviços Médio	83
Figura 33 – Gráfico de Quantidade de Recursos com Classe de Serviços Médio . . .	84
Figura 34 – Gráfico de Serviços Alocados com Classe de Serviços Alto	84
Figura 35 – Gráfico de Serviços Não Alocados com Classe de Serviços Alto	85
Figura 36 – Gráfico de Quantidade de Recursos com Classe de Serviços Alto	86

LISTA DE TABELAS

Tabela 1 – Critérios de importância definidos por Saaty.	28
Tabela 2 – Comparações dos Trabalhos Relacionados	62
Tabela 3 – Parâmetros definidos para a simulação	79

LISTA DE ABREVIATURAS E SIGLAS

AHP	<i>Analytic Hierarchy Process</i>
BA	<i>Bat Algorithm</i>
BBO	<i>Biogeography-Based Optimization</i>
BS	<i>Base Station</i>
CC	<i>Cloud Computing</i>
CN	<i>Core Network</i>
COA	<i>Cuckoo Optimization Algorithm</i>
CR	<i>Cognitive Radio</i>
CS	<i>Cuckoo Search</i>
DCOSS	<i>International Conference on Distributed Computing in Sensor Systems</i>
DE	<i>Differential Evolution</i>
EC	<i>Edge Computing</i>
EFO	<i>Electromagnetic Field Optimization</i>
ELR	<i>Egg Laying Radius</i>
ETSI	<i>European Telecommunications Standards Institute</i>
FA	<i>Firefly Algorithm</i>
FENABRAVE	Federação Nacional de Distribuição de Veículos Automotores
GA	<i>Genetic Algorithm</i>
GDS	<i>Gaussian Distribution Strategies</i>
GSA	<i>Gravitational Search Algorithm</i>
GWO	<i>Gray Wolf Optimizer</i>
GWOA	<i>Genetic Whale Optimization Algorithm</i>
HHO	<i>Harris Hawk Optimization</i>
HODA	<i>Heuristic Offloading Decision Algorithm</i>

ICC	<i>IEEE International Conference on Communications</i>
ICN	<i>Information-Centered Networks</i>
IEA	<i>International Energy Agency</i>
IoV	<i>Internet of Vehicles</i>
ISCC	<i>IEEE Symposium on Computers and Communications</i>
ITS	<i>Intelligent Transportation System</i>
IoT	<i>Internet of Things</i>
LatinCom	<i>Latin-American Conference on Communication</i>
LF	<i>Levy Flyght</i>
LID	<i>Lower ID Algorithm</i>
LuST	<i>Luxembourg SUMO Traffic</i>
MEC	<i>Multi-Access Edge Computing</i>
MFO	<i>Moth-Flame Optimization</i>
NFV	<i>Network Function Virtualization</i>
NLOS	<i>Non-Line-Of-Sight</i>
OBL	<i>Opposition-based Learning</i>
OBU	<i>On-Board Unit</i>
OICA	<i>International Organization of Motor Vehicle Manufacturer</i>
QoS	<i>Quality of Service</i>
PSO	<i>Particle Swarm Optimization</i>
RAN	<i>Radio Access Network</i>
RSU	<i>RoadSide Unit</i>
SA	<i>Simulated Annealing</i>
SBRC	<i>Simpósio Brasileiro de Redes de Computadores</i>
SDN	<i>Software-Defined Networks</i>
SI	<i>Swarm-based Intelligence</i>

UE	<i>User Equipment</i>
V2R	<i>Vehicle-to-RSU</i>
V2V	<i>Vehicle-to-Vehicle</i>
V2X	<i>Vehicles-to-Everything</i>
VANET	<i>Vehicular Ad Hoc Network</i>
VCC	<i>Vehicular Cloud Computing</i>
VEC	<i>Vehicular Edge Computing</i>
Veins	<i>VEhicles In Network Simulation</i>
WOA	<i>Whale Optimization Algorithm</i>

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Justificativa e motivação	16
1.2	Objetivos	17
1.3	Organização do trabalho	17
2	TIPOS DE COMPUTAÇÃO DE BORDA	18
2.1	Computação de Borda (Edge Computing - EC)	19
2.2	Computação de Borda de Múltiplos-Acessos (Multi-Access Edge Computing - MEC)	20
2.3	Computação de Borda Veicular (Vehicular Edge Computing - (VEC)	22
2.4	Considerações Parciais	24
3	MÉTODOS PARA TOMADA DE DECISÃO NA ALOCAÇÃO DE RECURSOS	26
3.1	Métodos Tradicionais	27
3.1.1	Processo Hierárquico Analítico (<i>Analytic Hierarchy Process</i> - AHP)	27
3.2	Tipos de Métodos Meta-heurísticos	29
3.3	Métodos Meta-Heurísticos de Inteligência de Enxame	31
3.3.1	Algoritmo de Otimização da Baleia (<i>Whale Optimization Algorithm</i> - WOA)	32
3.3.2	Algoritmo de Otimização da Mariposa (<i>Moth-Flame Optimization Algorithm</i> - MFO)	34
3.3.3	Algoritmo do Morcego (<i>Bat Algorithm</i> - BAT)	37
3.3.4	Algoritmo do Vaga-lume (<i>Firefly Algorithm</i> - FA)	40
3.3.5	Busca do Cuco (<i>Cuckoo Search</i> - CS)	42
3.3.6	Otimização por Enxame de Partículas (<i>Particle Swarm Optimization</i> - PSO)	48
3.3.7	Otimização do Falcão Harris (<i>Harris Hawk Optimization</i> - HHO)	49
3.3.8	Otimizador do Lobo Cinzento (<i>Gray Wolf Optimizer</i> - GWO)	51
3.4	Considerações Parciais	56
4	ESTADO DA ARTE	57
4.1	Trabalhos Relacionados	57
4.2	Considerações Parciais	61
5	ALGORITMO METRA-HEURÍSTICO E ALOCAÇÃO DE RECURSOS EM VEC	64
5.1	Estrutura	64

5.2	Algoritmo	66
5.3	Parâmetros de Simulação	75
5.4	Resultados	80
6	CONCLUSÃO	88
6.1	Principais contribuições	89
6.2	Produção Científica	89
6.3	Trabalhos Futuros	89
	REFERÊNCIAS	90

1 INTRODUÇÃO

Como meio facilitador no dia-a-dia da população, os veículos automotores têm papel importante na rotina diária de muitas pessoas. De carro, moto, caminhão ou transporte público, como ônibus por exemplo, grande parte da população necessita deles para se locomoverem, seja para uma atividade de lazer ou para trabalhar. Com essa perspectiva, foi observado no relatório “*World Motor Vehicle Production*” de 2022 da *International Organization of Motor Vehicle Manufacturers* (OICA), que houve um crescimento de 6% na quantidade de veículos automotores produzidos no mundo no ano de 2022 em relação ao ano de 2021. Esse crescimento abrange automóveis e veículos comerciais leves em sua totalidade, porém alguns países também reportaram os demais tipos de veículos. Em números absolutos, no ano de 2022 foram produzidos cerca de 4,8 milhões de novos veículos a mais que no ano de 2021 (OICA, 2023).

Outro dado importante é sobre a emissão de CO_2 , de acordo com o relatório da “International Energy Agency” (IEA), “Global CO2 emissions by sector, 2019-2022”, divulgado em março de 2023, o setor de transportes foi responsável por emitir 22,96% de todo o CO_2 mundial, o que equivale a 7,98Gt (gigatoneladas) desse gás, ou seja, aproximadamente 0,25 Gt a mais que no ano anterior (IEA, 2023). Esse aumento no número de veículos, juntamente com uma infraestrutura inadequada para atender a alta demanda do tráfego, gera vários problemas socioeconômicos como engarrafamentos que resultam em maiores custos de combustível e aumentam as emissões de dióxido de carbono, além do estresse diário no trânsito que pode causar problemas de saúde futuros (Peixoto *et al.*, 2023).

Como paliativo a esses problemas emergentes, tem-se o uso de um Sistema de Transporte Inteligente (*Intelligent Transport System* - ITS), em que os veículos munidos de sensores, interface de comunicação e algum poder computacional trocam informações e compartilham recursos com seus pares para explorar o ambiente que transitam. Dessa forma, consegue-se prever congestionamentos a fim de evitá-los, o que minimiza o consumo de combustível, a emissão de gases poluentes, e o impacto socioeconômico e ambiental (Meneguette; Grande; Loureiro, 2018).

Para esse tipo de comunicação, os veículos podem ser organizados em uma Rede Veicular (*Vehicular Ad-Hoc Network* - VANET) para prover comunicação entre eles e com Estruturas de Beira de Estrada (*RoadSide Units* - RSU), caso elas existam (Meneguette; Madeira; Bittencourt, 2012). Porém, neste modelo, o processamento e armazenamento de toda essa informação é feito entre os próprios veículos, que detêm pouco poder computacional; ou na Nuvem com uma alta latência, por meio das RSUs. Para essa situação surgiu

um novo paradigma, o da Computação de Borda Veicular (*Vehicular Edge Computing* - VEC) que une diversas tecnologias, como *Cloud Computing*, VANETs e *Edge Computing*, para minimizar o problema de capacidade computacional e resolver o problema de latência (Raza *et al.*, 2019).

Porém, mesmo com a extensão da nuvem para as proximidades dos usuários, os recursos disponíveis não são altamente escalonáveis, o que torna necessário um bom gerenciamento deles. Para isso, otimizadores de problemas contínuos e não lineares podem ser adaptados para descobrir qual recurso será atribuído à solicitação feita (Liu *et al.*, 2021).

Nessa classificação de otimizadores, têm-se os algoritmos meta-heurísticos que tiveram sua criação inspirada em algum fenômeno natural, seja ele um fenômeno evolutivo (Algoritmo Genético), físico (Otimização por Campo Eletromagnético) ou de inteligência de enxame (Algoritmo do Vaga-lume). Esses algoritmos são classificados como de soluções múltiplas, porque iniciam com uma quantidade de soluções candidatas que serão melhoradas iterativamente, ao longo da execução. Outra característica é a sua natureza estocástica, o que os torna métodos de solução aproximada, e não exigem um poder computacional intensivo por um longo período de tempo para que uma solução aceitável seja encontrada (Yang; Karamanoglu, 2020).

1.1 Justificativa e motivação

Muita incerteza ainda paira sobre a real solução dos problemas descritos acima. Logo, diversas pesquisas são desenvolvidas para de nortear a resolução dessas questões, mesmo que a solução proposta seja parcial. Para isso, o processo de aprendizado e evolução das tecnologias devem prosseguir mesmo que morosamente e é com esse intuito que se dá a elaboração deste trabalho.

Com o crescimento do uso de tecnologias de comunicação, a preocupação com a qualidade dos serviços oferecidos também aumenta. Assim espera-se que as tecnologias atualmente implementadas consigam absorver a crescente demanda com qualidade suficiente para atender inclusive serviços sensíveis a latência. A implementação da Computação de Borda surge para auxiliar na escalabilidade e *overhead* da rede, porém deve ser acompanhada de uma boa técnica de gerenciamento de recursos para que não surjam outros problemas.

Dada as necessidades do ambiente a ser implementado, é de suma importância a escolha dos métodos e tecnologias, assim recursos poderão ser melhor aproveitados dentro do cenário de sua alocação.

1.2 Objetivos

Nesta monografia tem-se o objetivo geral de implementar o RAPTOR, com a finalidade de maximizar a quantidade de “Recursos Alocados” e “Serviços Atendidos”, enquanto minimiza a quantidade de “Serviços Não Alocados”.

Como objetivos específicos, este trabalho visa:

- Desenvolver o RAPTOR com base no método meta-heurístico da Otimização do Falcão Harris para otimizar o processo de alocação de recursos em Computação de Borda Veicular;
- Implementar diferentes algoritmos meta-heurísticos para comparar os desempenhos, eficácia e eficiência no processo de alocação de recursos;
- Realizar simulações com um simulador de rede para que sejam colhidos os parâmetros necessários para validação da técnica proposta e comparação de desempenho e resultados com os outros algoritmos;
- Maximizar a quantidade de recursos alocados e serviços atendidos;
- Minimizar as falhas no processo de escolha, ou seja, reduzir os serviços não atendidos.

1.3 Organização do trabalho

Esta monografia está estruturada da seguinte maneira:

No **Capítulo 2** são apresentados os tipos de Computação de Borda considerados, como a própria *Edge Computing*, a *Multi-Access Edge Computing* e a *Vehicular Edge Computing*.

Por meio do **Capítulo 3** são apresentados os diferentes tipos de otimizadores, com foco para os métodos meta-heurísticos de inteligência de enxame.

No **Capítulo 4** são apresentados vários trabalhos relacionados com alocação de recursos, tarefas e serviços que utilizam os métodos detalhados no capítulo 3.

No **Capítulo 5** é apresentado o cenário das simulações, o algoritmo RAPTOR e os resultados obtidos.

E por fim, por meio do **Capítulo 6** é apresentada a conclusão, produções acadêmicas e direções futuras.

2 TIPOS DE COMPUTAÇÃO DE BORDA

A Computação em Nuvem (*Cloud Computing* - CC) é um grande avanço tecnológico que revolucionou a computação e os serviços de rede (Donno; Tange; Dragoni, 2019). Porém, com o aumento significativo do número de dispositivos que utilizam essa tecnologia, e a necessidade de atender toda a demanda por serviços com baixa latência e de forma eficiente, surgiu a necessidade de aproximação entre os serviços e os usuários, sejam eles pessoas, empresas ou até mesmo dispositivos autônomos, logo, esse é o paradigma da Computação de Borda (*Edge Computing* - EC) (Li *et al.*, 2019).

Com a evolução da Computação de Borda, outros avanços vieram, um deles com o conceito de Computação de Borda Móvel (*Mobile Edge Computing*), mais tarde denominado de Computação de Borda de Múltiplos Acessos (*Multi-Access Edge Computing* - MEC). Essa tecnologia é indicada por muitos pesquisadores como uma evolução da Computação de Borda pois ela possui a vantagem de aumentar a largura de banda e minimizar a latência de comunicação e download de dados (Mach; Becvar, 2017).

Por meio do amadurecimento das tecnologias de Computação de Borda e da Computação de Borda de Múltiplos Acessos, com a agregação de outras tecnologias como Computação em Névoa (*Fog Computing*) e em Redes Veiculares Ad-Hoc (*Vehicular Ad-Hoc Networks* - VANET), um novo paradigma surgiu, a Computação de Borda Veicular (*Vehicular Edge Computing* - VEC). Nesse paradigma, o objetivo é a prestação de serviços e recursos aos veículos por meio de sua agregação (Liu *et al.*, 2021). Em outras palavras, ele visa estender as capacidades computacionais (armazenamento, processamento e rede) de um veículo com a agregação de recursos ociosos de outros para fornecer uma maior gama de serviços aos usuários.

Neste capítulo é apresentado o contexto em que se insere o trabalho, ou seja, os ambientes utilizados para o cenário de alocação de recursos. Inicialmente é definido o conceito de Computação de Borda, no qual define-se como a tecnologia funciona e seu uso para comunicação e compartilhamento de recursos. Após, é apresentado o ambiente de Computação de Borda de Múltiplos Acessos, que complementa os serviços da EC com suas próprias características e peculiaridades. Como terceiro passo é explorado o ambiente de Computação de Borda Veicular com suas características, vantagens e desafios no âmbito proposto. E por fim, são apresentadas as considerações parciais deste capítulo.

2.1 Computação de Borda (*Edge Computing* - EC)

Na busca pela solução de problemas que surgiram com a popularização da *Cloud Computing*, um conceito denominado *Edge Computing* foi elaborado para lidar com dispositivos que necessitam de recursos computacionais intensivos. Ela possui uma camada adicional na sua infraestrutura para que o EC distribua parte dos recursos computacionais antes centralizados nos *data centers*, para as bordas da rede. Assim, a capacidade computacional é fragmentada para mais próxima do usuário, o que permite uma menor latência e melhor eficiência das aplicações (Donno; Tange; Dragoni, 2019).

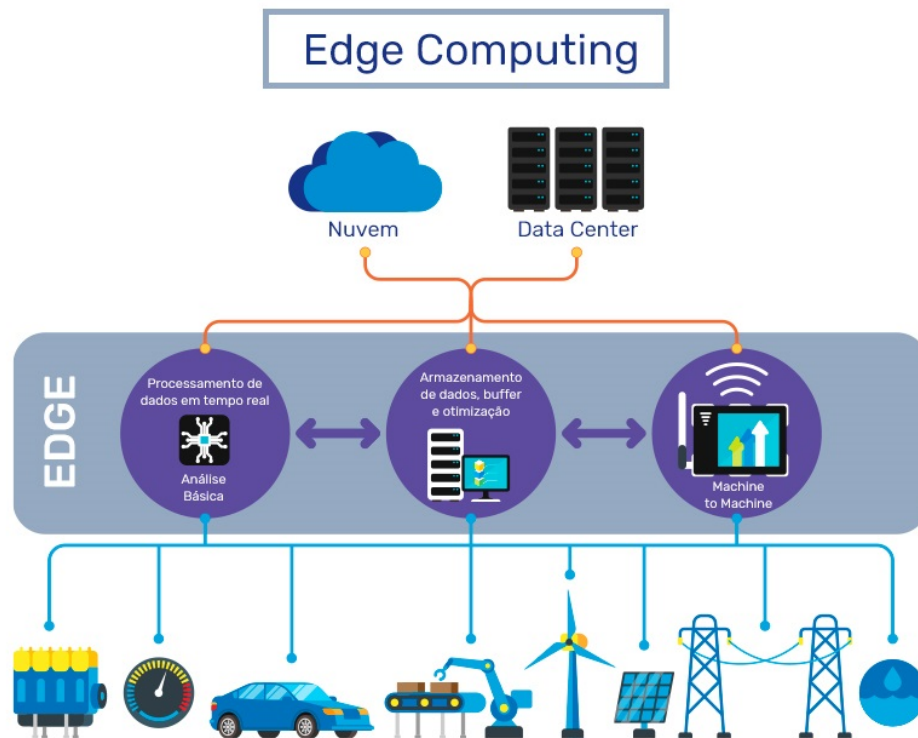
De acordo com Hassan, Yau e Wu (2019), existem cinco objetivos principais para a Computação de Borda:

- **Melhorar o gerenciamento de dados:** Necessário para administrar grandes volumes de dados que devem ser processados em tempo real e são potencialmente sensíveis a atrasos gerados pelos dispositivos de usuários ou pela rede.
- **Melhorar a qualidade do serviço:** Importante para melhorar a experiência do usuário, o que ajudará a suportar serviços sob demanda e aplicativos altamente interativos, por exemplo.
- **Prever a demanda de rede:** É necessário para estimar as quantidades de recursos que serão utilizados no atendimento à demanda da rede na proximidade local. O que proporciona uma alocação ótima de recursos e auxilia na decisão de onde a demanda será tratada, localmente na borda ou na nuvem.
- **Gerenciar e estar ciente das localizações das bordas:** Importante para oferecer suporte a serviços que dependam de localizações para seu funcionamento, como por exemplo pontos de interesse próximos. Habilitar geograficamente servidores distribuídos de borda irá induzir suas localizações e consequentemente rastrear dispositivos dos usuários para oferecer esse tipo de suporte.
- **Melhorar a gestão de recursos:** Necessário para aproveitar satisfatoriamente o uso de recursos de rede, que são limitados nas *Edges* em comparação com a nuvem em si. Dessa forma, consegue-se um melhor desempenho da rede como um todo.

Na Figura 1 ilustra-se um exemplo da localização de uma EC. A Nuvem representa a nuvem tradicional e centralizada com os *Data Centers*, a camada da *Edge* é composta pelos recursos computacionais descentralizados, localizados entre a nuvem e os usuários, e distribuídas ao longo da cidade, com vários dispositivos que os acessam e utilizam a rede.

Embora esta tecnologia traga ganhos quanto à disponibilidade e latência, ela também traz um novo desafio, que é gerenciar a alocação dos recursos disponíveis nas

Figura 1 – Conectividade da Edge Computing.



Fonte: (Cronapp, 2020).

bordas da rede (Li *et al.*, 2019). Para isso as *Edges* devem ser capazes de prover esse gerenciamento e compartilhamento de maneira eficiente para poder alcançar uma alocação de recursos globalmente eficaz.

2.2 Computação de Borda de Múltiplos-Acessos (*Multi-Access Edge Computing* - MEC)

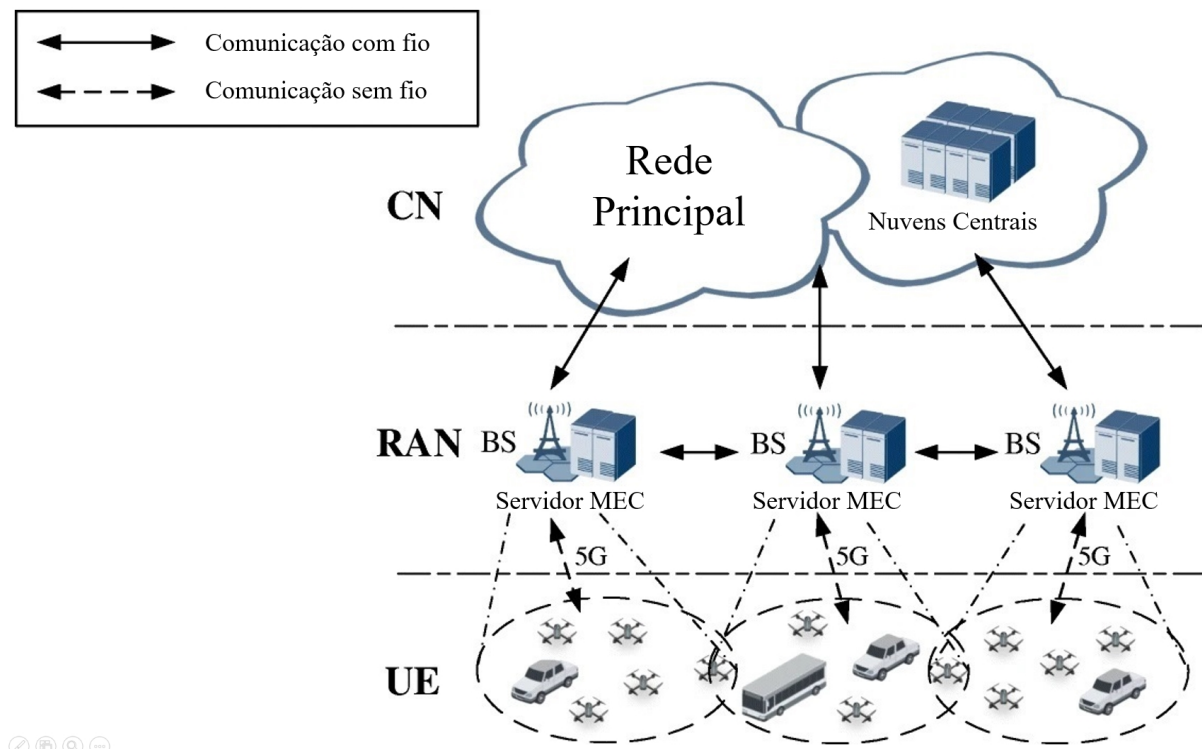
O conceito de MEC foi apresentado pela primeira vez pelo European Telecommunications Standards Institute (ETSI) em 2014 sob o nome de Mobile Edge Computing (Computação de Borda Móvel), em que uma rede de celular, por exemplo, pode adquirir os recursos de computação em nuvem ou um ambiente de serviços de tecnologia da informação (Computing; Initiative *et al.*, 2014; Hu *et al.*, 2015). Em 2017, a própria ETSI renomeou oficialmente a *Mobile Edge Computing* para *Multi-Access Edge Computing*. Segundo eles, essa mudança é para englobar os desafios de uma segunda fase de trabalho além de refletir melhor os requisitos de operadoras não celulares (Sabella; Reznik; Frazao, 2019).

O MEC foi criado para disponibilizar recursos computacionais em ambiente de borda do serviço de nuvem e dentro da Rede de Acesso Via Rádio (*Radio Access Network* - RAN). Isso permite que os dispositivos troquem informações em tempo real dentro dessa

rede (Sabella; Reznik; Frazao, 2019).

Na Figura 2 pode-se observar, de modo geral, a localização de uma MEC e seu funcionamento. No topo da imagem esta a Rede Central (*Core Network* - CN) com as *Clouds* Centrais (*Cloud Centers*) que alimentam a camada do meio, a Rede de Acesso Via Rádio. Na RAN estão contidos os servidores MEC e as Estações Base (*Base Stations* - BSs) interconectados para que possam prover o acesso da camada de Equipamentos dos Usuários (*User Equipment* - UE) ao CN e demais serviços na *Cloud*. Nesse exemplo, a comunicação é realizada por 5G, logo os BSs são estações emissoras e receptoras dessa faixa de frequência.

Figura 2 – Funcionamento de uma MEC.



Fonte: (Lin *et al.*, 2020), adaptada e traduzida pelo autor.

Lin *et al.* (2020) reforçam que os servidores MEC estão conectados às Estações Base correspondentes por meio de links dedicados e confiáveis para fornecer serviços específicos aos usuários conectados a eles. Por sua vez, os servidores MEC e os *Cloud Centers* são conectados com o *backbone* da *Internet* e compartilham informações entre si, ou seja, trabalham de forma cooperativa.

A Computação de Borda de Múltiplos Acessos é implementada com a utilização de virtualização com o uso de tecnologias como Virtualização de Funções de Rede (Network Function Virtualization - NFV), Redes de Informação Centralizada (Information-Centered Networks - ICN) e Redes Definidas por Software (Software-Defined Networks - SDN) (Mao *et al.*, 2017).

O benefício trazido com o uso da tecnologia MEC vai além da qualidade de serviço fornecido ao usuário. Ela permite que operadores disponibilizem sua RAN à terceiros, desde que autorizados, de forma rápida e dinâmica, para implementação de novos serviços e aplicações oferecidos aos utilizadores de dispositivos móveis, empresas e demais *stakeholders*. Outro ponto no cenário de dispositivos móveis, que têm capacidades limitadas de bateria, memória e processamento, é que os MECs podem receber os arquivos ao invés dos equipamentos dos usuários, assim é possível liberar os recursos computacionais de hardware desses dispositivos (Sabella; Reznik; Frazao, 2019).

O principal desafio do MEC é direcionar os dados baixados nele, e em que momento deverão ser migrados para que fiquem próximos ao usuário que os solicitou, ao passo que é necessário a manutenção da qualidade de experiência do usuário na rede (Lin *et al.*, 2020). Isso se intensifica ao considerar que os usuários se movem ao longo da rede e que pode haver troca por um MEC mais próximo.

2.3 Computação de Borda Veicular (*Vehicular Edge Computing* - (VEC))

O paradigma da Computação de Borda Veicular surgiu da necessidade de atender a serviços solicitados por veículos, com baixa latência e alta disponibilidade, por meio da agregação e comunicação veicular enquanto se locomovem (Raza *et al.*, 2019).

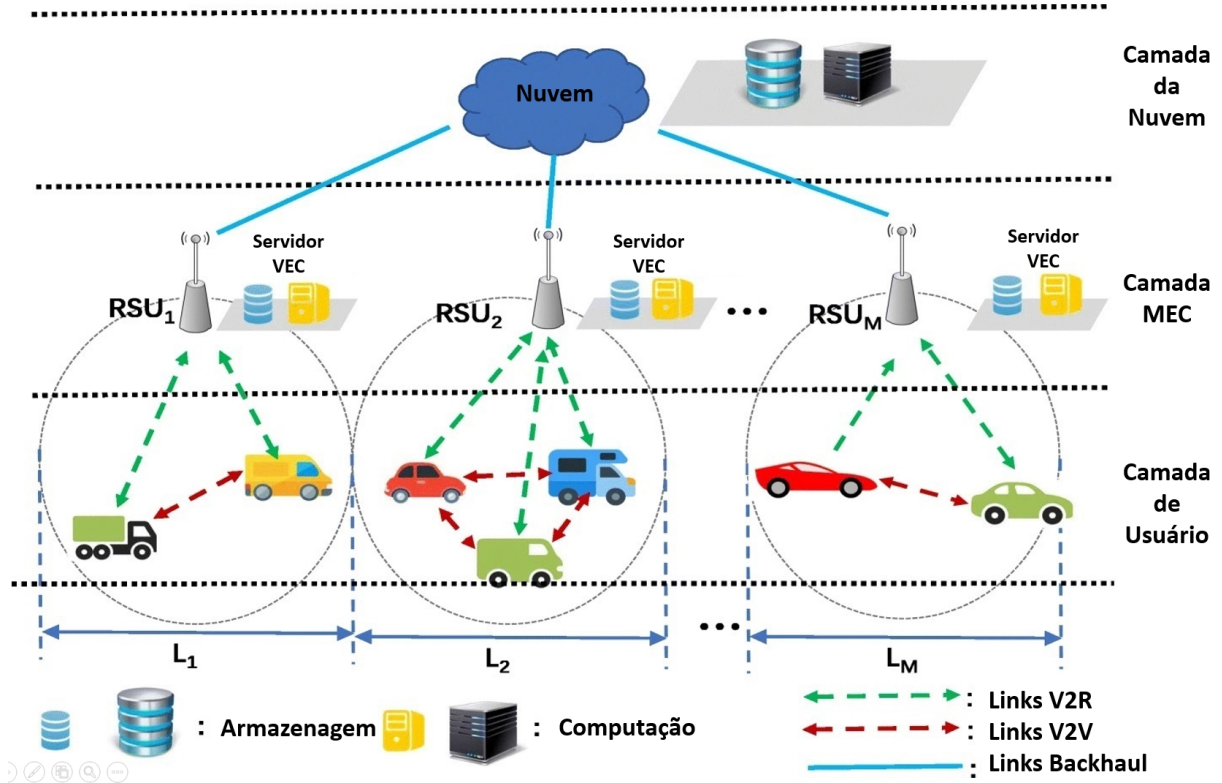
Na VEC, os veículos contam com recursos de comunicação, memória, processamento e armazenamento, para disponibilizar serviços e aplicativos aos usuários. Como os veículos têm uma capacidade limitada de recursos de computação incorporados, e que muitas vezes não atendem à carga de serviço, eles podem solicitar a um servidor, ou a um veículo próximo, que executem o serviço (Raza *et al.*, 2019).

Geralmente, as Unidades de Beiras de Estrada (*Road Side Units* - RSU) atuam como servidores autônomos (*off-grid*), pois são colocados próximos aos veículos, para reduzir o tempo de resposta e minimizar o conjunto de informações transmitidas na rede (Guidoni *et al.*, 2022). As RSUs também podem agregar, armazenar e processar dados prontamente, o que atende à qualidade dos requisitos de serviço de um aplicativo (Kang *et al.*, 2019).

Através da Figura 3 é possível observar uma arquitetura completa de *Vehicular Edge Computing*. É possível notar sua semelhança com a arquitetura de uma MEC, isso acontece por causa da mobilidade que as duas estruturas são obrigadas a oferecer aos usuários. Na imagem observa-se a camada de *Cloud* mais acima, camada MEC ao centro e a camada de usuário mais abaixo. A comunicação entre a *Cloud* e infraestrutura da camada MEC é obtida por meio do *backhaul* de *internet*, enquanto a comunicação entre

os Veículos e a RSU é por meio de alguma tecnologia sem fio, como por exemplo 5G, LTE ou IEEE 802.11p (Liu *et al.*, 2021).

Figura 3 – Arquitetura de uma VEC.



Fonte: (Liu *et al.*, 2021), traduzida pelo autor.

Também podemos notar na Figura 3 que as flechas vermelhas representam a comunicação entre dois veículos e as verdes representam a comunicação entre um veículo e uma RSU (ou infraestrutura), isso demonstra que mesmo com uma infraestrutura de apoio, a comunicação e suporte entre veículos não é eliminada, apenas organizada. Outro detalhe está nas agregações dos veículos por região geográfica (L_1 , L_2 , ..., L_M) e proximidade com a RSU correspondente (RSU_1 , RSU_2 , ..., RSU_M).

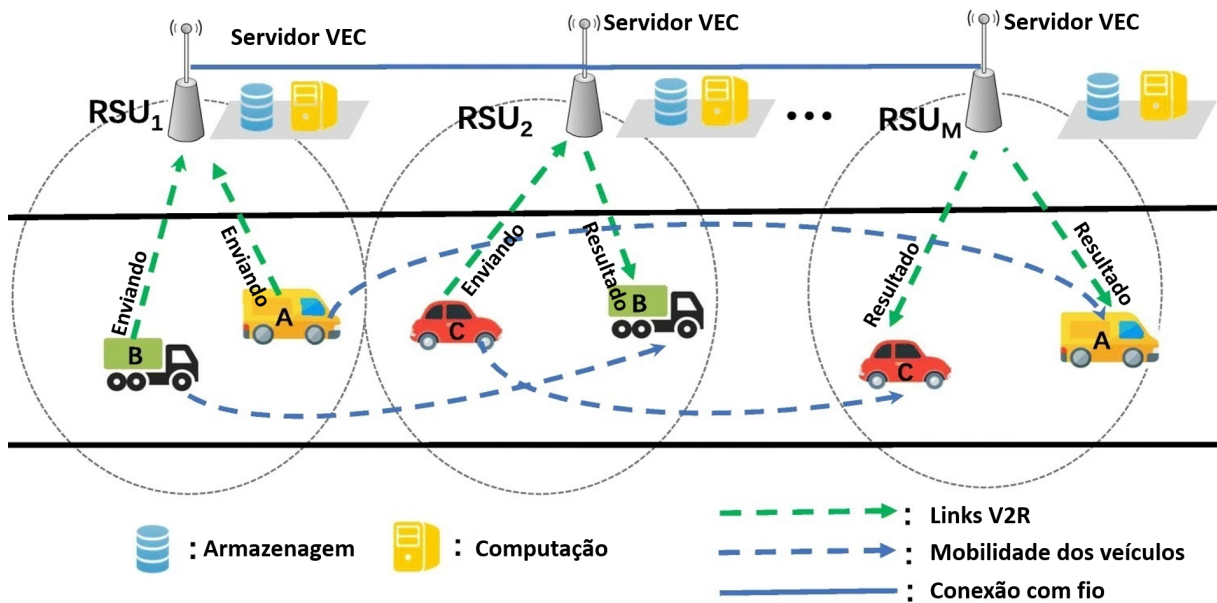
Em Liu *et al.* (2021) é reforçada a ideia de que um solicitante pode obter diretamente os recursos ou serviços necessários de nós próximos que se disponibilizam, o que nesse caso torna desnecessário o solicitante acessar o núcleo da rede. Porém, devido à quantidade limitada de recursos de cada nó, a VEC tem que lidar com três questões principais: determinar qual nó deve ser alocado, qual serviço a ser servido e qual recurso deve ser disponibilizado para atendê-lo.

Novas tecnologias emergentes são combinadas para responder a essas perguntas, como *Fog*, SDN, *Cloud* Veicular e veículos inteligentes, o que permite à VEC oferecer serviços de maneira eficiente. Um exemplo disso é a possibilidade de utilização do MEC como base para as conexões e serviços fornecidos pelos veículos, uma vez que os veículos podem transmitir seus dados e recursos do sistema para uma borda direcionada que pode

ser implementada por uma Estação Base ou RSU. Assim, o VEC pode se beneficiar do MEC, pois permite um acesso rápido aos serviços e recursos dos veículos, ou da infraestrutura rodoviária, por meio da agregação de dados e poder computacional com a formação de um *cluster* entre os veículos e RSUs (Sun *et al.*, 2019b).

Ao contrário de outras tecnologias de borda, as VECs precisam lidar com a alta mobilidade dos veículos. A mobilidade provoca uma mudança frequente na dinâmica da topologia, por causa do curto espaço de tempo para comunicação e das rápidas mudanças de posicionamento no ambiente, como pode ser observado na Figura 4. A partir dela é possível visualizar a solicitação e atendimento de três veículos em uma estrada enquanto se movimentam. Nota-se a capacidade de comunicação e previsão de movimento dos veículos entre as RSUs, para que o veículo tenha a informação solicitada no momento certo e na RSU correta, mesmo que seja uma informação sensível a latência.

Figura 4 – Exemplo de mobilidade de uma VEC.



Fonte: (Liu *et al.*, 2021), traduzida pelo autor.

2.4 Considerações Parciais

Neste capítulo foram abordados as principais arquiteturas de Computação de Borda, com o início na própria *Edge Computing*, que provê o surgimento da *Multi-Access Edge Computing* dada a necessidade de melhora na latência e processos de comunicação. Por fim, foi detalhado sobre a *Vehicular Edge Computing*, que tem sua origem por meio do amadurecimento das duas tecnologias citadas e da agregação com *Fog Computing* e VANETs, tudo para prestar serviços à veículos com alta mobilidade em uma topologia extremamente dinâmica.

No próximo capítulo, será abordado sobre os diferentes métodos para tomada de decisão na alocação de recursos, com foco nos métodos meta-heurísticos de Inteligência de Enxame, ou seja, que foram concebidos a partir da observação de determinadas espécies de animais.

3 MÉTODOS PARA TOMADA DE DECISÃO NA ALOCAÇÃO DE RECURSOS

Neste capítulo estão contidos alguns métodos utilizados para a otimização na tomada de decisão em diversos contextos, assim como para alocação de recursos. Esse processo é a ação de descoberta de uma solução ideal (também chamada de ótima solução) ou a ação de melhorar a alocação de um recurso, isso consiste em decidir qual das opções é a melhor para se destinar um recurso (Divya; Jayanthi, 2020). Neste trabalho são apresentadas algumas opções de métodos capazes de otimizar o problema de escolha da melhor requisição para atribuir os recursos disponíveis.

Nos dias de hoje, diversos problemas do dia-a-dia relacionados a tomada de decisão devem ser enfrentados, como por exemplo qual caminho seguir para se deslocar de casa até o trabalho, ou até o mercado. Se confrontarmos pessoas diferentes sobre o mesmo trajeto teremos repostas diferentes, pois cada uma irá priorizar algo em específico, ou dará pesos diferentes para as possíveis variáveis. O que priorizar? Consumo de combustível? Tempo de viagem? O caminho com melhor asfalto para maior conforto?

Com a motivação da melhor escolha, dado um determinado problema, se iniciou a busca por soluções otimizadas para resolver impasses, não só do dia-a-dia, mas principalmente problemas complexos de diferentes áreas como engenharias (Moodi *et al.*, 2018; Qu; He, 2016; Nalcaci; Ermis, 2018; Sudabattula; Kowsalya, 2016), inteligência computacional (Kumar; Pant; Ram, 2017), processamento de imagens (Raja; Vishnupriya, 2016), redes (Chahal; Harit, 2019), agendamento de tarefas (Reddy; Kumar, 2018), classificação (Soliman *et al.*, 2018), medicina (Sayed *et al.*, 2017), agricultura (Hassanien *et al.*, 2017), economia (Khalilpourazari; Pasandideh, 2017), transporte (Zhang; Gajpal; Appadoo, 2018) e até mesmo os esportes (Toffolo *et al.*, 2019).

Segundo Khan *et al.* (2016), os métodos de solução podem ser categorizados em dois diferentes grupos: soluções exatas e soluções aproximadas. Nos métodos de solução exata é realizado uma busca intensiva pela solução exata no espaço de busca, com o uso de argumentos matemáticos e técnicas estatísticas para obter essa solução. Historicamente pode-se citar Perhaps Fermat (Tikhomirov, 1991) como um dos primeiros a utilizar cálculo para problemas de otimização, e Newton e Gauss (Yamamoto, 2000) como primeiros métodos iterativos para o mesmo fim, mas outros podem ser citados como o Método de Gradiente Decendente (*Gradient Descent Method*) (Hao, 2021), Método Simplex (*Simplex Method*) (Borgwardt, 2012) e o Método de Ramificar e Limitar (*Branch and Bound*) (Lawler; Wood, 1966).

Porém, dado o aumento da complexidade dos problemas elaborados por meio de aspectos reais tão complexos quanto, observou-se a dificuldade na utilização de métodos determinísticos para solucioná-los. Com essa demanda por novas maneiras de solucionar os problemas, surgiram os métodos de soluções aproximadas, mais conhecidos como meta-heurísticos (Khan *et al.*, 2016).

De forma geral, métodos meta-heurísticos de solução aproximada são utilizados para resolver problemas de otimização, eles se valem de propriedades estocásticas para supor uma solução (problemas baseados em solução única) dentro do espaço de busca e melhora-la por meio de iterações; ou é gerado um conjunto de soluções candidatas (problemas baseados em soluções múltiplas), formadas inicialmente de modo aleatório e que iterativamente irão explorar o ambiente (exploração) e aproveitá-lo (intensificação/busca local), para também resolver problemas de otimização (Khan *et al.*, 2016).

Mesmo que as soluções propostas não sejam determinísticas, elas são testadas para garantir sua adequabilidade e a aceitação. Por sua natureza estocástica, eles não são diretamente afetados pela natureza do problema o que os torna mais fáceis de serem implementados e adaptados do que métodos determinísticos.

A organização desse capítulo se dá na seguinte estrutura: na Seção 3.1 é explicado sobre o *Analytic Hierarchy Process*, um método tradicional de otimização, na Seção 3.2 é explicada a diferença entre os tipos de métodos meta-heurísticos encontrados na literatura e na Seção 3.3 são abordados alguns métodos meta-heurísticos inspirados em inteligência de enxame, que são o foco deste trabalho.

3.1 Métodos Tradicionais

Nesta seção é descrito um método tradicional, o Processo Hierárquico Analítico (*Analytic Hierarchy Process* - AHP), que é encontrado e utilizado na literatura para solução de problemas de otimização. Este trabalho considera como Métodos Tradicionais os algoritmos que não se relacionam com o conceito de métodos meta-heurísticos.

Os métodos tradicionais são baseados em modelos estatísticos/matemáticos encontrados na literatura. Ou seja, partem de uma formulação matemática teórica para praticarem a tomada de decisão da alocação e gerenciamento de recursos por exemplo.

3.1.1 Processo Hierárquico Analítico (*Analytic Hierarchy Process* - AHP)

O AHP é um método tradicional utilizado para otimizar o processo de tomada de decisão em vários cenários, em redes podemos citar seu uso em Computação de Borda de Múltiplos-Acessos (MEC) (Pereira *et al.*, 2020) e em Nuvens Veiculares (Vehicular Clouds - VC) (Pereira *et al.*, 2019).

O AHP foi proposto por Thomas L. Saaty na década de 1970, e tem como objetivo ser um método estruturado para organizar e analisar decisões complexas. Ou seja, ele é um método multi-critério para tomada de decisão (Saaty, 1991).

Saaty (1987) recomenda utilizar até 9 níveis de critérios para a tomada de decisão. Porém, outro ponto a ser considerado, é o uso de diferentes pesos nesses critérios, ou seja, conforme a maior relevância de determinado critério, maior o peso que deve ser atribuído a ele (Golden; Wasil; Harker, 1989). Por meio da Tabela 1 é possível observar critérios de importância definidos por Saaty (1987).

Tabela 1 – Critérios de importância definidos por Saaty.

Valor de Importância	Classificação	Descrição
1	Mesma importância	Duas atividades contribuem igualmente para o objetivo
3	Importância moderada	Uma atividade é levemente mais importante que a outra.
5	Importância grande	Uma atividade possui maior importância em relação a outra..
7	Importância muito grande	Uma atividade é extremamente favorecida e essa dominância é demonstrada na prática.
9	Importância extremamente forte	É absolutamente evidente o favorecimento de uma atividade.
2, 4, 6, 8	Valores intermediários	Quando o compromisso entre as atividades é necessário.
Recíprocos		Se a atividade “x” tiver um dos números acima atribuídos a ela ao ser comparada com a atividade “y”, então “y” tem o valor recíproco ao ser comparado com “x”.

Fonte: (Saaty, 1987; Golden; Wasil; Harker, 1989), adaptado pelo autor.

Adicionalmente, Jaiswal (1997) descreve o método AHP em 7 passos:

1. O sistema é estudado em detalhes para que se identifique o objetivo, as alternativas e os critérios de comparação dessas alternativas. As informações são estruturadas hierarquicamente para facilitar a avaliação e definir os critérios e comparações necessários;
2. São coletados os dados de especialistas sobre o assunto, por meio da estrutura hierárquica. E são realizadas comparações, aos pares, em relação aos critérios do

nível anterior. Ou seja, cada critério é comparado dada a sua importância com objetivo final, e para isso é utilizado a Tabela 1.

3. Essas comparações geradas na Etapa 2 devem ser organizadas em uma matriz quadrada de tamanho correspondente à quantidade de critérios. Esse processo é realizado, também, em outros níveis da hierarquia, o que irá gerar matrizes com relação a cada critério.
4. O autovalor principal é avaliado, assim como o autovetor de cada matriz. É realizada a normalização dos elementos dos vetores, e posteriormente, é feita a organização da estrutura das matrizes norteada pela avaliação realizada dos critérios comparados.
5. A matriz tem sua taxa de consistência CI calculada por meio da fórmula $CI = (\lambda_{max} - N)/(N - 1)$, em que λ_{max} é o maior número possível de autovalor e N é a ordem da matriz. Saaty indica um Índice de Inconsistência Aleatória RI e o compara com CI . Caso $CI < 0,1$ a matriz é consistente, porém, caso $CI \geq 0,1$ então há inconsistências e é solicitado que o especialista reveja sua opinião.
6. Devido a divergências de opiniões de cada especialista, a mesma matriz é refeita por outro especialista e uma média geométrica entre as duas é calculada como forma de equilibrar esse problema.
7. Cada classificação obtida de cada alternativa é multiplicado pelo peso de cada critério, e com isso a classificação global de cada alternativa é encontrada.

Essa rotina é repetida para cada solução que se deseja otimizar, ou para cada problema que se precise resolver com o método AHP.

3.2 Tipos de Métodos Meta-heurísticos

Para os métodos meta-heurísticos, existem duas categorias: baseados em solução única e baseado em soluções múltiplas.

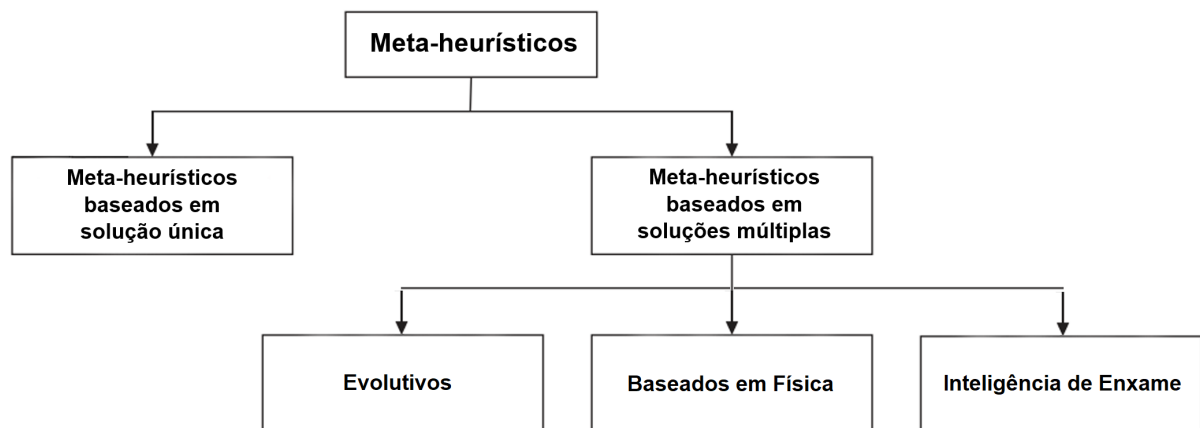
Para os métodos de solução única, o processo de busca se inicia com apenas uma solução candidata que será melhorada/otimizada até o melhor ponto possível por meio das iterações do método escolhido. Mas nem sempre isso é o suficiente para satisfazer a condição do seu uso, pois muitos problemas são multimodais e não-lineares, logo, métodos de solução não-lineares e com capacidade de tratar problemas multimodais são necessários (Gharehchopogh; Gholizadeh, 2019).

Em Gharehchopogh e Gholizadeh (2019) os autores evidenciaram algumas vantagens dos métodos meta-heurísticos de soluções múltiplas, como por exemplo o uso de uma população de soluções candidatas geradas aleatoriamente e espalhas ao longo da dimensão

considerada. Outra vantagem é a capacidade do compartilhamento de informações entre as várias soluções locais encontrados (ótimos locais), para que possam ser evitados e o ótimo global encontrado. A última vantagem destacada é a maior capacidade de exploração (diversificação) desses métodos, pois a população é inicialmente dispersa pelo ambiente e irá convergir com o passar das iterações.

Dentro da classificação de métodos meta-heurísticos de soluções múltiplas, eles são subdivididos em outras três categorias (Shehab; Khader; Al-Betar, 2017; Gharehchopogh; Gholizadeh, 2019): Evolutivos, Baseados em Física e de Inteligência de Enxame, como observado na Figura 5.

Figura 5 – Classificação dos métodos meta-heurísticos de soluções múltiplas.



Fonte: (Shehab; Khader; Al-Betar, 2017; Gharehchopogh; Gholizadeh, 2019), adaptado pelo autor.

Os métodos Evolutivos são baseados na evolução biológica, ou seja, recombinação, mutação e seleção. Em sua aplicação, por meio de um algoritmo, ele iniciará com uma população de indivíduos gerada aleatoriamente, que a cada iteração uma nova geração é criada com base nas características preferenciais de cada indivíduo da geração atual (Shehab; Khader; Al-Betar, 2017). Entre alguns exemplos temos o Algoritmo Genético (*Genetic Algorithm* - GA) que usa a teoria de evolução de Darwin (Holland *et al.*, 1992; Lü *et al.*, 2020), a Evolução Diferencial (*Differential Evolution* - DE) que provou seu valor em funções de *benchmark* conhecidas (Storn, 1995; Piotrowski, 2017) e a Otimização Baseada em Biogeografia (*Biogeography-Based Optimization* - BBO) que consistem na distribuição geográfica dos organismos biológicos (Bhattacharya; Chattopadhyay, 2010).

Os métodos Baseados em Física foram inspirados em fenômenos físicos para sua elaboração. Nele, os agentes de busca se movem ao longo do espaço por determinada regra da física (Gharehchopogh; Gholizadeh, 2019). Como exemplo, o Recozimento Simulado (*Simulated Annealing* - SA) que foi inspirado no processo metalúrgico de mesmo nome (Ingber, 1989; Wei *et al.*, 2018), o Algoritmo de Pesquisa Gravitacional (*Gravitational Search Algorithm* - GSA) que utiliza a lei da gravidade e iterações de massas em seu

processo de otimização (Rashedi; Nezamabadi-pour; Saryazdi, 2009; Rashedi; Rashedi; Nezamabadi-pour, 2018) e por fim a Otimização por Campo Eletromagnético (*Electromagnetic Field Optimization* - EFO) que utiliza o conceito do eletromagnetismo para sua aplicação (Abedinpourshotorban *et al.*, 2016).

Como ultimo grupo dos métodos meta-heurísticos de múltiplas soluções, têm-se os baseados em Inteligência de Enxame, que foram concebidos a partir da observação de algum comportamento especial de determinadas espécies de animais (Shehab; Khader; Al-Betar, 2017). Alguns exemplos são: o Algoritmo de Otimização da Baleia (*Whale Optimization Algorithm* - WOA) que reproduz a técnica de caça das baleias jubarte (Mirjalili; Lewis, 2016), o Otimização de Enxame de Partículas (*Particle Swarm Optimization* - PSO) que foi inspirado no comportamento de cardumes de peixes e bandos de pássaros (Wang; Tan; Liu, 2018), Algoritmo do Vaga-lume (*Firefly Algorithm* - FA) que usa o método de comunicação dos vaga-lumes para resolver problemas de otimização (Yang, 2010a), Algoritmo do Morcego (*Bat Algorithm* - BAT) que faz uso do sistema e ecolocalização dos micro-morcegos para guiar a busca pelo ótimo global (Yang, 2010b), entre muitos outros métodos.

Na representação desses métodos por meio de algoritmos, eles são iniciados aleatoriamente com um grupo de possíveis soluções (população) e tentam melhorar a qualidade dessa população com base em uma função objetivo. O que mudará de um algoritmo para o outro são as regras definidas para o mesmo, com base nessas regras a população será iterada e buscará uma melhor solução ao longo do espaço considerado (Gharehchogh; Gholizadeh, 2019). Outro ponto de diferença entre os algoritmos é o equilíbrio que fazem entre a exploração do ambiente (diversificação) e o aproveitamento dele (intensificação/busca local), em que cada método tem sua peculiaridade e seu foco.

3.3 Métodos Meta-Heurísticos de Inteligência de Enxame

Nesta seção são descritos alguns métodos e algoritmos de Inteligência de Enxame (*Swarm-based Intelligence* - SI). Eles são utilizados na solução de problemas complexos da vida real em várias áreas de pesquisa, como engenharias (Moodi *et al.*, 2018; Qu; He, 2016; Nalcaci; Ermis, 2018; Sudabattula; Kowsalya, 2016), agendamento de tarefas (Reddy; Kumar, 2018), agricultura (Hassanien *et al.*, 2017), etc, em que o foco é a intensificação e diversificação das possíveis soluções. Os métodos foram concebidos a partir da observação do comportamento de espécies de animais que levam o seu nome, em sua maioria por suas táticas de caça e sobrevivência (Chakraborty; Kar, 2017).

Existem várias razões para a popularidade dos métodos baseados em SI, porém as mais importantes são a flexibilidade e versatilidade oferecidas por esse tipo de método. Eles têm a capacidade de autoaprendizagem e adaptabilidade a variações externas, carac-

terísticas que atraem a atenção e interesse de pesquisadores em diversas áreas de aplicação (Chakraborty; Kar, 2017).

Ao se falar de redes veiculares é importante ter em mente que além de recursos limitados a serem distribuídos, não existe abundância de recursos computacionais de forma centralizada (Liu *et al.*, 2021). Dito isso, é importante que o método utilizado para a otimização da alocação de recursos seja eficiente para não consumir todo esse poder de processamento e eficaz para atingir o seu objetivo o mais rápido e limpo possível.

E com essa mentalidade, de eficiência e eficácia, que os métodos baseados em Inteligência de Enxame se destacam dos demais, tanto matemáticos como outros meta-heurísticos que são utilizados e aceitos pela comunidade científica (Chakraborty; Kar, 2017).

Logo, serão apresentados alguns métodos e algoritmos meta-heurísticos baseados em Inteligência de Enxame para processos de otimização, são eles: Algoritmo de Otimização da Baleia (*Whale Optimization Algorithm* - WOA), Algoritmo de Otimização da Mariposa (*Moth-Flame Optimization Algorithm* - MFO), Algoritmo do Morcego (*Bat Algorithm* - BAT), Algoritmo do Vaga-lume (*Firefly Algorithm* - FA), Busca do Cuco (*Cuckoo Search* - CS), Otimização do Falcão Harris (*Harris Hawk Optimization* - HHO), Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO), e por último o Otimizador do Lobo Cinzento (*Gray Wolf Optimizer* - GWO). É importante notar que todos eles têm como base algum comportamento daquela determinada espécie de animal.

3.3.1 Algoritmo de Otimização da Baleia (*Whale Optimization Algorithm* - WOA)

O Algoritmo de Otimização da Baleia (WOA) é um método de otimização bio-inspirado criado por Mirjalili e Lewis (2016) com base no comportamento de caça da baleia Jubarte. Esses animais são considerados os maiores mamíferos do mundo além de serem capazes de pensar, interagir, julgar, aprender e sentir emoções similarmente aos seres humanos, elas comumente são encontradas em grupos mas podem viver sozinhas (Gharehchopogh; Gholizadeh, 2019).

A dieta da baleia jubarte envolve pequenos peixes e pequenos crustáceos (*krill*) que costumam habitar a superfície do mar. Para conseguir caça-los as Jubartes utilizam uma técnica chamada “rede de bolhas” que pode ser observada por meio da Figura 6. Nela, as baleias soltam bolhas de ar pelo seu espiráculo ao nadar em volta das presas, o que forma uma espiral cônica ascendente até a superfície, para que nesse movimento as presas fiquem encurraladas e possam ser abatidas facilmente (Eler, 2019; Mirjalili; Lewis, 2016).

No algoritmo, cada solução candidata é uma baleia. Segundo Gharehchopogh e

Por meio do “Mecanismo de Encolhimento Envolvente” as presas são encontradas e são definidos todos os coeficientes de movimentação da Baleia (X , Y) e da Presa-Alvo (X^* , Y^*). As prováveis posições de movimentação das Baleias estão representadas em um espaço 2D por meio da Figura 7 (a) (Gharehchopogh; Gholizadeh, 2019).

O segundo processo, “Atualização da Posição na Espiral” pode ser observado por meio da Figura 7 (b) e consiste em calcular a distância entre a Baleia (X , Y) e a Presa-Alvo (X^* , Y^*), para poder iniciar o movimento de aproximação que se dá em espiral ao redor da Presa-Alvo (Gharehchopogh; Gholizadeh, 2019).

O método para iniciar o algoritmo, é buscar as Presas-Alvo de forma aleatória. Isso irá gerar um conjunto de possíveis soluções que serão utilizados pelo algoritmo para atualizar as posições em relação a um dos elementos aleatórios ou em relação à melhor solução encontrada até a atual iteração (Mirjalili; Lewis, 2016).

Os autores do algoritmo explicam que além de considerarem o WOA um otimizador global, eles optaram por reduzir a complexidade algorítmica com a redução de parâmetros internos na reprodução do comportamento das baleias Jubarte, e deixam como trabalhos futuros a maior fidelização ao comportamento delas assim como a possibilidade de hibridizar com esquemas de busca evolutiva (Mirjalili; Lewis, 2016).

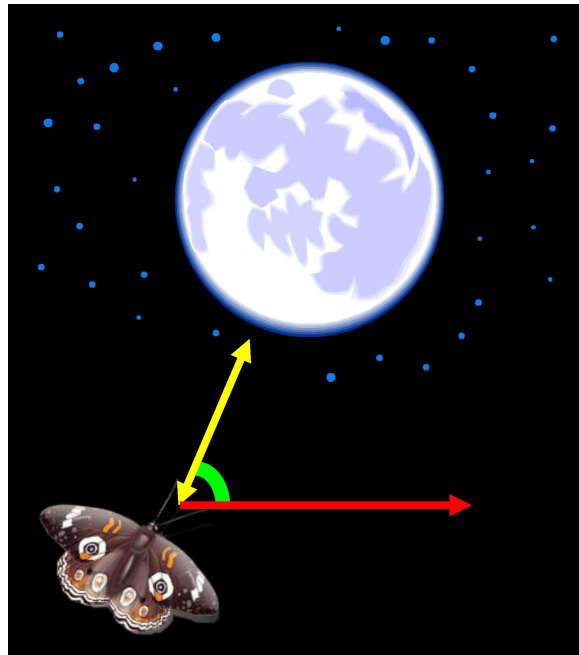
3.3.2 Algoritmo de Otimização da Mariposa (*Moth-Flame Optimization Algorithm* - MFO)

O Algoritmo de Otimização da Mariposa (MFO) é um método de otimização bio-inspirado que foi criado com base na navegação das mariposas durante a noite, sob a lua. Para se movimentarem em linha reta, as mariposas utilizam um método denominado “orientação transversal” que consiste em voar a um ângulo fixo de um ponto luminoso, que a noite será a lua. Como esse ponto de referência está muito distante, isso garante que um caminho em linha reta para o animal (Mirjalili, 2015). Na Figura 8 ilustra-se de maneira simples essa relação.

Apesar de ter uma técnica de orientação interessante, as mariposas são enganadas por luzes artificiais, como lampadas e velas. Isso associado à observação de que o ângulo de voo das mariposas em relação ao ponto luminoso normalmente é menor que 90 graus, garante um voo em espiral ao redor do ponto luminoso se ele estiver muito próximo, como pode ser observado por meio da Figura 9. Com a observação desse comportamento, com fontes luminosas próximas, o modelo matemático foi desenvolvido e o otimizador criado (Mirjalili, 2015).

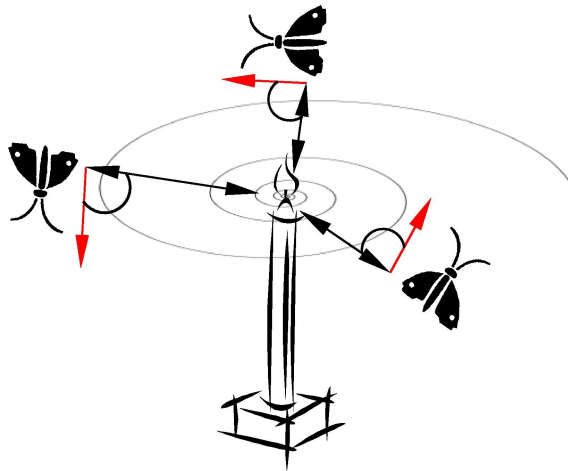
O algoritmo MFO assume que as mariposas da população são as soluções candidatas e suas posições são as variáveis do problema, em que elas podem voar até em um espaço hiper-dimensional com a mudança desses vetores de posição. Outro ponto

Figura 8 – Exemplo de Orientação Transversal.



Fonte: (Mirjalili, 2015).

Figura 9 – Voo em espiral com fonte luminosa muito próxima.



Fonte: (Mirjalili, 2015).

relevante é que o conjunto de mariposas é representado por uma matriz $n \times d$, em que n é o número de mariposas e d é o número de variáveis ou dimensões (Mirjalili, 2015).

Para todas as mariposas existe uma matriz para armazenar os valores de aptidão (*fitness*), esse valor é o retorno da função de aptidão de cada uma delas. Para esse cálculo é utilizado o vetor de posição da mariposa (Kaur; Singh; Salgotra, 2020).

Outro componente desse algoritmo são as chamas, e sua localização é armazenada em uma matriz parecida com a das mariposas. E assim como nas mariposas, as chamas também têm seu valor de aptidão calculado e armazenado (Kaur; Singh; Salgotra, 2020).

Mirjalili (2015) salienta que tanto as mariposas como as chamas são soluções em

potencial e que sua diferença está na maneira de atualiza-las a cada iteração. Enquanto as mariposas são os agentes ativos da busca e se movem ao longo do espaço considerado, as chamadas são as melhores posições encontradas pelas mariposas. Em outras palavras, as chamadas são marcadores colocados pelas mariposas enquanto elas buscam pela melhor solução, dessa forma a busca pode ocorrer livremente sem perder a posição da melhor solução.

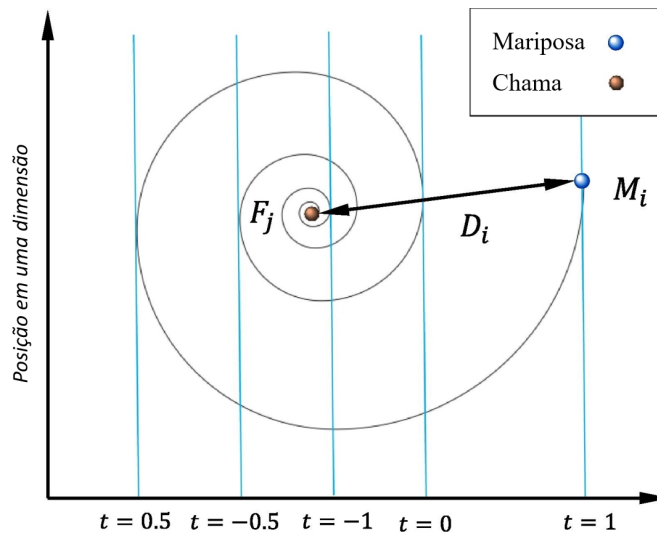
O autor do algoritmo apresenta o MFO como uma tupla, dada pela Equação 3.1. Em que I é a função que gera a população inicial de mariposas e seus vetores de aptidão, P é a função que move as mariposas pelo espaço de busca e T é o critério de parada, se falso a execução continuará, se verdadeiro a execução cessará (Mirjalili, 2015).

$$MFO = (I, P, T) \quad (3.1)$$

Para a movimentação das mariposas, Mirjalili (2015) escolheu uma espiral logarítmica para representar o voo. Mas também pode ser utilizado outro tipo de espiral desde que atenda três condições: O ponto inicial da espiral é sempre a localização da mariposa; O ponto final da espiral é sempre a chamada; e o alcance do movimento em espiral da mariposa não pode exceder o espaço de busca.

Por meio da Figura 10 pode ser observado o movimento desejado para as mariposas no modelo matemático. Em que M_i é a posição da mariposa, F_j é a posição da chamada, D_i é a distância entre esses dois pontos e t é um parâmetro na equação espiral que define o quão perto a mariposa está da chamada, em que $t = 1$ para a maior distância possível e $t = -1$ para o mais próximo possível (Mirjalili, 2015).

Figura 10 – Movimento da mariposa em espiral logarítmica.



Fonte: (Mirjalili, 2015), traduzido pelo autor.

No MFO tenta-se equilibrar os movimentos de exploração (diversificação) e apro-

veitamento (intensificação) por meio do movimento em espiral, e tenta-se também evitar a estagnação em ótimos locais ao guardar as informações das melhores soluções de cada iteração (Shehab *et al.*, 2020).

Outro ponto importante é que cada mariposa está ligada diretamente a sua chama na mesma ordem numérica, e que a cada iteração a ordem das chamadas é alterada com base no seu valor de aptidão para promover uma maior exploração do espaço considerado, ou seja, a primeira mariposa atualiza sua posição com base na melhor chama e a última mariposa atualiza sua posição com base na pior chama (Kaur; Singh; Salgotra, 2020). Isso ajuda a evitar que todas as mariposas sejam atraídas apenas pela melhor solução e que fique estagnado apenas nela.

Logo, a cada iteração, as melhores soluções são alteradas e as mariposas são obrigadas a alterar suas posições com base nas chamadas recém criadas. Um último ponto importante a ser evidenciado é a redução na quantidade de chamadas ao longo das iterações, isso auxilia com que inicialmente as mariposas possam explorar o ambiente o máximo possível e ao longo das iterações a convergência seja garantida para a melhor solução (Kaur; Singh; Salgotra, 2020).

Mesmo com o cuidado que os criadores do algoritmo tiveram, Shehab *et al.* (2020) apontam algumas fraquezas e sugerem algumas melhorias para o MFO. Eles ressaltam que apesar dos esforços para evitar ótimos locais, o algoritmo sofre recorrentemente com isso, pois é intrínseco dele que a fase de aproveitamento (intensificação) seja melhor trabalhada que a fase de exploração (diversificação), e sugerem o ajuste de parâmetros ou a hibridização com outro modelo de otimização para melhorar e resolver esse quesito, pois o modelo é relativamente simples, com boa velocidade de pesquisa e se adapta bem a hibridização com outros algoritmos.

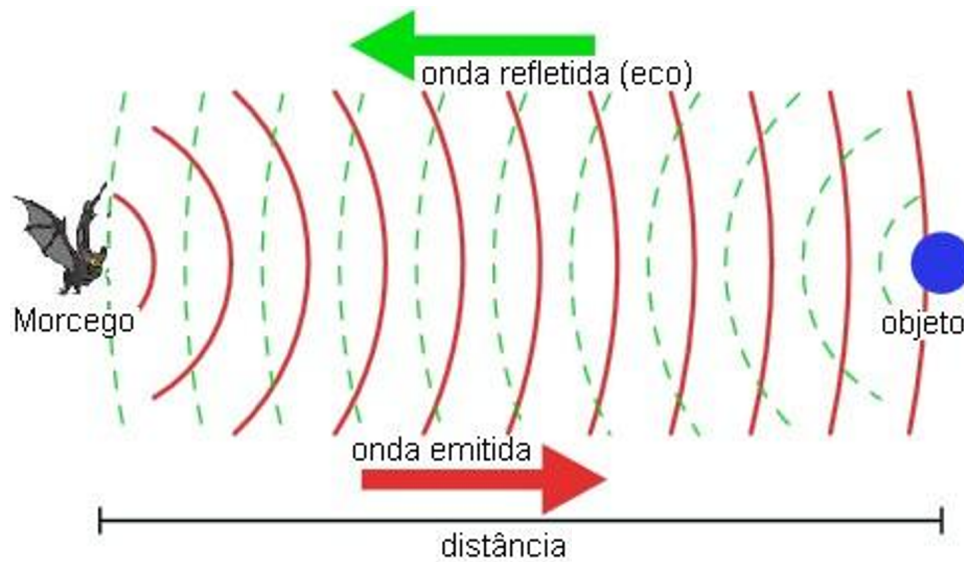
3.3.3 Algoritmo do Morcego (*Bat Algorithm* - BAT)

O Algoritmo do Morcego é inspirado no comportamento de ecolocalização dos micro-morcego (Yang, 2010b). Esses pequenos animais, em sua maioria insetívoros, fazem uso intensivo desse tipo de sonar nomeado de ecolocalização, enquanto outras espécies dependem mais de outros sentidos como olfato e visão.

O seu sistema de ecolocalização é utilizado para se movimentar, desviar de obstáculos, localizar brechas em que ele possa se abrigar e também para caçar insetos. Ele consiste na emissão de pulsos sonoros que se propagam até um objeto, presa ou superfície e ecoam de volta na direção do morcego, com isso ele consegue se localizar no ambiente e saber se existe um obstáculo, a distância e o seu tamanho (Yang, 2010b). Por meio da Figura 11, é possível observar de forma simplificada esse sistema.

Para o uso da ecolocalização durante a caça, algumas peculiaridades são observa-

Figura 11 – Ecolocalização dos morcegos.



Fonte: (Ribeiro, 2017).

das pelos criadores, como por exemplo o aumento de pulsos por segundos que são emitidos. Durante um voo normal costuma-se emitir cerca de 10 a 20 pulsos por segundo, durante a caça esse número pode chegar a 200, e quanto mais próximo da presa maior é a quantidade de pulsos. O volume dos pulsos também sofrem alteração durante a caça, pode chegar até 110dB durante a busca pela presa com redução e tendência ao silêncio conforme se aproxima dela (Yang, 2010b).

Os Movimentos de caça dos micro-morcegos foram separados em duas etapas (Yang, 2010b):

- **Busca pela presa:** Consistem em voos aleatórios para reconhecer e identificar presas em potencial. Isso se traduz no algoritmo como uma busca global;
- **Caçar a presa:** São os movimentos de ataque, em que o morcego avança em direção a presa a fim de captura-la. No algoritmo é o equivalente à busca local, em que o melhor candidato explora os arredores da sua dimensão.

Os autores também explicam quatro das simplificações realizadas a fim de tornar o algoritmo mais objetivo e eficiente, são elas (Yang, 2010b):

- Os morcegos utilizam a ecolocalização para descobrir a distância e sabem diferenciar comida de obstáculos;
- Os morcegos voam aleatoriamente e possuem cinco variáveis definidas pelos criadores: velocidade, posição, frequência de emissão de pulsos, comprimento de onda dos pulsos e volume. E eles podem ajustar o comprimento de onda e a frequência de emissão dos pulsos conforme se aproximam da presa;

- O volume poderia variar de forma aleatória ou em função do comprimento de onda, mas foi considerado que essa variável será ajustada linearmente a cada iteração, do valor máximo até seu mínimo;
- Também não foi aproveitada nenhuma informação da topografia tridimensional que poderia ser obtida, assim como o atraso entre as emissões e recepções dos pulsos sonoros.

Outras informações importantes foram observadas por Cui, Li e Zhang (2019), como por exemplo a velocidade dos morcegos que é atualizada com o passar das iterações, assim como o volume e frequência de emissão de pulsos, ou seja, de acordo com a posição da melhor solução encontrada até o momento. O comprimento de onda também varia, pois comprimentos de ondas inferiores se propagam de forma mais rápida ao custo do alcance, enquanto comprimento de ondas superiores se propagam mais lentamente e com maior alcance.

Em Cui, Li e Zhang (2019) é explicado de forma resumida e detalhada a estrutura e funcionamento do Algoritmo do Morcego:

1. A população de morcegos é inicializada e cada um deles gera sua posição inicial aleatoriamente dentro do espaço de busca proposto. Após isso, é calculado um valor de *fitness* para cada morcego, o que determinará a melhor solução naquele momento (quanto menor o valor de *fitness* melhor é a solução);
2. A pesquisa global ocorre, nela são utilizadas as variáveis de posição do morcego, sua velocidade e comprimento de onda que foram inicialmente gerados de forma aleatória por cada um deles. Outro ponto é que a velocidade da próxima geração de morcegos é influenciada pela atual melhor localização, e de forma indireta a próxima localização também sofre esse tipo de interferência;
3. A pesquisa local é a terceira etapa do algoritmo. Nesse ponto a melhor solução é aprimorada por meio de passeios aleatórios (*random walks*). Essa atualização de posições é calculada com base na posição da atual da melhor solução e no volume médio da população de morcegos, também é adicionado uma variável estocástica de distribuição uniforme para reduzir a probabilidade de estagnação em boas posições locais;
4. E por fim, são atualizadas as variáveis de volume e frequência de emissão de pulsos de cada morcego. Como explicado anteriormente, quanto mais próximo da presa, menor será o volume e maior a frequência de emissão dos pulsos. Ou seja, essas variáveis ajudam a avaliar as soluções candidatas e classifica-las, caso alguma seja melhor que a atual então ela é armazenada e a variável da melhor solução é atualizada.

As etapas 2, 3 e 4 se repetem até que o critério de parada seja atendido, seja ele um fator de convergência limite, número máximo de iterações ou até mesmo o valor calculado de *fitness*.

Algumas questões foram levantadas por Fister *et al.* (2014) e elas sugerem pontos a serem melhorados no algoritmo, como por exemplo a evolução da taxa de convergência. Segundo os autores, o algoritmo converge rapidamente no início de sua execução, porém há uma redução nesse ritmo ao longo das iterações.

Outro ponto observado é a capacidade dos métodos e algoritmos meta-heurísticos de resolverem uma alta gama de problemas diferentes sem os devidos ajustes, além da sua escalabilidade, o exemplo citado é sobre o fator de ajuste de volume e frequência de emissão de pulsos que são valores monotônicos, que podem oferecer ganho de desempenho em determinados tipos de problemas caso esses valores sejam diferentemente definidos (Fister *et al.*, 2014).

3.3.4 Algoritmo do Vaga-lume (*Firefly Algorithm* - FA)

Animais diferentes têm meios diferentes de se comunicar, e foi com esse pensamento que Yang (2010a) propôs o algoritmo do vaga-lume para solucionar problemas de otimização. Esses animais têm a capacidade de se comunicarem por emissão de *flashes* luminosos em seus corpos, isso ocorre graças a um processo químico de bioluminescência em que cada subespécie de vaga-lume tem seu próprio padrão de emissão de *flash*, assim como o ritmo deles (Yang, 2010a).

Outra característica a se destacar é o uso desses *flashes*, normalmente usados para a comunicação, tanto no processo de acasalamento como no alerta à presença de predadores (Kumar; Kumar, 2021). E é com o pensamento no processo de acasalamento que esse algoritmo foi inspirado.

Durante o período de acasalamento, os vaga-lumes machos aumentam a intensidade de seus *flashes* para atrair as fêmeas. As fêmeas por sua vez correspondem ao macho escolhido com a mesma sintonia rítmica de *flashes* ou com um ritmo próprio, o que pode variar de uma subespécie para outra (Kumar; Kumar, 2021).

E com esse pensamento de atração dos vaga-lumes que o algoritmo foi idealizado com base em três regras (Yang, 2010a):

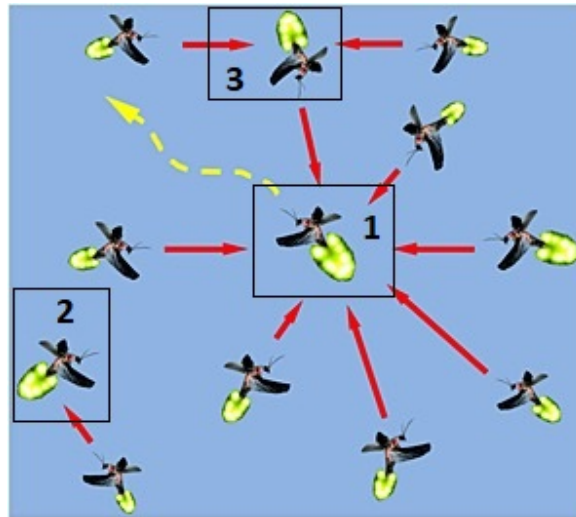
- Os vaga-lumes não têm sexo, ou seja, qualquer vaga-lume pode ser atraído por outro;
- O poder de atração vem do seu brilho, logo, para quaisquer dois vaga-lumes que possam interagir entre si, o menos brilhante irá convergir para o mais brilhante. Outro ponto, é que a percepção desse brilho por outros vaga-lumes é reduzida ao longo da distância, quanto maior a distância, menos brilhará. Caso não houver outro

vaga-lume mais brilhante do que um determinado vaga-lume, ele se movimentará aleatoriamente (passeio aleatório);

- O brilho de um vaga-lume é afetado pelo panorama de uma função objetivo.

A regra sobre a movimentação dos vaga-lumes pode ser observado na Figura 12, nela o vaga-lume “1” é considerado o mais brilhante dada a quantidade de outros vaga-lumes que convergem para ele e a sua execução de um passeio aleatório. Porém, existem outros dois vaga-lumes que são pontos de referência para seus vizinhos (“2” e “3”), ou seja, esses vizinhos os têm como melhores vaga-lumes graças a grande distância até o vaga-lume “1”.

Figura 12 – Convergência e passeio aleatório dos vaga-lumes.



Fonte: (Yang, 2010a), adaptado pelo autor.

Para essa degradação de brilho do vaga-lume, foi elaborada função com base na lei do inverso do quadrado e adicionado um vetor que controla o quanto a luz pode ser absorvida pelo ambiente (Yang, 2009). Se esse vetor tender a 0, um vaga-lume pode ser visto em qualquer lugar do espaço considerado e o algoritmo sofrerá com convergência prematura. Ao passo que se esse vetor tender ao infinito, os vaga-lumes não poderão ser vistos por nenhum outro, devida a capacidade do ambiente de absorver tanta luminosidade quanto uma névoa extremamente densa, e nesse caso, pelas regras pré-estabelecidas, os vaga-lumes todos realizarão apenas passeios aleatórios sem conversão alguma (Yang, 2010a). Isso demonstra o quão importante é o equilíbrio dessa variável para a saúde da solução.

Outro ponto a destacar é que não basta o vaga-lume brilhar, mas também os vaga-lumes ao redor enxergarem esse brilho e conseguirem definir se ele é mais intenso que o seu ou que o de um terceiro elemento. Para essa questão, foi inserido outro parâmetro que ajuda a calcular essa atratividade com base na intensidade inicial da luz emitida por um vaga-lume e a distância entre ele e o próximo (Yang, 2009).

Por meio da Figura 13, os autores trazem um fluxograma que auxilia a compreensão de funcionamento do algoritmo. Inicialmente cria-se a população inicial de vaga-lumes e é definido/calculado a função objetivo. Nesse ponto o processo de iteração é iniciado com o cálculo da atratividade e distância relativa de cada vaga-lume, que serão comparados e movidos conforme a checagem da atratividade entre eles. Após a movimentação em direção ao de maior atratividade, são calculadas novamente as soluções e cada vaga-lume tem sua intensidade luminosa atualizada. Um *ranking* é criado e atualizado ao fim desse processo para que se possa saber qual é a melhor solução candidata, e caso o critério de parada não esteja satisfeito, o algoritmo inicia mais uma iteração com o cálculo da atratividade e distância relativa entre os vaga-lumes (Kumar; Kumar, 2021).

É interessante notar que essa distância utilizada para degradar a luz ao longo dela pode ser alterada para outra forma de medida, tudo depende do problema a ser otimizado. Por exemplo, um problema de agendamento de trabalho pode ter sua unidade de medida de distância como intervalo de tempo. Outro exemplo são problemas que envolvem redes sociais, neles as distâncias podem ser definidas por agrupamentos locais e proximidade dos vértices (Yang, 2009).

Uma outra observação é válida, cada vaga-lume pode trabalhar de forma independente, o que favorece o paralelismo em sua implementação. Essa adaptabilidade é capaz de tornar o algoritmo eficiente em problemas que inicialmente ele não foi modelado para trabalhar, o mesmo ocorre para outros processos inseridos no algoritmo original como hibridização com outro método meta-heurístico, substituição da função de movimentação (ou adaptação dela) para utilizar a função de *Levy Flight* por exemplo, e também o ajuste de parâmetros existentes (Khan *et al.*, 2016).

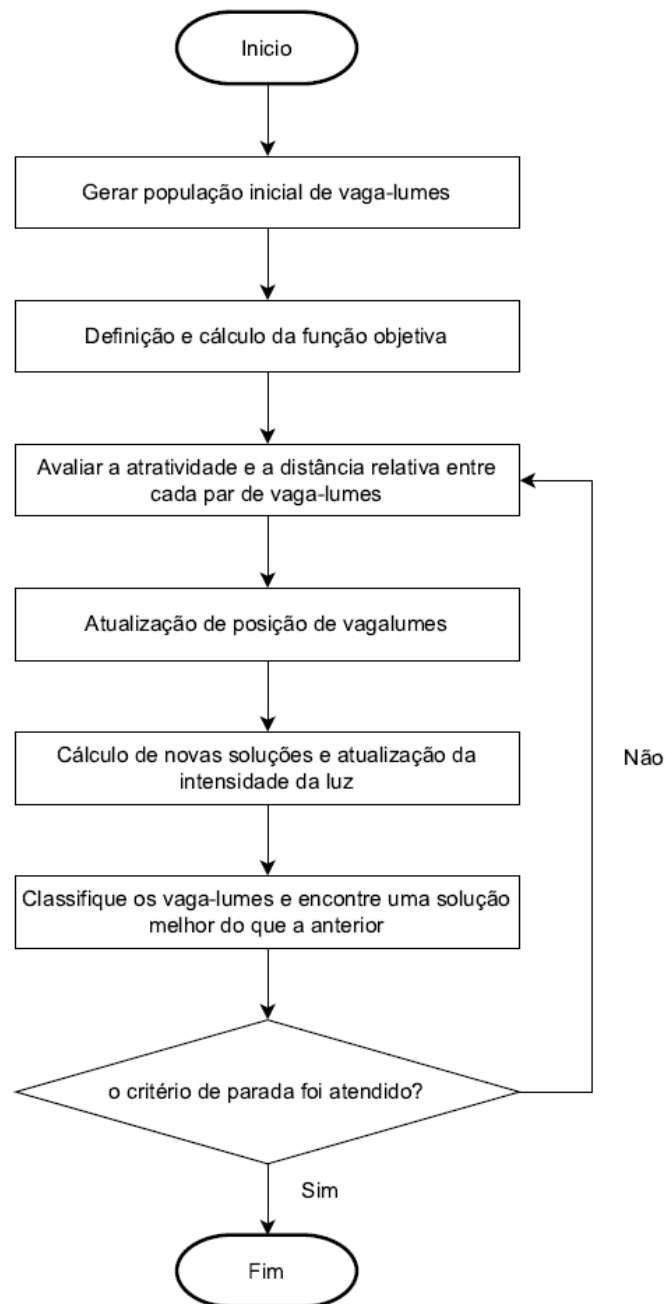
3.3.5 Busca do Cuco (*Cuckoo Search* - CS)

Idealizado por Yang e Deb (2009), esse método de otimização tem a inspiração no comportamento reprodutivo obrigatório de algumas espécies de pássaros Cuco. Esse comportamento consiste em utilizar ninhos de outras espécies de aves para colocar seus ovos.

De maneira mais detalhada, os Cucos costumam observar e acompanhar ninhos de outras espécies de aves e substituir um ovo do ninho hospedeiro por um ovo seu, é interessante observar a preferência por ninhos com ovos recém colocados. Com o seu ovo posto, a fêmea do Cuco se livra do ovo que ela está a substituir para maximizar a chance de sucesso no nascimento do seu filhote (Yang; Deb, 2009).

Curiosamente, os ovos de Cuco tendem a nascer primeiro que ovos de outras espécies. Associado ao comportamento instintivo do filhote de Cuco em expulsar do ninho os demais ovos, é garantido que ele receberá toda a alimentação para crescer (Yang;

Figura 13 – Fluxograma do FA.



Fonte: (Kumar; Kumar, 2021), traduzido pelo autor.

Deb, 2009). E mesmo que ele nasça depois dos ovos da ave hospedeira, o seu tamanho avantajado (até três vezes maior que os filhotes da ave que o abriga) e sua capacidade de imitar o som dos filhotes dessa outra espécie, irão ajudar a garantir que toda comida do ninho venha para ele, o que matará os demais filhotes de fome em poucos dias (Rajabioun, 2011).

Obviamente que se o pássaro hospedeiro descobrir que um dos ovos não é seu, ele o jogará para fora do ninho ou abandonará o ninho para montar outro em um lugar mais seguro. Para minimizar esse risco, algumas subespécies de Cuco são capazes de

colocar ovos na mesma cor e forma dos ovos do ninho a ser parasitado o que revela uma capacidade de adaptação incrível dessa espécie (Yang; Deb, 2009).

Para o modelo matemático criado, foram idealizadas três regras a serem seguidas (Yang; Deb, 2009):

- Cada Cuco escolhe um ninho de forma aleatória e coloca um ovo por vez;
- Os melhores ninhos com os melhores ovos serão a próxima geração de Cucos;
- O número de ninhos hospedeiros é fixo, e a probabilidade de descoberta do ovo intruso é uma fração do total a ser definida.

Por definição, cada ovo em um ninho é uma solução, cada ovo de Cuco representa uma nova solução em que o objetivo é utilizar essas novas soluções (e potencialmente melhores) para substituir as soluções não tão boas assim, esse cálculo e *ranking* é feito por meio do valor de aptidão (*fitness*) (Yang; Deb, 2009).

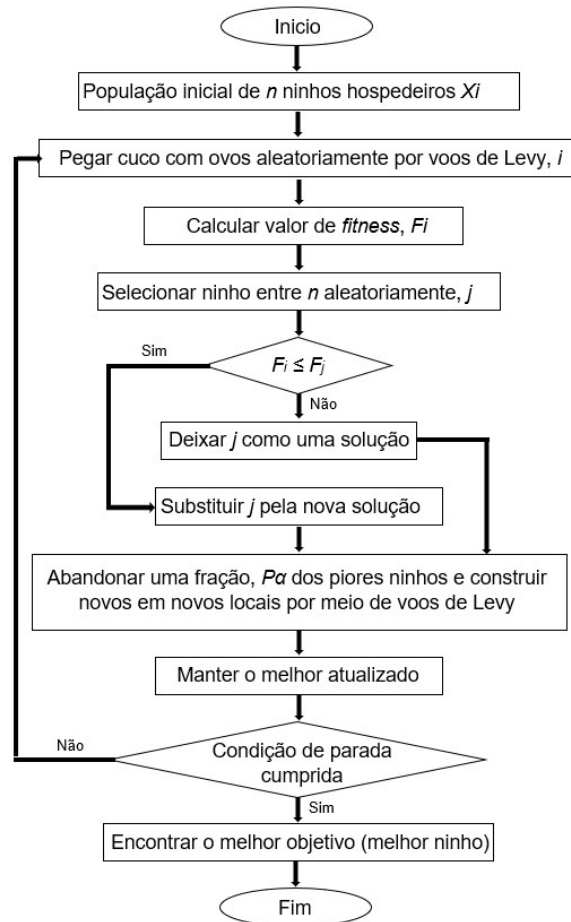
Por meio da Figura 14, é possível observar o passo-a-passo do algoritmo CS. Ele é inicializado com a criação de n ninhos hospedeiros, em que X_i é um ninho selecionado. O próximo passo é pegar aleatoriamente pela função de *Levy Flight* um Cuco que tenha ovos e calcular o seu valor de *fitness* (F_i) para então selecionar um ninho aleatoriamente (ninho j escolhido como exemplo). Nesse ponto, a primeira checagem acontece, é comparado os valores de *fitness* de i e j , caso o valor de F_i for menor ou igual à F_j , j será substituído por essa nova solução, caso contrário a solução j será mantida (Yang; Deb, 2009).

Após essa decisão de qual é a melhor solução candidata, uma fração dos piores ninhos serão destruídos e novos serão criados em novas localizações por meio da função de *Levy Flight* (LF). Uma atualização ocorre para saber qual melhor ninho. Após essa segunda atualização, uma nova checagem é feita, caso o critério de parada tenha sido atendido, o algoritmo será finalizado e o objetivo terá sido o melhor ninho. Caso o critério não seja atendido, os Cucos realizarão outro LF e colocarão novos ovos, esse *loop* é repetido sucessivamente até que o critério de parada seja atendido (Yang; Deb, 2009).

Em Shehab, Khader e Al-Betar (2017) é comparado esse algoritmo com outro de mesma inspiração, chamado Algoritmo de Otimização do Cuco (*Cuckoo Optimization Algorithm* - COA) que foi criado em 2011 por Rajabioun (2011) de maneira menos simplista mas com algumas semelhanças.

Em Rajabioun (2011), os autores apresentaram a estrutura básica do COA em forma de fluxograma, como pode ser observado por meio da Figura 15. Nela, observa-se que o algoritmo iniciará os Cucos com seus ovos, posteriormente os ovos serão colocados em diferentes ninhos, para posteriormente terem uma porcentagem deles detectados e mortos. Uma checagem é feita para saber se existe a quantidade de Cucos igual ao número máximo

Figura 14 – Fluxograma do Algoritmo CS.



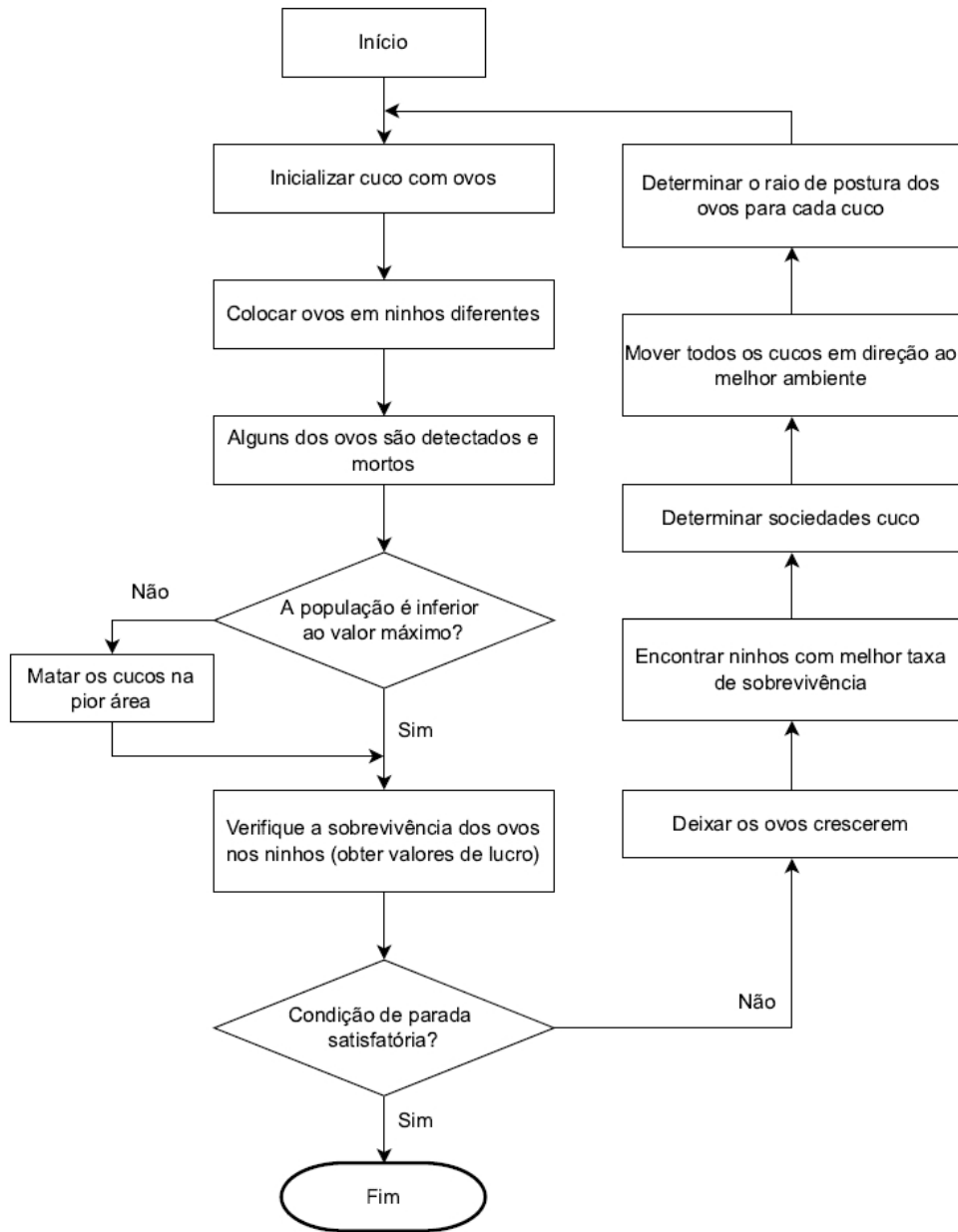
Fonte: (Shehab; Khader; Al-Betar, 2017), traduzido pelo autor.

estabelecido. Caso afirmativo, alguns Cucos serão eliminados e o algoritmo avança para a próxima fase. Caso não, o algoritmo avançará sem alterações. O próximo passo é calcular o valor de lucro dos ovos sobreviventes e checar se a condição de parada do algoritmo foi atendida; caso sim, o algoritmo é finalizado; caso não, alguns passos adicionais são realizados antes de começar uma nova iteração.

Após a checagem e a definição por continuar a execução do algoritmo, os ovos irão eclodir e se tornarão Cucos adultos. Então o algoritmo irá ranquear os melhores ninhos com base no valor de lucro e determinará as sociedades de Cucos com base na sua proximidade. Em grupos, os Cucos voarão em direção a melhor solução encontrada, porém estocasticamente com o uso de uma função própria para favorecer a exploração do ambiente, isso pode ser observado por meio da Figura 16. Com a posição do Cuco definida, o algoritmo calcula o raio de postura de ovos de cada Cuco em suas micro regiões para fomentar o processo de aproveitamento/intensificação. E assim o processo de postura de ovos se inicia novamente com o fechamento desse ciclo (Rajabioun, 2011).

De maneira mais detalhada, nessa segunda abordagem as variáveis do problema proposto são formadas por uma matriz de $1 \times N$, em que cada valor na matriz repre-

Figura 15 – Fluxograma do Algoritmo de Otimização do Cuco.

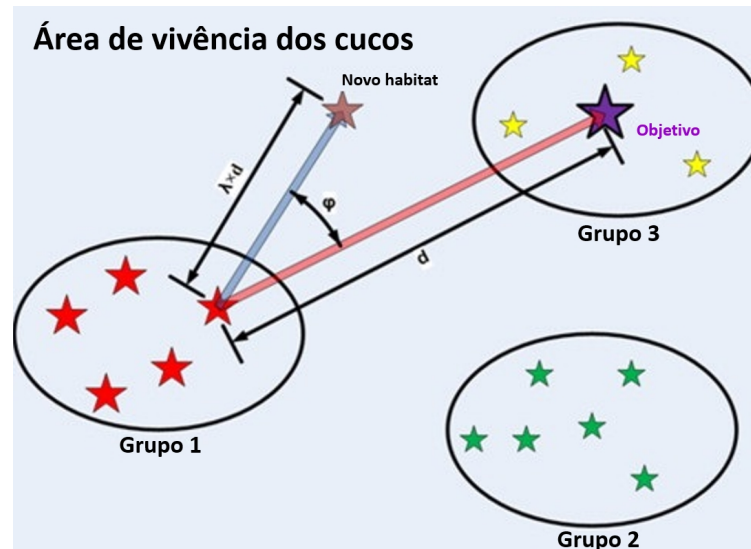


Fonte: (Rajabioun, 2011), traduzido pelo autor.

senta a posição atual de vida de um Cuco, essa matriz é chamada de *habitat*. Nota-se também que cada variável dessa é um número de ponto flutuante e uma função lucro é utilizada para calcular o valor de aptidão desse *habitat*. Os autores também explicam que caso seja necessário utilizar essa função como minimizadora (função custo) ao invés de maximizadora, é possível inverter o operador da função (Rajabioun, 2011).

A inicialização do algoritmo se dá com a criação de uma matriz de *habitat* candidata de tamanho $N_{pop} \times N_{var}$. Logo, um número aleatório de ovos é atribuído para cada um dos *habitats* e o raio de postura de ovos (*Egg Laying Radius* - ELR) é calculado com base no total de ovos, quantidade de ovos do Cuco atual e limites máximo e mínimo que um Cuco pode carregar (Rajabioun, 2011).

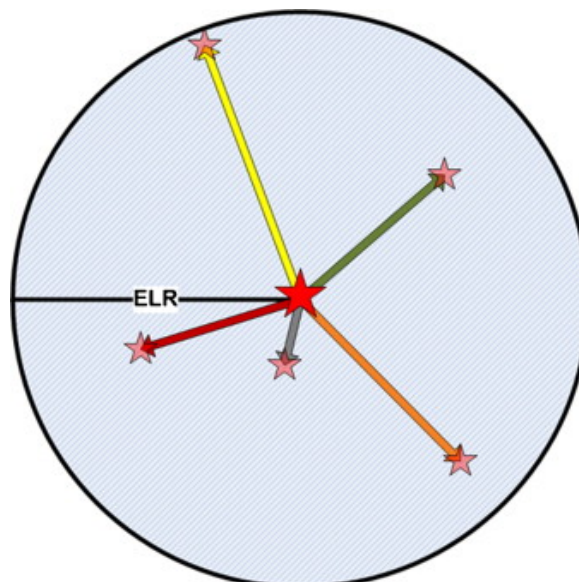
Figura 16 – Exemplo de imigração de um grupo de Cucos.



Fonte: (Rajabioun, 2011), traduzido pelo autor.

Com o raio de postura calculado, os Cucos fazem passeios aleatórios (*random walks*) para encontrar ninhos e colocarem seus ovos, a partir da Figura 17 é possível observar essa fase. Após a postura dos ovos, a porcentagem com menor valor de lucro (os autores definiram em 10%) são eliminados, o restante deles se tornarão Cucos adultos (Rajabioun, 2011).

Figura 17 – Raio de postura de ovos de um Cuco.



Fonte: (Rajabioun, 2011).

Na fase adulta e com sociedade formada, os Cucos migram de *habitat* com destino à localização do grupo de Cucos com maior valor de lucro. Com essa convergência, os autores optaram por utilizar um agrupamento por K-means (*K-means Clustering*) para diferenciar esses grupos (Rajabioun, 2011). E essa migração é feita por meio de uma

equação própria que considera a localização do melhor grupo e altera essa rota com a mudança do angulo colocado na sua origem, como observado por meio da Figura 16, isso auxilia na exploração de todo o espaço considerado (diversificação).

Ao migrarem e começarem a morar nos novos *habitats*, cada Cuco recebe uma quantidade de ovos, o valor de ELR é calculado para cada Cuco e um novo processo de postura de ovos é iniciado (Rajabioun, 2011). É importante salientar que é definido no algoritmo uma quantidade máxima de Cucos, caso essa quantidade extrapole, sobreviverão apenas aqueles com maior valor de lucro.

Por fim, tem-se a definição do critério de parada, normalmente número máximo de iterações, mas também pode ser definido com fator de convergência, os autores definiram como 95% de toda a população de Cucos para o mesmo *habitat* como um possível critério de parada (Rajabioun, 2011).

As diferenças entre os algoritmos é evidente, em Shehab, Khader e Al-Betar (2017) é explicada algumas dessas diferenças, como por exemplo a preferência em se montar uma matriz para as localizações no COA e outra para as variáveis do problema, assim como o uso de uma função lucro para ranquear as melhores soluções ou invés do uso de valores de *fitness*.

Outro ponto evidenciado por Shehab, Khader e Al-Betar (2017) é que no COA os Cucos podem colocar mais de um ovo, não ao mesmo tempo e nem no mesmo ninho, mas o cálculo do ELR e o passeio aleatório é justamente para que ele possa depositar todos os seus ovos dentro dos limites dessa micro região. Outra alteração feita no COA é o agrupamento dos Cucos de regiões próximas e a sua movimentação em grupo, isso demonstra o foco dos autores com o contexto da otimização do Cuco.

Em Shehab, Khader e Al-Betar (2017) é evidenciado a qualidade do CS, como por exemplo sua baixa complexidade de implementação devido à poucos parâmetros a serem configurados, bom equilíbrio entre os processos de exploração e aproveitamento, e ser utilizado em várias áreas e em diferentes problemas na literatura. Porém ressaltaram que carece de alguma atenção no ajuste desses poucos parâmetros e também necessita de uma análise matemática mais profunda para que esses parâmetros e possíveis adaptações sejam feitas de maneira precisa ao invés do “tentativa e erro”.

3.3.6 Otimização por Enxame de Partículas (*Particle Swarm Optmization - PSO*)

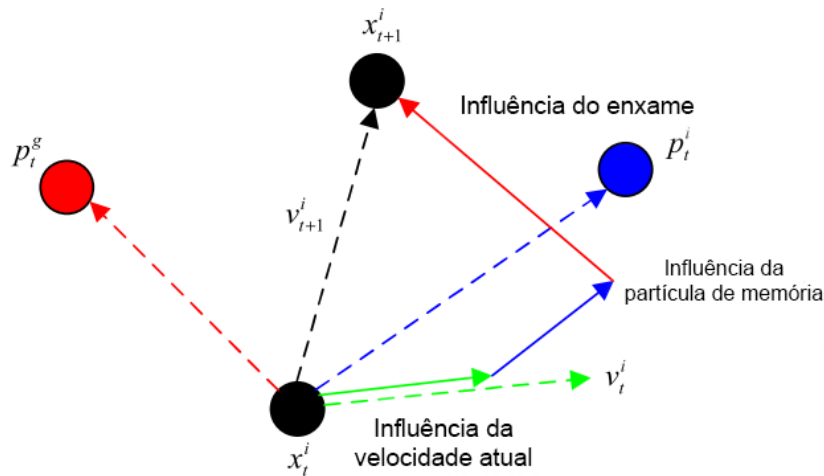
A Otimização por Enxame de Partículas é um método de otimização meta-heurístico de inteligência de enxame criado por Eberhart e Kennedy (1995) com base na observação do comportamento de cardume de peixes e bandos de aves na busca pelo alimento. Essa otimização leva em conta a contínua iteração entre as partículas na busca

pela solução ideal. Cada membro é considerado uma partícula que representa uma solução candidata ao resultado do processo de otimização (Wang; Tan; Liu, 2018).

Esse método não busca diretamente o procedimento de cálculo para atingir seu objetivo, mas sim as informações dos ótimos membros do enxame e dos ótimos membros locais, obtidos durante as iterações do algoritmo (Wang; Tan; Liu, 2018).

O algoritmo PSO possui dois vetores de informações, um vetor local e um global. Na vetor local, o algoritmo armazena as informações das posições ótimas das próprias partículas, chamada de $pbest$, e as informações das posições ótimas de suas vizinhas, chamada de $nbest$. No vetor global, o algoritmo também armazena as posições $pbest$, porém ele irá armazenar a melhor posição do enxame, nomeada de $gbest$ (Wang; Tan; Liu, 2018). Na figura 18 ilustra-se a iteração de uma partícula durante essa execução.

Figura 18 – Iteração do esquemas de partículas



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

Ao analisar a figura 18, é possível observar que a primeira parte considerada é a influência da velocidade atual, que além de utilizar sua velocidade, utiliza o seu movimento em inércia. Na segunda fase, chamada de “fator de aceleração cognitiva”, o movimento é realizado a partir de sua posição atual e seu $pbest$, ou seja, sua melhor posição. Na terceira parte, chamada de “fator de aprendizagem social”, é considerada sua posição atual e o $gbest$, ou seja, a melhor posição do enxame (Wang; Tan; Liu, 2018). De forma resumida, uma partícula se movimentará com base na sua experiência e na experiência de outras partículas do enxame.

3.3.7 Otimização do Falcão Harris (*Harris Hawk Optimization* - HHO)

O método HHO foi desenvolvido com base no comportamento cooperativo de caça à presas fugitivas (normalmente coelhos) dos falcões Harris. Em sua concepção, Heidari

et al. (2019) modelaram matematicamente o comportamento dessas aves para a busca de melhores soluções em diversos problemas de otimização. Como todo método bio-inspirado citado neste trabalho, a natureza estocástica dessa estratégia de caça é traduzida também para o algoritmo.

Esse tipo de busca por alimentos é conhecido apenas em algumas espécies de mamíferos carnívoros. Aves de rapina normalmente caçam sozinhas, diferente dos falcões Harris que vivem em grupos, cooperam entre si para a caça além de compartilharem o alimento (Heidari *et al.*, 2019).

De forma resumida, o forrageamento dessas aves consiste primeiramente em observar o ambiente de caça empoleirados em árvores gigantes ou outros pontos altos que possam existir. Ocasionalmente eles fazem pequenos voos de reconhecimento pelo ambiente como uma busca ativa pela presa desejada, além de se reposicionarem. Ao encontrar uma presa, esses falcões iniciam seu movimento de cerco à presa seguido do movimento de ataque, também conhecido como “ataque surpresa” (Heidari *et al.*, 2019).

O “ataque surpresa” consiste na coordenação dos falcões do grupo para realizar investidas de várias direções e tentar surpreender a presa para que possam finalizá-la. Normalmente essa estratégia é precedida de uma perseguição que pode demorar vários minutos, pois a presa inicia seus movimentos de fuga assim que se sente ameaçada. Para os falcões, o objetivo dessa perseguição é cercar a presa e levá-la a exaustão, o que facilita o abate (Heidari *et al.*, 2019).

O que pode ocorrer também durante a perseguição é uma troca de liderança na coordenação desses ataques. Isso ocorre após um movimento errado do líder do grupo que se descoordena e é rapidamente substituído por outro, isso também ajuda a confundir a presa que está em fuga. Todo esse cerco é mantido até que a presa seja abatida ou consiga alguma cobertura/fugir (Heidari *et al.*, 2019).

O algoritmo HHO têm 2 fases em sua composição: fase de exploração (diversificação) e a fase de ataque (aproveitamento).

Na primeira fase, a de exploração, o algoritmo reproduz o poder de observação dos falcões Harris, aquele momento em que estão no topo das árvores e observam o ambiente a procura de uma presa. Para o algoritmo, cada falcão é um membro da população e uma possível solução, que é inicialmente gerada de forma aleatória, e a cada rodada a melhor possível solução é considerada a presa pretendida (Heidari *et al.*, 2019). Existem dois tipos de exploração nessa fase, explorar o espaço aleatoriamente ou explorar com base na posição média da população. O que irá definir o tipo de exploração de cada falcão é o fator de conversão q que tem seu valor gerado aleatoriamente e igualmente distribuído entre 0 e 1 (Qu *et al.*, 2020).

Para a melhor solução candidata, também chamada de presa, é calculado o valor

de E . Esse valor corresponde a energia que a presa tem para fugir do anel de cerco formado pelos falcões, ela é decrescida a cada iteração e seus valores vão de 2 (totalmente fortalecida) à 0 (totalmente exaurida). Essa energia E tem uma tendência decrescente e os falcões tomarão diferentes atitudes com base nela, em outras palavras, quanto menos energia a presa tiver, menor é a sua possibilidade de se esquivar do cerco e consequentemente maior a capacidade de ser capturada (Song *et al.*, 2021). Caso $|E| \geq 1$ os falcões procurarão diferentes áreas para explorar, ou seja, irão mudar de árvore para observar um novo local (diversificação). Se $|E| < 1$ o algoritmo irá transitar para a fase de cerco, em outras palavras, os falcões irão intensificar a busca, cerco à presa e ataca-la (Heidari *et al.*, 2019).

Para a fase de ataque (aproveitamento) os falcões cercam a presa com a formação de um anel de cerco ao redor da mesma e realizam o “ataque surpresa” para tentar surpreender e abater a presa (Song *et al.*, 2021). Como as presas geralmente são coelhos, que são ágeis e expertos, isso normalmente acaba em perseguição durante uma situação real.

Para modelar de uma forma mais fidedigna, os criadores do HHO definiram 4 estratégias de ataque contra a presa, baseando-se em duas variáveis: o valor aleatório r (oscila entre 0 e 1) que traduz a possibilidade de uma presa escapar e o valor absoluto da energia $|E|$ que foi explicado acima (Heidari *et al.*, 2019).

Caso $|E| \geq 0,5$ um cerco leve irá ocorrer, caso esse valor seja $|E| < 0,5$ então ocorrerá um cerco pesado. Para a variável r , caso ela seja $r \geq 0,5$ apenas o cerco ocorrerá e nenhuma manobra adicional será realizada, porém se $r < 0,5$ além do tipo de cerco definido por $|E|$ os falcões realizarão mergulhos rápidos e progressivos durante o cerco (Qu *et al.*, 2020). Em outras palavras $r \geq 0,5$ indica que a presa não tem capacidade de fugir do ataque dos falcões, logo apenas a estratégia de cerco para exaurir a energia da presa será necessária. Mas caso a presa tenha capacidade de fugir dos falcões ($r < 0,5$) um movimento a mais é necessário para que a caça tenha sucesso.

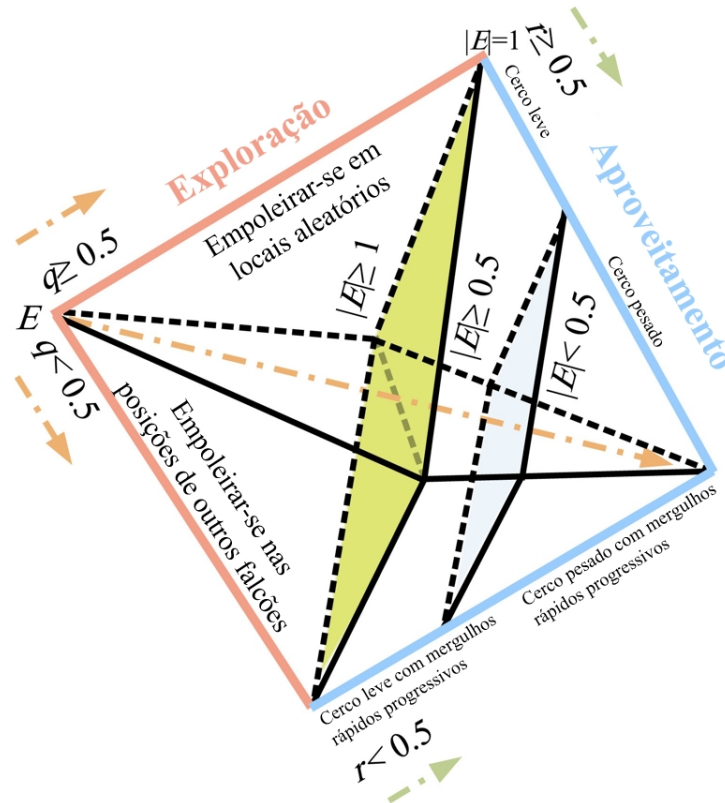
Através da Figura 19, é possível observar todas as fases da caça, tanto na exploração como no aproveitamento, assim como as quatro estratégias de ataque que serão detalhadas no Seção 5.

3.3.8 Otimizador do Lobo Cinzento (*Gray Wolf Optimizer* - GWO)

O método GWO é inspirado no comportamento dos lobos cinzentos, tanto na sua hierarquia social quanto na sua estratégia de caça. Criado por Mirjalili, Mirjalili e Lewis (2014), esse método tem por objetivo otimizar a busca por ótimas soluções globais sem ficar preso em boas soluções locais.

Os lobos cinzentos são predadores que caçam em bando e estão no topo da sua

Figura 19 – Descrição das fases de caça do algoritmo HHO.



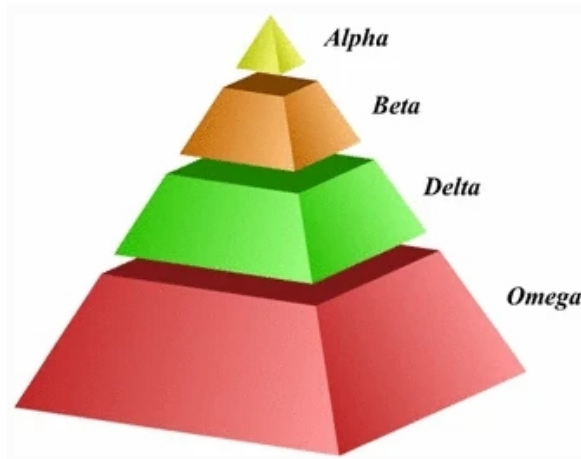
Fonte: (Heidari *et al.*, 2019), traduzido pelo autor.

cadeia alimentar, uma matilha costuma ter de 5 à 12 lobos que são divididos hierarquicamente em quatro níveis (Kumar; Pant; Ram, 2017), como pode ser observado por meio da Figura 20:

- **Alpha** (α): Está no topo da hierarquia, ele é o responsável por decisões coletivas da matilha como a caça, onde dormirão, que horas retomarão as atividades, e assim por diante. Todos os comandos do *alpha* devem ser obedecidos por toda a matilha;
- **Beta** (β): Hierarquicamente abaixo do *alpha* e são responsáveis por auxiliá-lo em tomadas de decisões e reforçar os seus comandos aos demais subordinados (*Deltas* e *Omeegas*). Também são os mais aptos a subirem para o posto de *alpha* em caso de morte ou envelhecimento do mesmo;
- **Delta** (δ): São subordinados ao *alpha* e aos *betas* mas ainda estão acima dos *omegas* na hierarquia. Esses lobos têm a função de suporte à matilha e desempenham os papéis de batedores (vigiam os limites do território), sentinelas (fazem a segurança da matilha), caçadores (auxiliam na caça de presas), anciãos (auxiliam o bando de forma geral, pois são antigos *alphas* e *betas*), e por fim os zeladores que são responsáveis por cuidar dos lobos feridos, doentes e fracos;

- **Omega** (ω): Estão na base da hierarquia da matilha, tem o papel de “bode expiatório” e se submetem a todos os outros lobos de maior hierarquia. Algumas vezes eles desempenham o papel de babás dos mais novos, e apesar de serem os últimos a se alimentarem e aparentarem não terem importância, foi observado que matilhas que sofrem perdas de *omegas* passam por uma fase de problemas e conflitos internos.

Figura 20 – Hierarquia social dos Lobos Cinzentos.



Fonte: (Faris *et al.*, 2018).

Além de reproduzir a hierarquia de uma alcateia de lobos cinzentos, o GWO também utiliza a técnica de caça desses animais na otimização da possível solução para o problema proposto. Ela consiste em três etapas (Mirjalili; Mirjalili; Lewis, 2014):

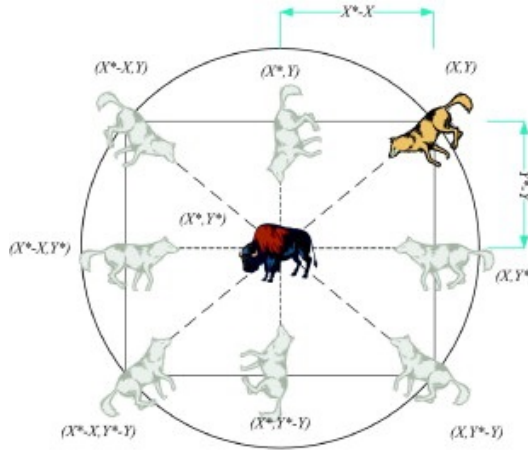
1. **Rastrear e Perseguir:** Procuram pela presa e a perseguem;
2. **Cercar e Assediar:** Cercam a presa e a avançam até que ela cesse os movimentos;
3. **Atacar:** Após a presa estar imóvel o grupo de caça atacam para abatê-la;

A reprodução matemática de todo esse comportamento é no que consiste o algoritmo GWO, o lobo α é a possível ótima solução global, seguido da segunda melhor possível solução como β , e δ como terceira melhor possível solução. As soluções candidatas restante são consideradas as ω (Mirjalili; Mirjalili; Lewis, 2014). É importante salientar que α , β e δ guiarão a caça, ou seja, as soluções consideradas ω irão convergir para seus líderes (Kumar; Pant; Ram, 2017).

Inicialmente, o algoritmo cria a população de lobos de forma aleatória. Ao longo das iterações, os lobos *alpha*, *beta* e *delta* estimam a provável posição da presa, essas informações são armazenadas e as demais possíveis soluções alteram sua posição com base nas posições desses três lobos (Mirjalili; Mirjalili; Lewis, 2014).

Por meio da Figura 21 é possível observar a capacidade do lobo se posicionar ao redor de uma presa, nela o lobo (X,Y) tem a liberdade de mudar de posição ao redor da presa (X*,Y*), isso ocorre ao ajustar os vetores A e C das Equações 3.2 e 3.3, em que o componente a da Equação 3.2 diminui linearmente ao longo das iterações (de 2 para 0) e o componentes r_1 e r_2 são valores gerados aleatoriamente para que os lobos possam alcançar qualquer posição ilustrada (Mirjalili; Mirjalili; Lewis, 2014). É possível observar na Equação 3.3 que o vetor C depende somente do valor aleatório de r_2 , isso faz parte da natureza estocástica do algoritmo que tenta promover uma maior exploração com a utilização desse vetor.

Figura 21 – Movimentação durante a perseguição à presa.



Fonte: (Mirjalili; Mirjalili; Lewis, 2014).

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \quad (3.2)$$

$$\vec{C} = 2 \cdot \vec{r}_2 \quad (3.3)$$

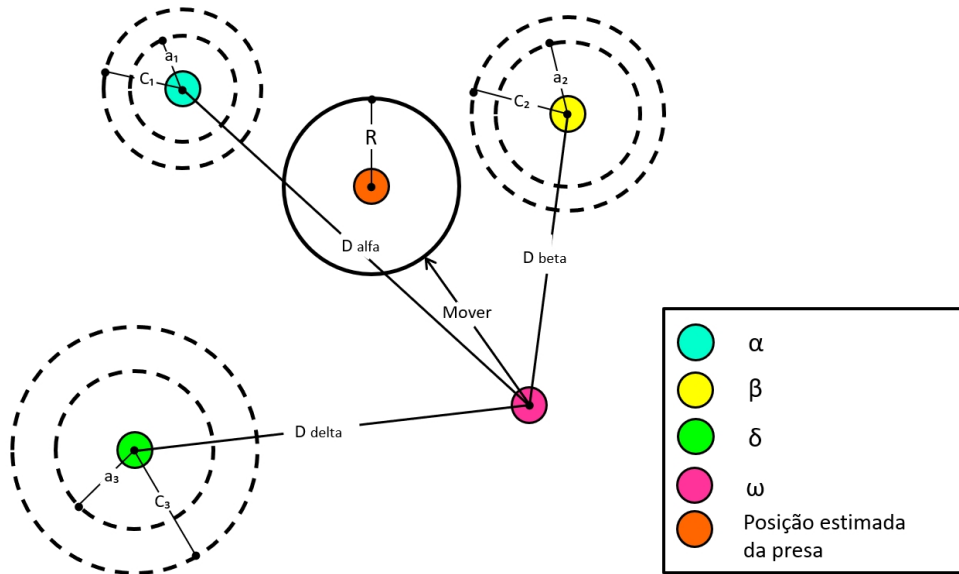
O passo seguinte é atualizar as posições dos demais lobos, isso se dá por meio das Equações 3.4 e 3.5, em que D é a distância a ser calculada entre o lobo e a possível presa, $X_p(t)$ é a atual possível localização da presa, $X(t)$ é a atual localização da melhor solução candidata (*alpha*) (Mirjalili; Mirjalili; Lewis, 2014). Na Figura 22 ilustra-se a atualização do *omega* em direção a presa com base nas posições dos lobos *alpha*, *beta* e *delta*.

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \quad (3.4)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \quad (3.5)$$

Após a atualização das posições, o valor de A de cada lobo é verificado, caso ele seja maior que 1 ($|A| > 1$) indicará que a posição atual do lobo não é a melhor possível,

Figura 22 – Cerco e assédio à presa.

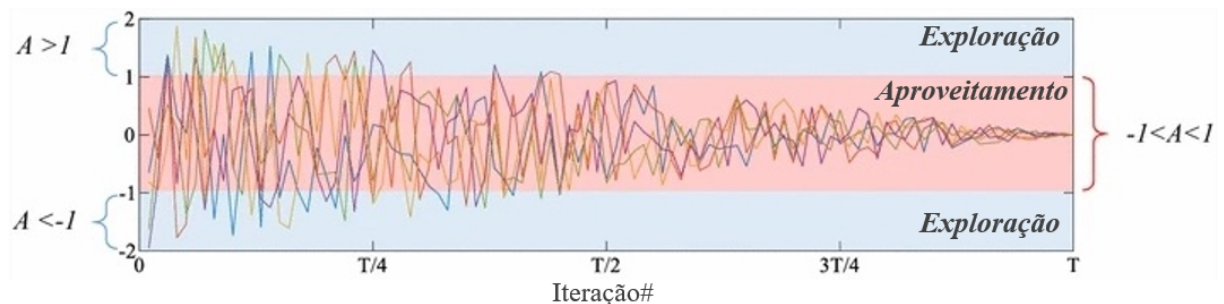


Fonte: (Mirjalili; Mirjalili; Lewis, 2014), traduzido pelo autor.

então ele irá divergir daquela posição. Caso contrário, o valor de A seja menor que 1 ($|A| < 1$), o lobo irá convergir para aquela posição e será forçado a atacar a presa, isso demonstra que ele está no caminho correto.

Por fim, a iteração de busca, exploração e aproveitamento se repete até o algoritmo ser encerrado no cumprimento do critério de parada, isso pode ser observado na Figura 23, em que os autores Faris *et al.* (2018) exemplificam a convergência para uma solução com 5 iterações. Nesse exemplo é possível notar que até a segunda iteração os lobos ainda exploram o ambiente, mas a partir dali a atividade fica toda focada em aproveitar a presa encontrada (intensificação).

Figura 23 – Exemplo de convergência de resultados.



Fonte: (Faris *et al.*, 2018), traduzido pelo autor.

É importante salientar que mesmo com a variável estocástica que introduz o conceito de busca por aleatoriedade do algoritmo, em sua essência, existe a possibilidade da solução candidata ficar presa à boa solução local devido aos lobos convergirem sempre em direção ao *alpha*.

3.4 Considerações Parciais

Neste capítulo foi explicado a diferença entre métodos tradicionais e meta-heurísticos, assim como os diferentes tipos de meta-heurísticos. Também foi abordado um método tradicional e oito diferentes métodos de otimização baseados em Inteligência de Enxame que podem ser adaptados para o processo de tomada de decisão para alocação de recursos.

Os métodos ditos tradicionais foram concebidos por meio de pura formulação matemática, enquanto os métodos meta-heurísticos surgiram com a observação de determinadas espécies de animais, evolução biológica ou em processos físicos. Esses métodos se destacam em cenários que os métodos tradicionais tem menor eficiência, ou seja, o seu uso traz um melhor aproveitamento de recursos computacionais e ganho de tempo para encontrar uma melhor solução.

Uma outra maneira de se pensar é que os métodos meta-heurísticos criam novas oportunidades de pesquisa em diversos campos, não só da computação, mas também da engenharia, negócios, finanças, política e até mesmo produção. É importante notar que esses métodos foram criados para resolver problemas de otimização, ou adaptado para tal, como é o caso de alocação de recursos em Computação de Borda Veicular.

No próximo capítulo serão abordados alguns trabalhos do atual estado da arte que utilizaram otimizadores tradicionais e baseados em Inteligência de Enxame para gerenciar recursos, tarefas e serviços a fim de resolver problemas de otimização.

4 ESTADO DA ARTE

Neste capítulo é apresentado alguns trabalhos publicados nos últimos cinco anos, relacionados com a otimização na alocação de recursos, principalmente em redes veiculares (VANETs), com foco em métodos meta-heurísticos baseados em Inteligência de Enxame. É importante notar que cada uma dessas contribuições representam diferentes cenários com diferentes tecnologias para gerenciamento de recursos, tarefas e serviços, mas todos com o objetivo de otimizar esse processo.

4.1 Trabalhos Relacionados

Em Pereira *et al.* (2020) foi utilizado um algoritmo baseado no Processo Hierárquico Analítico (*Analytic Hierarchy Process* - AHP) para alocação de recursos em *Mobile Edge Computing* por meio de redes 5G. O foco dos autores foi a maximização da quantidade de serviços atendidos ao mitigar problemas de comunicação que atrapalhariam a alocação de recursos. De maneira geral, o algoritmo teve melhor desempenho que seus concorrentes, principalmente em cenários com maior quantidade de usuários. Porém, suas comparações foram apenas com outros algoritmos tradicionais, como Greedy por exemplo, além de não ter sido implementada a mobilidade dos membros da MEC.

Pereira *et al.* (2019) utilizaram um método matemático de decisão de múltiplos atributos para criar o NANCY, um algoritmo que utiliza o método AHP para aplicar um fator de influência em cada parâmetro. Para isso os autores consideraram *Fogs*, em um cenário rodoviário, para alocação de recursos. Foram estipuladas as métricas de serviços atendidos, negados e bloqueados como forma de avaliação. A partir dos resultados, é possível observar o melhor desempenho do NANCY em relação aos seus concorrentes em todas as métricas. Porém, a comparação foi realizada apenas contra outros algoritmos tradicionais e não foi utilizado um simulador de redes para a análise dos resultados.

Em Lieira *et al.* (2022), os autores desenvolveram um algoritmo baseado em WOA, chamado de MARTIAL, para realizar o processo de tomada de decisão durante a alocação de recursos no ambiente de *Edge Computing* (EC). O algoritmo teve melhor desempenho em comparação com os demais algoritmos dos testes, porém os autores não levaram em consideração a quantidade de recursos alocados e a comparação foi feita apenas contra algoritmos ditos tradicionais.

Mohanakrishnan e Ramakrishnan (2020) utilizaram o algoritmo híbrido de WOA com Genético (*Genetic Algorithm* - GA), chamado de Algoritmo de Otimização Genético da Baleia (*Genetic Whale Optimization Algorithm* - GWOA), para conseguir encontrar

canais de rádio subutilizados que possam ser alocados a veículos em uma VANET. A atribuição desses canais de forma eficiente é requisito mínimo em uma VANET para que a comunicação entre os veículos não sofra com perdas e atrasos devido mobilidade dos veículos. Porém, apesar de utilizarem um simulador de redes, os autores não consideraram um mapa com mobilidade real durante as simulações.

Em Ishtiaq *et al.* (2019), foi proposto um *framework* para clusterização e agrupamentos de veículos em uma rede comunitária. Esse algoritmo foi criado com base no MFO e tem como objetivo o agrupamento mais eficiente dos veículos dada a sua proximidade e poder computacional. Os autores chamam de algoritmo inteligente pois não é necessário supervisão alguma para que o agrupamento ocorra. Eles também enfatizam que o método proposto também pode ser utilizado para transmissão de dados por essa rede. O principal benefício de seu uso está na redução do custo de roteamento, pois a rede trabalha com *clusters* menores e mais simples de gerenciar. Contudo, os autores focaram apenas na estabilidade dos *clusters* formados e não testaram a capacidade de transmissão dos dados e atendimento de serviços.

Nematollahi, Ghaffari e Mirzaei (2023) criaram um algoritmo híbrido de MFO e Aprendizagem Baseada em Oposição (*Opposition-based Learning* - OBL) para auxiliar a alocação de recursos e o offloading dentro de *Fogs* em um ambiente de Internet das Coisas (*Internet of Things* - IoT). Os autores concluem que o algoritmo proposto foi capaz de reduzir o fator de atraso para executar o *offloading* e a energia consumida em todo processo de tomada de decisão. Porém, não foi implementado um ambiente de simulação com um simulador de redes, nem foi considerado que alguns membros da *Fog* possam possuir mobilidade.

Em Quessada *et al.* (2021) os autores propuseram um algoritmo baseado no Algoritmo do Morcego, chamado de NAUTILUS, para auxiliar o processo de alocação de recursos em *Clouds* Veiculares. Os autores utilizam o paradigma de *Fog* para auxiliar a alocação de armazenamento, memória, tempo de execução e processamento dos veículos. Ao final, os autores demonstraram que o algoritmo obteve melhor desempenho que outros dois concorrentes em todas as métricas empregadas: serviços atendidos, bloqueados e negados. Contudo, o algoritmo proposto não foi comparado a outro meta-heurístico, apenas à algoritmos ditos tradicionais, além dos autores não considerarem as quantidades de recursos efetivamente alocados como métrica de avaliação.

Em Sun *et al.* (2019a) os autores explanam sobre a dificuldade de alocação de recursos em VECs e propõe modificações no Algoritmo do Morcego para conseguir adaptá-lo a esse problema multiobjetivo. Eles otimizam essa alocação de recursos por meio do escalonamento de tarefas para reduzir o tempo de execução e manter a confiabilidade das mesmas. Os autores compararam a solução proposta com outras três soluções da literatura e concluíram que o algoritmo proposto é capaz de reduzir o tempo de execução

de uma tarefa enquanto mantém a alta confiabilidade do sistema. Porém, não foi utilizado um cenário urbano com mobilidade real dos veículos para os testes, além das simulações não terem sido realizadas contra outros algoritmos meta-heurísticos.

Altabeeb *et al.* (2021) desenvolveram um algoritmo modificado baseado em FA para solução do problema de roteamento de veículos. Esse é um problema conhecido na literatura e pode ser descrito como combinatório e NP-*Hard*. O algoritmo foi implementado e testado em várias instancias de *benchmarks* e obteve melhor rendimento em vários quesitos contra os outros algoritmos testados, como por exemplo custo de roteamento, número de iterações, taxa de convergência e diversidade. Contudo, não foi considerado pelos autores a capacidade dos veículos de alterarem suas rotas durante o percurso, dado a fatores ambientais como trânsito e acidentes, característica inerente à cenários com mobilidade real como é o caso do *Luxembourg SUMO Traffic* (Codeca *et al.*, 2017).

Em Rajput (2022), os autores propuseram um *framework* de provisionamento robusto, cooperativo e confiável de QoS baseado no Algoritmo do Vagalume. O objetivo principal da proposta é a Qualidade de Serviço para redes veiculares em um ambiente de serviços heterogêneos. Foram avaliados cinco diferentes aspectos durante as simulações, como segurança, tráfego de voz, tráfego multimídia, tráfego de texto e a comunicação entre os nós. Porém, os autores não consideraram um mapa urbano, com intersecções, semáforos e tráfego em sentidos opostos, o que além de trazer maior realidade ao ambiente, impactaria diretamente nas métricas avaliadas.

Por meio de uma pesquisa, Zhang, Jiang e Li (2019) propõe o algoritmo Multi-Objetivo baseado em Decomposição da Busca do Cuco (*Cuckoo Search* - CS) para alocação de espectros na Rede Veicular Cognitiva. O trabalho leva em consideração a alta mobilidade da topologia devido a movimentação dos veículos para maximizar a taxa de transferência e proporcionar equidade na rede. Com uma rápida taxa de convergência, o algoritmo decompõe o problema multi-objetivo em sub-problemas com diferentes coeficientes de peso que serão otimizados simultaneamente. A partir dos resultados é possível observar que o algoritmo proposto tem uma taxa de convergência mais rápida e estável que seus concorrentes, além de aumentar a taxa de transferência média de dados. Porém, os autores não consideraram as interferências que esse tipo de rede pode sofrer durante seu funcionamento.

Em Ramakrishnan, Sreedivya e Selvi (2017), é proposto um protocolo adaptativo baseado na Busca do Cuco. Esse protocolo combina recursos de roteamento de topologia e geográficos para garantir a transmissão dos dados de forma segura e sem atrasos. Devido a natureza estocástica do CS, o protocolo consegue variar as rotas e evitar o congestionamento delas, assim é possível garantir uma maior taxa de entrega de pacotes e com menor atraso na comunicação. Porém, os autores não utilizaram um simulador de redes veiculares para analisar os resultados, além de considerarem um cenário único com no

máximo 45 nós.

Lieira *et al.* (2022) utilizaram um método meta-heurístico para otimizar a tomada de decisão durante a alocação de recursos em VANETs. Foi desenvolvido o MARIA, um algoritmo baseado no método meta-heurístico da Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO) e foi utilizado um cenário de *Vehicular Edge Computing* para testá-lo em simulações. Para comparação foram utilizados algoritmos tradicionais da literatura, e sua avaliação se deu pela quantidade de serviços aceitos e recusados. O MARIA demonstrou ser mais eficiente que seus concorrentes na maximização desses atendimentos. Porém, as simulações não consideram as interferências que uma comunicação em rede pode sofrer durante a troca de informações na execução do algoritmo.

Em Midya *et al.* (2018), os autores propuseram um algoritmo para otimizar o processo de alocação de recursos em uma estrutura de Computação em Nuvem Veicular (*Vehicular Cloud Computing* - VCC) de três camadas. O algoritmo é baseado em PSO, e otimiza a decisão de quais recursos serão utilizados para atender uma determinada tarefa por meio dos conceitos de Qualidade de Serviço (*Quality of Service* - QoS). Ele teve melhor performance em todas as três métricas de sua avaliação: atraso de serviço, convergência dos valores de fitness e erro quadrático médio. Porém, os resultados foram obtidos com um pequeno conjunto de subtarefas, além de não considerar a quantidade de serviços atendidos, não atendidos e recursos alocados, assim como o ambiente de simulação controlado e restrito.

Gottsfritz *et al.* (2023a) propuseram um algoritmo baseado em HHO para auxiliar a tomada de decisão durante a alocação de recursos em VANETs. O algoritmo recebe como parâmetros de entrada os dados dos veículos do *cluster* e as informações do serviço a ser executado, e devolve como resposta qual daqueles veículos do *cluster* deverá processar o serviço. O algoritmo proposto teve melhor desempenho que seus concorrentes em todas as métricas de avaliação: serviços alocados, serviços negados e recursos alocados. Porém, não foi considerado diferentes classes de serviços para melhor análise do comportamento do algoritmo com ambientes mais restritivos ou com menor margem para erro.

Em Dehkordi *et al.* (2021) foi proposto um algoritmo modificado baseado na Otimização do Falcão Harris com uso de mapa caótico e parâmetros de controle não lineares para resolver problemas de otimização em Internet de Veículos (*Internet of Vehicles* - IoV). As áreas beneficiadas por essa contribuição incluem alocação de recursos, recuperação de informação e planejamento de rotas. A partir dos resultados, é possível observar que o algoritmo proposto teve melhor resultado que seus concorrentes, tanto nas funções de *benchmark* como no problema de roteamento para *Internet of Vehicles* abordado no trabalho. Porém, não foi utilizado um simulador de redes veiculares para analisar os resultados, assim como não ficou claro se foi implementada a mobilidade dos

veículos nos testes realizados.

Lieira *et al.* (2021) propuseram um mecanismo de otimização de tarefas baseado em GWO, nomeado de TOVEC, que tem por objetivo melhorar o uso dos recursos disponíveis em uma *Vehicular Cloud* por meio da maximização na alocação de tarefas. A partir dos resultados, é possível observar um aumento significativo na quantidade de tarefas atendidas e recursos utilizados, em comparação com seus concorrentes. Porém, a comparação é realizada apenas contra algoritmos ditos tradicionais da literatura, ou seja, o TOVEC não é comparado a outra meta-heurística.

Por meio de seu trabalho, Brahmi, Hamdi e Zarai (2021) propuseram um algoritmo de otimização baseado em GWO e Teoria do Caos para a tomada de decisão na alocação de recursos em redes Veículos-a-Tudo (*Vehicles-to-Everything* - V2X). O objetivo é maximizar a taxa de transferência total do sistema e manter a qualidade do serviço oferecido aos dispositivos. O algoritmo foi comparado a outras duas meta-heurísticas (PSO e GWO puro) e a partir dos resultados pode-se observar que mesmo com uma taxa de convergência menor do algoritmo proposto, ele foi capaz de ter uma melhor performance em todas as demais métricas discutidas, pois os autores consideram que o algoritmo concorrente sofreu uma convergência prematura. Contudo, os autores não implementaram um cenário urbano e com mobilidade real para as simulações, além de não avaliarem a quantidade de recursos que poderiam ser efetivamente alocados.

4.2 Considerações Parciais

Para melhor demonstrar os trabalhos citados na Seção 4.1, foi elaborada uma tabela que resume as principais características e tópicos abordados pelos autores. Em outras palavras, a Tabela 2 relaciona o método base para o desenvolvimento do algoritmo dos trabalhos, a estrutura de rede utilizada nas simulações e o problema a ser resolvido/otimizado.

Com exceção dos trabalhos (Pereira *et al.*, 2019; Pereira *et al.*, 2020), todos os demais citados na Seção 4.1 utilizaram algoritmos baseados em métodos meta-heurísticos de Inteligência de Enxame, mas para diferentes finalidades em diferentes estruturas. Em Nematollahi, Ghaffari e Mirzaei (2023) o algoritmo foi empregado para realizar o *offloading* de dados para uma Fog, além da alocação desses recursos. Em Pereira *et al.* (2020), Pereira *et al.* (2019), Quessada *et al.* (2021), Zhang, Jiang e Li (2019), Lieira *et al.* (2022), Midya *et al.* (2018), Gottsfritz *et al.* (2023a), Brahmi, Hamdi e Zarai (2021), que possui o maior volume das pesquisas apresentadas, alocação de recursos. Desde recursos de processamento e armazenamento, até recurso de rede e banda, nos mais variados tipos de infraestrutura como: MEC, Fog, VANETs com e sem Radio Cognitivo, VEC, VCC e V2X.

Tabela 2 – Comparações dos Trabalhos Relacionados

Trabalho	Método	Estrutura	Problema
Pereira <i>et al.</i> (2020)	AHP	MEC	Alocação de Recursos
Pereira <i>et al.</i> (2019)	AHP	Fog	Alocação de Recursos
Lieira <i>et al.</i> (2022)	WOA	EC	Alocação de Tarefas
Mohanakrishnan e Ramakrishnan (2020)	WOA + GA	VANET + CR	Roteamento
Ishtiaq <i>et al.</i> (2019)	MFO	VANET	Clusterização + Comunicação
Nematollahi, Ghaffari e Mirzaei (2023)	MFO + OBL	Fog	Alocação de Recursos + <i>Offloading</i>
Quessada <i>et al.</i> (2021)	BAT	Fog	Alocação de Recursos
Sun <i>et al.</i> (2019a)	BAT	VEC	Alocação de Tarefas
Altabeeb <i>et al.</i> (2021)	FA		Roteamento
Rajput (2022)	FA	VANET	Roteamento
Zhang, Jiang e Li (2019)	CS	VANET + CR	Alocação de Recursos
Ramakrishnan, Sreedivya e Selvi (2017)	CS	VANET	Roteamento
Lieira <i>et al.</i> (2022)	PSO	VEC	Alocação de Recursos
Midya <i>et al.</i> (2018)	PSO	VCC	Alocação de Recursos
Gottsfritz <i>et al.</i> (2023a)	HHO	VANET	Alocação de Recursos
Dehkordi <i>et al.</i> (2021)	HHO	IoV	Roteamento
Lieira <i>et al.</i> (2021)	GWO	VCC	Alocação de Tarefas
Brahmi, Hamdi e Zairai (2021)	GWO + Caos	V2X	Alocação de Recursos
RAPTOR	HHO	VEC	Alocação de Recursos

Fonte: Produzido pelo autor.

Outra utilização para os métodos meta-heurísticos é na alocação de tarefas, como pode ser observado em Lieira *et al.* (2022), Sun *et al.* (2019a), Lieira *et al.* (2021). Esses trabalhos utilizaram diferentes estruturas (EC, VEC e VCC respectivamente) e diferentes algoritmos para alocar de modo otimizado as tarefas solicitadas por usuários da rede.

Outro uso que se repetiu foi para a tomada de decisão em encaminhamento de pacotes de roteamento veicular, como pode ser observado em Mohanakrishnan e Ramakrishnan (2020), Altabeeb *et al.* (2021), Rajput (2022), Ramakrishnan, Sreedivya e Selvi (2017), Dehkordi *et al.* (2021), e abordado principalmente na estrutura de VANET, mas também para IoV. E por fim, em Ishtiaq *et al.* (2019), foi utilizado um algoritmo baseado em um método meta-heurístico para otimizar o sistema de agregação e comunicação entre os veículos em uma VANET.

Com base na pesquisa apresentada, este trabalho propõe o RAPTOR, um algoritmo que utiliza o método meta-heurístico da Otimização do Falcão Harris (HHO) para

otimizar a tomada de decisão durante a alocação de recursos em Computação de Borda Veicular. A escolha do HHO se dá por ele ser um método novo e conseguir performar de maneira satisfatória a alocação de recursos em redes veiculares como visto em Gottsfritz *et al.* (2023a) e Dehkordi *et al.* (2021), inclusive com melhor performance que outros meta-heurísticos.

A capacidade de gerar várias soluções candidatas e aprimorá-las de diferentes modos por meio do seu método iterativo, assim como a possibilidade de modelá-lo para definir como será alocado determinado recurso disponível na Borda Veicular, torna esse método de otimização adaptável o suficiente para resolver problemas de alocação de recursos em Bordas Veiculares com ganho de performance suficiente para torná-lo uma escolha confiável no lugar de seus concorrentes.

5 ALGORITMO METRA-HEURÍSTICO E ALOCAÇÃO DE RECURSOS EM VEC

Neste capítulo é apresentado o RAPTOR, uma modelagem do método de Otimização do Falcão Harris, criado para gerenciar a alocação de recursos em uma VEC. Essa modelagem consiste na adaptação do modelo matemático proposto por Heidari *et al.* (2019) e codificado por Faris *et al.* (2016), disponibilizado de forma gratuita para a comunidade científica por meio do Github¹.

A escolha do HHO como método base para o gerenciamento de recursos dentro de uma Borda Veicular se deve à adaptabilidade e resultados alcançados com o uso de duas diferentes estratégias na fase de exploração e quatro diferentes estratégias na fase de aproveitamento do modelo em outros tipos de redes veiculares (Gottsfritz *et al.*, 2023a; Gottsfritz *et al.*, 2023b). Isso o torna capaz de lidar com a volatilidade desse tipo de rede, em que se têm inúmeras requisições de serviços a serem atendidas pelos recursos em movimento e dispersos na *Edge*.

A estrutura deste capítulo é a seguinte: na Seção 5.1 é explicado a estrutura em que a simulação se insere com os recursos dos veículos e requisitos dos serviços, premissas da simulação e a organização dos veículos em uma VEC; na Seção 5.2 é descrito o funcionamento do algoritmo proposto com suas premissas, pseudocódigo e fórmulas; na Seção 5.3 é abordado sobre o ambiente com as ferramentas necessárias para sua criação, o cenário utilizado, parâmetros de simulação, métricas de avaliação e uma breve explicação sobre os algoritmos utilizados nas comparações; e por fim, na Seção 5.4, são apresentados e discutidos os dados obtidos por meio das simulações.

5.1 Estrutura

A estrutura escolhida para o trabalho foi a Computação de Borda Veicular, apresentada no Capítulo 2, em que veículos inteligentes se agrupam, sem Estrutura de Beira de Estrada no caso, para que consigam executar serviços que antes não seriam possíveis sozinhos (Meneguette *et al.*, 2021). Isso se dá pela solução conseguir oferecer uma capacidade computacional a um veículo, que pertence a outro. Ou seja, ao invés desse veículo ter apenas os seus recursos disponíveis para executar algum serviço, ele poderá solicitar que outro veículo inteligente da VEC o faça.

Cada veículo inteligente que circula no cenário possui uma quantidade finita de

¹ <https://github.com/7ossam81/EvoloPy>

recursos de: Armazenamento, Processamento e Memória. Eles estarão disponíveis para compartilhamento com seus vizinhos assim que ociosos. Logo, é considerado um conjunto de veículos V em que $\{V = 1, \dots, n\}$, em que todos os veículos possuem um identificador único v_{id} , capacidade de armazenamento v_{ar} , processamento v_{pr} e memória v_{me} . Eles possuem diferentes capacidades de cada um desses recursos, isso ajuda na representatividade heterogênea da frota urbana.

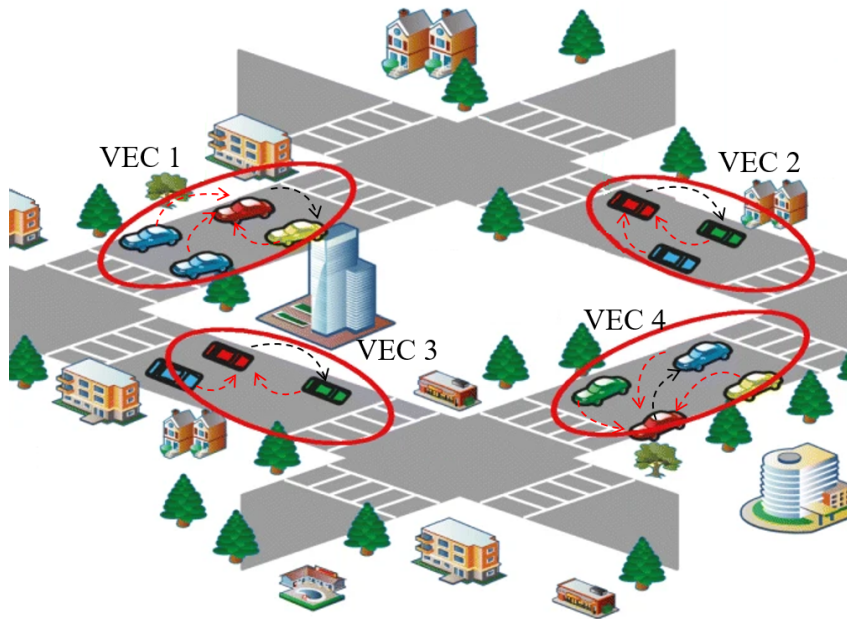
Cada serviço possui requisitos a serem cumpridos para que sua execução possa ocorrer, e eles são os mesmos dos recursos que os veículos possuem: Armazenamento, Processamento e Memória. Logo, é considerado um conjunto S em que $\{S = 1, \dots, n\}$, em que todos os serviços possuem um identificador único s_{id} , necessidade de armazenamento s_{ar} , processamento s_{pr} e memória s_{me} . Assim como nos veículos, cada serviço possui uma necessidade diferente para cada um desses requisitos, o que auxilia a representar a heterogeneidade dos serviços computacionais que podem estar presentes em um ambiente veicular como: trocas de mensagens em tempo real, monitoramento de tráfego, sistema de alerta de acidentes, sugestões de rotas, assim como serviços de entretenimento de áudio/vídeo.

Na realização das simulações, algumas premissas foram definidas para que sua execução fosse homogênea, ou seja, não houvesse diferenças no cenário entre a execução de um algoritmo ou outro:

- Os veículos se locomovem segundo padrões de mobilidade impostos pelo cenário LuST (Codeca *et al.*, 2017), além de respeitarem suas leis de trânsito;
- Todos os veículos são inteligentes e possuem a capacidade de se comunicar diretamente (comunicação veículo-a-veículo - V2V);
- Os veículos localizam seus vizinhos pela emissão de mensagens *beacon*, e assim, podem se organizar em uma *Vehicular Edge Computing*. A partir desse momento cada membro é conhecido como *Edge Node*;
- Após a organização, ocorre a eleição de um líder por meio da política de eleição LID, esse membro passa a ser chamado de Líder;
- O Líder possui uma fila de serviços que serão executados pelos membros da sua VEC (*Edge Nodes*), a frequência em que esses serviços chegam até ele é uma distribuição de Poisson com valor de Lambda λ definido na simulação;
- O Líder é responsável por executar o algoritmo selecionado, pois ele detém a informação sobre os recursos ociosos dos *Edge Nodes* e também os requisitos dos serviços que chegaram até ele.

Por meio da Figura 24, é possível observar a formação da VEC com a agregação e troca de mensagens entre os veículos próximos enquanto trafegam pela via. Na Figura 24 estão representados quatro diferentes VECs (1, 2, 3 e 4), em cada uma delas o veículo vermelho representa o Líder, as setas vermelhas são mensagens enviadas dos membros (*Edge Nodes*) para o Líder, como requisições de recursos para execução de um serviço, e a seta preta representa uma resposta do Líder, seja de aceite à requisição até um pedido para que o membro execute um dos serviços que está na fila..

Figura 24 – Abstração do cenário de *Edge* com representação de comunicação.



Fonte: (Liu *et al.*, 2021), adaptado pelo autor.

5.2 Algoritmo

Nesta seção é explicado em detalhes o RAPTOR, mecanismo proposto por este trabalho, que gerenciará a alocação de recursos entre os veículos de uma VEC. Esse mecanismo é baseado no método da Otimização do Falcão Harris (HHO) (Heidari *et al.*, 2019), o qual é capaz de escolher o *Edge Node* que atenderá um serviço.

Como mencionado anteriormente, a escolha do HHO como método base para o algoritmo se deve a capacidade desse método de lidar com a adversidade de uma rede veicular, especificamente uma VEC, como por exemplo a dinamicidade inerente dos veículos. Isso se dá pelo uso de duas estratégias na fase de exploração do algoritmo e quatro estratégias na fase de aproveitamento.

O algoritmo considera que cada *Edge Node* seja um falcão da população, e o serviço a ser executado seja a presa. Além disso, algumas outras premissas foram estabelecidas para o correto funcionamento do RAPTOR:

1. O algoritmo considerará cada *Edge Node* como um membro da população;
2. Ao fim da rotina, o membro da população com menor valor de *fitness* terá seus recursos alocados para o serviço solicitado;
3. Sempre haverá perseguição, ou seja, os falcões sempre perseguirão a presa;
4. A ação dos falcões depende do valor de 3 variáveis: Energia ($|E|$), capacidade de fuga (r) e uma variável estocástica (q) para decidir o tipo de exploração;
5. Apesar dos falcões terem suas primeiras posições geradas por meio da distância euclidiana para com os requisitos do serviço, as suas atualizações acontecem com base em informações da presa (posição, energia e potência de salto), dos falcões (posição média da população) e variáveis estocásticas.

A variável $|E|$ reproduz a energia da presa, logo é ela a variável que define em qual fase o algoritmo está. Essa energia tende a decrescer a cada iteração, com valores entre 0 e 2, em que 2 significa que a presa está totalmente fortalecida e 0 totalmente exaurida.

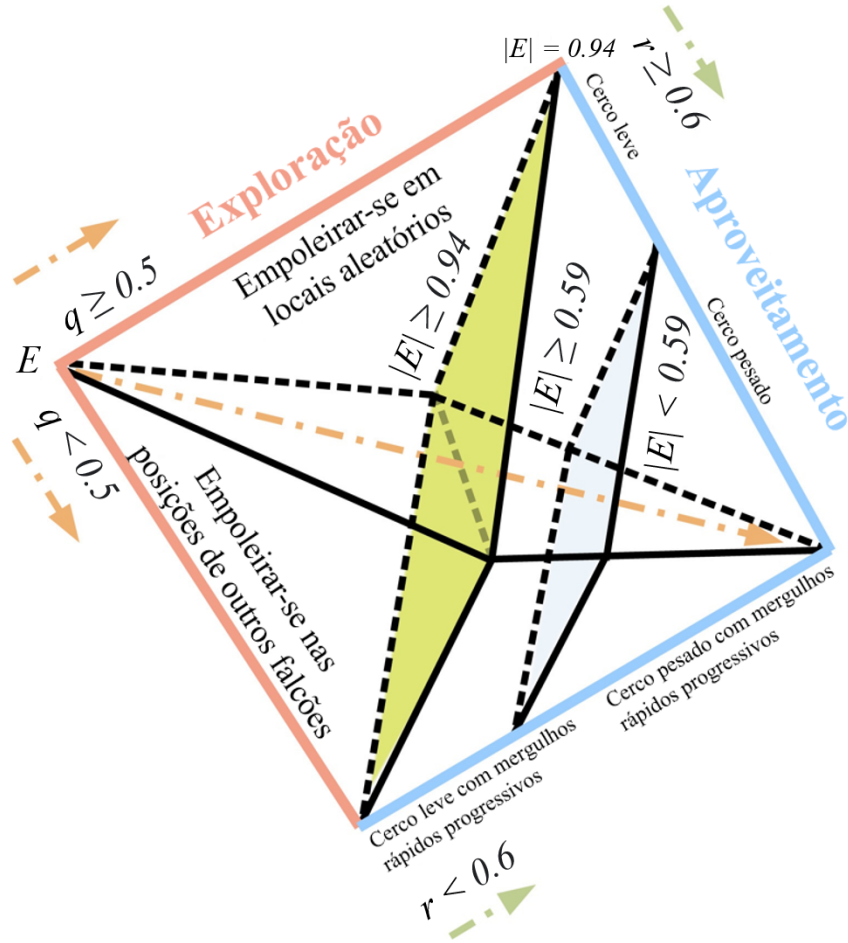
Por meio da Figura 25 é possível notar o controle que $|E|$ exerce sobre as duas transições que controla. A primeira acontece caso seu valor decresça além de $|E| < 0.94$, então o algoritmo irá parar de explorar o ambiente como um todo e iniciará os movimentos de caça à presa (aproveitamento) por meio do Cerco Leve. A segunda transição ocorre se o seu valor decrescer novamente, agora para $|E| < 0.59$, em que o algoritmo iniciará o Cerco Pesado contra a presa.

A variável q está ligada apenas ao processo de exploração. Ela define se durante a fase de exploração as atualizações irão ocorrer aleatoriamente, quando $q \geq 0.5$, ou por meio da posição média da população, quando $q < 0.5$. A chance para cada um desses casos é de 50%, ou seja, o valor de q é gerado a cada iteração durante a fase de exploração, e seus valores vão de 0 a 1.

Por fim, a variável r , que também é estocástica e também varia entre (0,1), em que $r < 0.6$ indica que a presa tem chance de escapar do cerco, seja ele leve ou pesado, e $r \geq 0.6$ que indica a incapacidade da presa de se desvencilhar dos seus predadores. Ou seja, se a presa tiver chances de escapar do cerco, os falcões realizarão mergulhos rápidos e progressivos para conseguir acompanhar a presa durante o cerco, isso é reproduzido no algoritmo com o uso do conceito de *Levy Flight* (LF) (Barthelemy; Bertolotti; Wiersma, 2008).

Dessa forma, as variáveis $|E|$ e r se combinam para comporem as quatro diferentes estratégias de aproveitamento do algoritmo:

Figura 25 – Descrição das fases de execução do RAPTOR.



Fonte: (Heidari *et al.*, 2019), adaptado pelo autor.

1. **Cercos Leves:** ocorrerá se $|E| \geq 0.59$ e $r \geq 0.6$, isso significa que a presa tem energia para fugir do anel de cerco mas não consegue fugir dos falcões. Para essa situação os falcões irão assediá-la para exaurir sua energia $|E|$ antes de abate-la (Heidari *et al.*, 2019). Nesse cálculo é levado em consideração a posição da presa, sua energia e uma variável aleatória J que representa a potência de salto na tentativa de fuga (Qu *et al.*, 2020).
2. **Cercos Pesados:** Sua ocorrência se dá logo que $|E| < 0.59$ e $r \geq 0.6$, isso significa que a presa está exausta e sem possibilidade de fuga dos falcões. Aqui o ataque surpresa de fato ocorre para que a presa seja abatida (Qu *et al.*, 2020). As posições são atualizadas com base apenas nas posições atuais e energia da presa.
3. **Cercos Leves com Mergulhos Rápidos e Progressivos:** Ocorre se $|E| \geq 0.59$ e $r < 0.6$, pois a presa tem energia para escapar do cerco e a capacidade de fugir dos falcões. Para esse caso o conceito de *Levy Flyght* é aplicado pelos criadores do método, para fidelizar os movimentos da presa em fuga e dos falcões ao redor dela (Song *et al.*, 2021). Aqui a atualização de posições utiliza as mesmas variáveis do

cerco leve (posição da presa, sua energia e potencia do salto), porém caso a posição não melhore após a sua atualização, é executado uma nova atualização de posição que irá considerar a nova posição, um vetor aleatório e o valor obtido por meio da equação de LF (Heidari *et al.*, 2019).

4. **Cerco Pesado com Mergulhos Rápidos e Progressivos:** Como última estratégia de ataque, assim que as variáveis atingirem os limites de $|E| < 0.59$ e $r < 0.6$, significará que a presa não tem energia para fugir do anel de cerco mas tem capacidade de fugir dos falcões (como um esconderijo, por exemplo). Com isso os falcões conduzem a presa para longe do esconderijo e ao mesmo tempo reduzem a distância entre eles enquanto se preparam para abate-la (Song *et al.*, 2021). Da mesma forma que na estratégia supracitada, caso a atualização de posição não traga ganho à solução, uma nova atualização de posição é realizada, agora com o valor de LF.

Uma parte importante de toda a rotina do algoritmo é o cálculo do valor de *fitness*, ele ocorrerá para cada membro da população e a cada iteração. Esse valor exprime a proximidade do falcão com a presa e é utilizado ao final das iterações para indicar o *Edge Node* que atenderá a execução do serviço, no caso, aquele membro com menor valor de *fitness* o fará. Para o seu cálculo será utilizado a função de *benchmark* de Ackley (Eq. 5.1), conhecida pela comunidade científica (Wong; Ming, 2019; Hussain *et al.*, 2017).

$$\begin{aligned}
 Ackley = & -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) \\
 & - \exp \left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i) \right) + 20 + e
 \end{aligned} \tag{5.1}$$

A Equação 5.1 recebe como parâmetro o vetor X_i , calcula sua dimensão D , em que $D = \text{len}(X)$ e prossegue com o cálculo da equação. Ela terá como saída o valor de *fitness* do membro i da população.

O algoritmo 1 é um pseudo-código elaborado para facilitar o entendimento do RAPTOR. Nele é possível observar as condições para cada acontecimento, e sua ordem, de maneira inteligível.

Logo após o seu cabeçalho são declaradas as variáveis de entrada, que é uma matriz dos membros da VEC e um vetor do serviço a ser alocado. A variável de saída *melhorMembro* retornará o V_{id} do *Edge Node* escolhido pelo algoritmo para atender o serviço.

Algoritmo 1 Pseudo-código do RAPTOR**Entrada:** *membros, servico***Saída:** *melhorMembro*

```

1:  $dim \leftarrow 3$ 
2:  $Populacao \leftarrow \text{len}(membros)$ 
3: para  $i$  in  $\text{range}(0, dim)$  faça
4:   para  $j$  in  $\text{range}(0, Populacao)$  faça
5:     Gera as primeiras posições da população.  $X[j,i] \leftarrow \text{euclidiana}(membros[j,i], \text{servico}[i])$  Eq.5.2
6:   fim para
7: fim para
8: para  $i$  in  $\text{range}(0, Populacao)$  faça
9:    $fitness = \text{Ackley}(X_i)$ 
10:  se  $fitness < \text{melhorFitness}$  então
11:    Atualiza o vetor da melhor solução.  $X_{best} \leftarrow X_i$ 
12:     $\text{melhorFitness} \leftarrow fitness$  |  $\text{melhorMembro} \leftarrow i$ 
13:  fim se
14: fim para
15: para  $t$  in  $\text{range}(0, Populacao)$  faça
16:   para  $i$  in  $\text{range}(0, Populacao)$  faça
17:     Atualiza a energia  $E_0$ . Eq. 5.3
18:     Atualiza a energia  $E$ . Eq.5.4
19:     se  $(|E| \geq 0.94)$  então
20:        $q \leftarrow \text{rand}()$ 
21:       se  $(q \geq 0.5)$  então
22:         Atualiza o vetor de localização  $X_i$  aleatoriamente. Eq.5.5
23:       senão se  $(q < 0.5)$  então
24:         Calcula posição média da população  $X_m$ . Eq. 5.6
25:         Atualiza o vetor de localização  $X_i$ . Eq.5.7
26:       fim se
27:     fim se
28:     se  $(|E| < 0.94)$  então
29:        $r \leftarrow \text{rand}()$ 
30:       se  $(r \geq 0.6 \ \& \ |E| \geq 0.59)$  então
31:         Atualiza força do salto  $J$ . Eq.5.8
32:         Atualiza o vetor de localização  $X_i$ . Eq.5.9
33:       senão se  $(r \geq 0.6 \ \& \ |E| < 0.59)$  então
34:         Atualiza o vetor de localização  $X_i$ . Eq.5.10
35:       senão se  $(r < 0.6 \ \& \ |E| \geq 0.59)$  então
36:         Atualiza força do salto  $J$ . Eq.5.8
37:         Calcula  $Y$ . Eq.5.11
38:         se  $\text{Ackley}(Y) < fitness$  então
39:           Atualiza o vetor de localização  $X_i$ .
40:         senão
41:           Calcula  $Z$ . Eq.5.12
42:           se  $\text{Ackley}(Z) < fitness$  então
43:             Atualiza o vetor de localização  $X_i$ .
44:           fim se
45:         fim se
46:       senão se  $(r < 0.6 \ \& \ |E| < 0.59)$  então
47:         Atualiza força do salto  $J$ . Eq.5.8
48:         Calcula posição média da população  $X_m$ . Eq. 5.6
49:         Calcula  $K$ . Eq.5.14
50:         se  $\text{Ackley}(K) < fitness$  então
51:           Atualiza o vetor de localização  $X_i$ .
52:         senão
53:           Calcula  $W$ . Eq.5.15
54:           se  $\text{Ackley}(W) < fitness$  então
55:             Atualiza o vetor de localização  $X_i$ .
56:           fim se
57:         fim se
58:       fim se
59:     fim se
60:      $fitness \leftarrow \text{Ackley}(X_i)$ 
61:     se  $(fitness < \text{melhorFitness})$  então
62:       Atualiza o vetor da melhor solução.  $X_{best} \leftarrow X_i$ 
63:        $\text{melhorFitness} \leftarrow fitness$  |  $\text{melhorMembro} \leftarrow i$ 
64:     fim se
65:   fim para
66: fim para
67: retorna melhorMembro

```

Na linha 1, a variável *dim* é inicializada com o valor 3, ela representa a quantidade de dimensões que o cenário possui. No caso são três dimensões devido a cada veículo possuir três recursos e à tarefa possuir três requisitos. Na linha 2 variável *Populacao* recebe o tamanho de *membros*, ou seja, se a *Edge* possuir dez veículos, então *Populacao* ← 10.

Nas linhas 3 à 7 serão geradas as primeiras posições da população. Na linha 3 tem-se uma estrutura de repetição “para” em que *i* vai de 0 à *dim*, ou seja, até 3, com uma outra estrutura de repetição “para” em que *j* vai de 0 à *Populacao* (linha 4). Na linha 5, as primeiras posições da população serão geradas por meio do cálculo da distância Euclidiana da matriz de *membros* com o vetor de *servico*, ou seja, os valores de “Entrada” do algoritmo serão utilizados para a Equação 5.2.

$$euclidiana = \sqrt{\sum_{j=1}^n (s_{ij} - f_{ij})^2} \quad (5.2)$$

Nas linhas 8 à 14 serão calculados os valores de *fitness* de todos os membros da população, e será armazenado apenas o menor deles. Então, na linha 8 é iniciado uma estrutura de repetição “para” em que *i* vai de 0 à *Populacao*, a variável *fitness* recebe o valor calculado pela função de *benchmark* de Ackley (Eq. 5.1) (linha 9) e a compara com o valor da variável *melhorFitness*, que representa o menor valor de *fitness* encontrado até o momento (linha 10).

Caso o valor recém calculado seja menor que o registrado, o algoritmo avançará para a linha 11, caso seja maior ele finalizará a estrutura condicional “se” e irá para o próximo membro, com a iteração de *i*. Ao avançar para a linha 11, o algoritmo irá atualizar o vetor de melhor solução X_{best} com os valores do membro setado em *i*, e posteriormente na linha 12, irá substituir o valor da variável *melhorFitness* pelo da variável *fitness* e gravará o valor de *i* em *melhorMembro*, pois é necessário registrar o *ID* do melhor membro.

Da linha 15 à 65, encontra-se o *core* principal de execução do algoritmo. Na linha 15, o primeiro dos principais laços de repetição do algoritmo se inicia, é um laço “para” em que *t* vai de 0 à *Populacao*. Ele é seguido de outro lado “para” (linha 16) em que *i* irá de 0 à *Populacao* também. Essa métrica variável foi escolhida devido a sua eficácia em encontrar o veículo ideal tanto para *Edges* menores, em que o algoritmo não precisa de tanto esforço para encontrar o veículo ideal, como em *Edges* grandes, que demanda maior esforço.

O laço da linha 15 exprime tempo, ele define quantas vezes tudo irá ocorrer, enquanto o laço da linha 16 exprime os membros da população, ou seja, os *Edge Nodes*. Em uma situação hipotética, no tempo $t = 1$ o segundo laço se iniciará pelo primeiro

membro da *Edge* ($i = 0$) e executará toda a rotina, e que cada vez que i for iterado, será um membro diferente que estará selecionado.

Com o membro da população selecionado, o algoritmo ingressa no laço por meio da linha 17, que calculará a “Energia Inicial” E_0 por meio da Equação 5.3. O valor gerado aleatoriamente $rand()$ vai de 0 à 1, ou seja, ele sempre estará no intervalo $-1 < E_0 < 1$.

$$E_0 = 2 \times rand() - 1 \quad (5.3)$$

Com o valor da “Energia Inicial” calculado, o algoritmo avança para a linha 18, que utilizará o valor de E_0 para calcular a “Energia” E por meio da Equação 5.4. O valor de E tende a decrescer, pois com o passar do tempo o valor de t sobe e *Populacao* tem seu valor fixado no início do algoritmo.

$$E = 2E_0(1 - \frac{t}{Populacao}) \quad (5.4)$$

Na linha 19 acontece a primeira verificação do valor de E , em que $|E| \geq 0.94$ indica que o algoritmo está na fase de “Exploração”. Nesse caso, será gerado um valor aleatório entre 0 e 1 para a variável q (linha 20). Se $q \geq 0.5$ (linha 21), a posição do membro selecionado é atualizada de forma aleatória pela Equação 5.5 (linha 22) em que: $X_{rand}(t)$ é a posição de um membro aleatório da população, r_1 e r_2 são variáveis estocásticas com valores entre 0 e 1, e $X(t)$ é a atual posição do membro da população selecionado.

$$X(t+1) = X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)| \quad (5.5)$$

Porém, se $q < 0.5$ (linha 23), então será calculado a posição média da população X_m por meio da Equação 5.6 (linha 24), em que $X_i(t)$ indica a localização de cada membro da população na iteração t . Com $X_m(t)$ calculado, o algoritmo avança para a linha 25 que atualizará a posição do membro i por meio da Equação 5.7, em que: $X_{best}(t)$ é a posição da melhor solução encontrada até o atual tempo t , r_3 e r_4 são variáveis estocásticas com valores entre 0 e 1, LB é o limite inferior das variáveis com seu valor dado por $LB = -2 * dim$, e UB é o limite superior com seu valor dado por $UB = 2 * dim$. LB e UB auxiliam a equação a não causar uma atualização de posição que seja discrepante com o que um falcão possa fazer na vida real (Song *et al.*, 2021).

$$X_m(t) = \frac{1}{Populacao} \sum_{i=1}^{Populacao} X_i(t) \quad (5.6)$$

$$X(t+1) = (X_{best}(t) - X_m(t)) - r_3(LB + r_4(UB - LB)) \quad (5.7)$$

Na linha 28 acontece a segunda verificação de $|E|$. O algoritmo transita para a fase de “Aproveitamento” e quatro possíveis ações podem ser executadas caso $|E| < 0.94$. Essas ações dependerão dos valores de $|E|$ e da variável aleatória r que é gerada na linha 29 e tem seu valor entre 0 e 1.

Na linha 30 começam as verificações das combinações dos valores de $|E|$ e r para a fase de “Aproveitamento”. Se $r \geq 0.6$ e $|E| \geq 0.59$, então será atualizado o valor da força de salto J por meio da Equação 5.8 (linha 31) em que $rand()$ é um número aleatório entre 0 e 1. E, na linha 32, é atualizada a posição do membro i por meio da Equação 5.9.

$$J = 2 \times (1 - rand()) \quad (5.8)$$

$$X(t+1) = (X_{\text{best}}(t) - X(t)) - E|JX_{\text{best}}(t) - X(t)| \quad (5.9)$$

Caso a condição da linha 30 não satisfaça, a próxima a ser checada será a linha 33 que será verificada se $r \geq 0.6$ e $|E| < 0.59$. Caso verdadeira, a posição do membro i será atualizada de acordo com a Equação 5.10.

$$X(t+1) = X_{\text{best}}(t) - E|X_{\text{best}}(t) - X(t)| \quad (5.10)$$

Novamente, caso a condição da linha 33 não satisfaça, a terceira checagem é feita na linha 35, e será verdadeira se $r < 0.6$ e $|E| \geq 0.59$, e então é calculada a força do salto J por meio da Equação 5.8 (linha 36), para que possa ser calculada a posição temporária Y na linha 37 por meio da Equação 5.11

$$Y = X_{\text{best}}(t) - E|JX_{\text{best}}(t) - X(t)| \quad (5.11)$$

Com a posição Y calculada, na linha 38 é checado se o valor de *fitness* da posição temporária Y é menor do que o atual valor da variável *fitness*. Caso verdadeiro, a linha 39 é executada e a posição do membro selecionado se torna a posição Y calculada. Caso contrário o algoritmo progride para a linha 41, e será calculado a posição temporária Z por meio da Equação 5.12, em que são utilizados: a posição Y , um vetor aleatório S de tamanho $1 \times \text{dim}$ e o valor da função de *Levy Flight* $LF(\text{dim})$ dado pela Equação 5.13, em que u e v são variáveis aleatórias com intervalo entre 0 e 1, β é uma constante setada em 1.5 e Γ representa a Função Gama definida por Meerschaert (2013) (Song *et al.*, 2021).

$$Z = Y + S \times LF(\text{dim}) \quad (5.12)$$

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}, \quad \sigma = \left(\frac{\Gamma(1 + \beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \quad (5.13)$$

Com a posição Z calculada, a linha 42 irá comparar se o valor de *fitness* da posição Z é menor que o valor gravado na variável *fitness*, caso verdadeiro o algoritmo irá substituir o valor da posição do membro pelo valor da posição Z (linha 43), ou seja, o membro sairá da sua posição original e se deslocará para a posição calculada Z . E com isso a terceira combinação possíveis valores de r e $|E|$, se completa, finalizada na linha 45. É importante notar que, nesse caso, só ocorrerá atualização de posição caso a nova posição se mostre melhor do que a anterior, e isso é medido pelo valor de *fitness*.

Caso as três verificações anteriores da fase de “Aproveitamento” não se cumpram, a única combinação que resta é a da linha 46, que será verdadeira se $r < 0.6$ e $|E| < 0.59$. Nesse caso, a força do saldo J é atualizada por meio da Equação 5.8 na linha 48, a posição média da população é calculada por meio da Equação 5.6 na linha 49 e é calculado a posição temporária K por meio da Equação 5.14.

$$K = X_{\text{best}}(t) - E |JX_{\text{best}}(t) - X_m(t)| \quad (5.14)$$

De forma parecida a condição da linha 38, a linha 50 irá comparar o valor de *fitness* da posição temporária K com o valor da variável *fitness*, caso seja menor o membro terá sua posição atualizada com o valor de K (linha 51). Caso contrário, será calculado a posição temporária W por meio da Equação 5.15 (linha 53), em que será utilizada a posição temporária K , um vetor aleatório S de tamanho $1 \times \text{dim}$ e o valor da função de *Levy Flight* $LF(\text{dim})$ dado pela Equação 5.13.

$$W = K + S \times LF(\text{dim}) \quad (5.15)$$

E novamente, na linha 54, serão comparados os valores de *fitness* agora da posição W com o da variável *fitness*, e caso $Ackley(W)$ seja menor, o membro i terá sua posição atualizada com o valor de W (linha 55). Das linhas 56 à 59 são fechados todas as estruturas de condição “se” para que na linha 60 a variável *fitness* receba o valor da função de *benchmark* de Ackley do membro selecionado.

Com o valor de *fitness* do atual membro gravado, uma nova estrutura condicional “se” é chamada na linha 61 para que compare o valor recém gravado da variável *fitness* com a variável *melhorFitness*, essa condição se torna verdadeira somente se a posição do atual membro selecionado i for melhor que a melhor posição encontrada durante toda a rotina do algoritmo até o momento. Logo, se verdadeiro, o vetor de posição X_{best} receberá a posição atual do membro selecionado (linha 62), a variável *melhorFitness* receberá o

valor da variável *fitness* e a variável *melhorMembro* receberá o valor de *i*, o que indica o *ID* do veículo escolhido até o momento (linha 63).

Após essa última checagem da estrutura condicional “se” (linhas 61 à 64), a variável *i* é iterada para que possa continuar a execução para o próximo membro. Caso tenha se passado todos os membros, então a variável *t* será iterada para que o algoritmo avance no tempo e inicie pelo primeiro membro novamente. Após o encerramento do laço de repetição que controla a variável *t*, o algoritmo finaliza na linha 67 ao retornar para o Líder o *ID* do veículo escolhido, gravado na variável *melhorMembro*, para executar o serviço que está na fila.

Para auxiliar na representação do algoritmo 1, foi elaborada a figura 26 em forma de fluxograma. Nela é possível observar todas as etapas do algoritmo, os principais laços de repetição, as checagens dos termos condicionais e as conexões entre as suas diferentes partes.

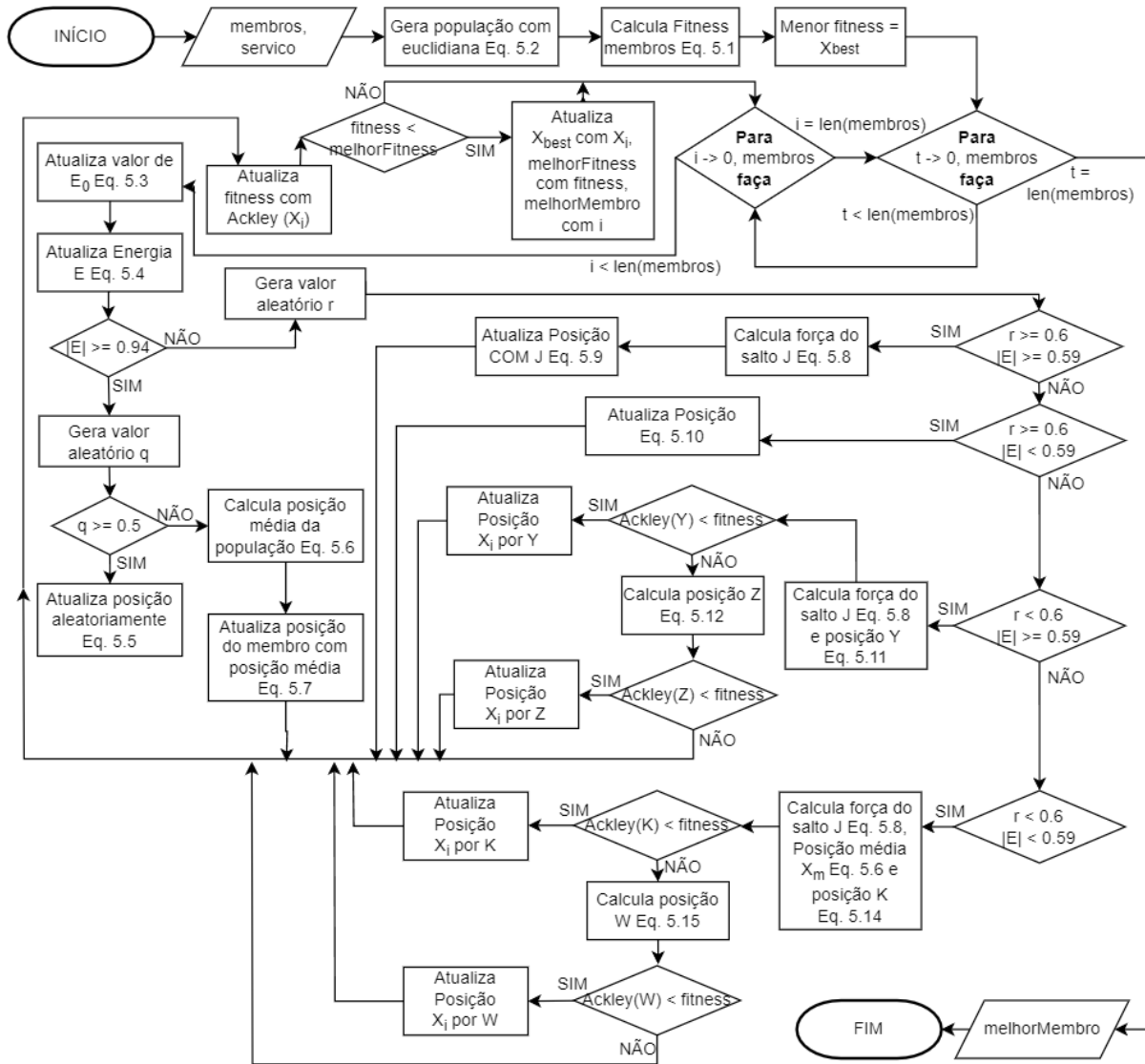
Um ponto observado e alterado durante a adaptação do modelo HHO para a criação do RAPTOR, é a localização dos processos de cálculo de *fitness* e atribuição da posição do melhor membro à X_{best} . Originalmente em (Heidari *et al.*, 2019) esses processos acontecem somente dentro do primeiro laço de repetição “para” no *core* principal do algoritmo, logo no começo do laço e antes da segunda estrutura “para”. Porém foi observado que dessa maneira se perderia a informação da última iteração, pois o cálculo do *fitness* era realizado antes da atualização de posições. Logo, foi decidido que o correto é calcular os valores de *fitness* logo após a criação da população, antes do primeiro laço “para”, e criar uma nova etapa desse cálculo ao final da segunda estrutura de repetição “para”, assim o algoritmo além de ter a oportunidade de atualizar o X_{best} com a última iteração de *t*, terá uma velocidade de convergência maior do que teria originalmente.

5.3 Parâmetros de Simulação

A simulação foi realizada por meio do *framework* de simulação veicular (VEINS - *Vehicular in Network Simulation*) (Sommer; German; Dressler, 2011) na versão 5.1, ele é o responsável por implementar e gerenciar os protocolos de comunicação de rede. O VEINS é um módulo do *framework* de Simulação de Eventos Discretos OMNeT++ (Varga, 2010), que foi utilizado na sua versão 5.6.2 e é responsável por fornecer todo o aparato matemático necessário à execução de uma simulação.

Outra ferramenta utilizada para realizar as simulações, foi o Simulador de Mobilidade Urbana (SUMO - *Simulation of Urban MObility*) (Lopez *et al.*, 2018) em sua versão 0.32.0. Nele é possível implementar o mapa escolhido para ser utilizado na simulação, assim como as regras de trânsito, sinalizações e a capacidade de locomoção dos veículos.

Figura 26 – Fluxograma do algoritmo RAPTOR.



Fonte: Produzido pelo autor.

O cenário escolhido para realizar as simulações foi o LuST (Luxembourg SUMO Traffic) (Codeca *et al.*, 2017). Nele, os pesquisadores utilizaram a cidade europeia de Luxemburgo - LU para mapear a mobilidade (*Traci*) real do trânsito de forma que a comunidade científica tenha um cenário padronizado e que possam utilizar desse conhecimento para desenvolver e testar em nível de realidade suas pesquisas nas área de ITS, infra-estruturas de comunicação e redes veiculares (Codeca *et al.*, 2017).

No LuST, além do mapa da cidade, também são carregados os arquivos de configuração sobre a mobilidade real da cidade, as linhas e paradas de ônibus e os polígonos das construções. Estão mapeados 24 horas de tráfego veicular, com transporte privado, público e temporização semafórica (Codeca *et al.*, 2017). Dentre as quatro possibilidades de configuração para executar o cenário, a escolhida se da pelo arquivo de configuração “*dua.static.sumocfg*”. Nele a rota selecionada pelos veículos será de menor caminho com

reencaminhamento e semáforos com temporização fixa.

O mapa de Luxemburgo é representado por meio da Figura 27, extraída diretamente do ferramenta SUMO e representa a área aproximada de 156 km^2 que a cidade possui.

Figura 27 – Cenário utilizado na simulação, mapa da cidade de Luxemburgo - LU.



Fonte: SUMO.

Outra configuração que precisou ser definida é o momento do dia no cenário em que a simulação ocorreria. Após a realização de alguns testes de carga na máquina virtual que foi utilizada para realizar as simulações, foram medidas as capacidades de processamento e tempo de execução da simulação, e foi definido que a hora de início da simulação no cenário seria as 06:00 com duração de 500 segundos.

O protocolo de rede utilizado foi o IEEE 802.11p (Arena; Pau; Severino, 2020), desenvolvido prioritariamente para comunicação veicular. O VEINS é o responsável por carregá-lo e gerenciá-lo durante a simulação..

A política de eleição de Líder implementada foi o Algoritmo de Menor ID (LID - *Lower ID Algorithm*), que elege como Líder da VEC o *Edge Node* com menor v_{id} (Nguyen

et al., 2015). A cada segundo na simulação, cada veículo do cenário envia uma mensagem de *beacon* que é recebida por todos os veículos dentro do raio de comunicação, logo, os veículos receptores são adicionados à tabela de vizinhança do remetente desse *beacon*. Cada veículo na simulação tem a sua própria tabela de vizinhança.

A formação da *Edge* se dá pelos vizinhos de cada veículo. Logo, essa *Edge* precisa de um Líder, então para que um veículo se torne Líder, ele necessita primeiramente verificar se o seu v_{id} é o menor ID dentre ele e seus vizinhos, caso ele seja, ele está habilitado a se tornar Líder daquela *Edge*. Uma verificação adicional é necessária para checar seu v_{id} é igual ao do atual Líder da *Edge*, se positivo, o nó transmite uma mensagem a toda sua tabela de vizinhança para informar que ele é o Líder. Porém, se ele não for o Líder designado, ele se tornará por meio da formação de uma nova *Edge* e transmissão uma mensagem para informar a toda sua vizinhança que ele é o Líder.

Enquanto isso, os *Edge Nodes* ao receberem uma mensagem de Líder, definida por um ID específico, verificam se possuem um Líder, caso negativo, ele se juntará a *Edge* do remetente da mensagem. Caso positivo, ele irá verificar se o id do remetente da mensagem de Líder é menor que o id do Líder da *Edge* que ele participa, caso seja, então significa que recebeu uma mensagem de outro Líder. Adicionalmente, se o *Edge Node* que recebeu essa mensagem de Líder for um Líder, significa que sua *Edge* se dissolveu e então ele se juntará a nova *Edge* formada, agora apenas como membro.

Como métricas de simulação, foram definidas três diferentes classes de serviços e três diferentes volumes de sua chegada ao Líder da *Edge*. Isso ajuda a testar de forma mais intensa o cenário de simulação e os algoritmos. Abaixo se encontram as três classes de serviços utilizadas nas simulações:

- **Baixo:** São serviços como monitoramento de tráfego e de sinais vitais dos passageiros, considerados de baixo custo operacional por necessitarem de baixa responsividade do sistema;
- **Médio:** São serviços de troca de mensagens, sugestão de rotas e cruzamento de referências, como um serviço de recomendações ou sugestão de lugares. Eles possuem requisitos operacionais de nível intermediário por necessitarem de maior prioridade de execução e responsividade do sistema;
- **Alto:** São os serviços com maior custo computacional, pois precisam de um fluxo contínuo de informações além de processamento constante, como por exemplo serviços de entretenimento de áudio/vídeo.

A outra métrica de simulação adotada é a variação na chegada de serviços ao Líder, foi utilizado uma distribuição de Poisson para gerenciar esse volume, em que três

diferentes valores para λ são atribuídos: 1, 5 e 10. Em que $\lambda = 1$ representa uma cadência menor na chegada de serviços ao Líder, e $\lambda = 10$ representa uma cadência maior nessa chegada. As simulações foram combinadas para que cada cenário tivesse uma classe de serviço e um valor de λ , ou seja, foram elaborados um total de nove cenários para testar cada um dos algoritmos.

Na Tabela 3 é possível observar todos os parâmetros definidos para a execução das simulações.

Tabela 3 – Parâmetros definidos para a simulação

Parâmetro	Valor
Chegada do Serviço	Distribuição de Poisson = 1, 5, 10
Classe de Serviço: Baixo	s_{ar} (1 ~ 3) s_{pr} (1 ~ 3) s_{me} (1 ~ 3)
Classe de Serviço: Médio	s_{ar} (4 ~ 6) s_{pr} (4 ~ 6) s_{me} (4 ~ 6)
Classe de Serviço: Alto	s_{ar} (7 ~ 10) s_{pr} (7 ~ 10) s_{me} (7 ~ 10)
Recursos dos Veículos	v_{ar} (1 ~ 10) v_{pr} (1 ~ 10) v_{me} (1 ~ 10)
Número de Veículos	≈ 2500
Protocolo de Comunicação	IEEE 802.11p
Distância Máxima de Comunicação	400m
Potência de Transmissão	20mW
Política de Eleição do Líder	LID
Tipo de cenário	LuST
Tamanho do Cenário	156 Km^2
Número de Rodadas	33 vezes
Hora de Início da Simulação	06:00
Tempo de Simulação	500 segundos

Fonte: Produzido pelo autor.

Para comparação do RAPTOR, outros cinco algoritmos da literatura foram selecionados, são eles: Otimização do Falcão Harris (HHO) (Gottsfritz *et al.*, 2023a), Otimização por Enxame de Partículas (PSO) (Faris *et al.*, 2016) (adaptado pelo autor), Otimizador do Lobo Cinzento (GWO) (Ladeira *et al.*, 2021), Algoritmo de Otimização da Baleia (WOA) (Faris *et al.*, 2016) (adaptado pelo autor) e Processo Hierárquico Analítico (AHP) (Pereira *et al.*, 2019).

Para cada algoritmo, em cada combinação de parâmetros, foram realizadas 33 execuções, e calculadas as médias aritméticas dos resultados para possa ser aplicado o intervalo de confiança de 95% por meio da tabela “*t-Student*”, conforme aponta Pearson e Wishart (1942). Foram utilizadas três métricas para avaliação:

- **Serviços alocados:** é a quantidade de serviços em que o *Edge Node* escolhido pelo algoritmo possui todos os recursos necessários para realizar a sua execução.

Adicionalmente, para que um serviço seja considerado como alocado, o Líder da VEC que solicitou a execução ao *Edge Node*, deve obter uma resposta positiva dele;

- **Serviços não alocados:** é a soma da quantidade dos serviços negados, aqueles que o *Edge Node* selecionado não foi capaz de executar o serviço dada a insuficiência de recursos; com os serviços perdidos, que são aqueles em que o Líder não pode verificar seu sucesso ou fracasso por falta de comunicação com o *Edge Node* escolhido.
- **Quantidade de recursos:** é a soma de todas as unidades de recursos de armazenamento (s_{ar}), processamento (s_{pr}) e memória (s_{me}) dos serviços alocados. Por exemplo, caso um serviço com requisitos 2 de processamento, 2 de memória e 2 de armazenamento for alocado, o valor de “Quantidade de Recursos” será acrescido em 6 pontos, que é a soma de todos eles.

5.4 Resultados

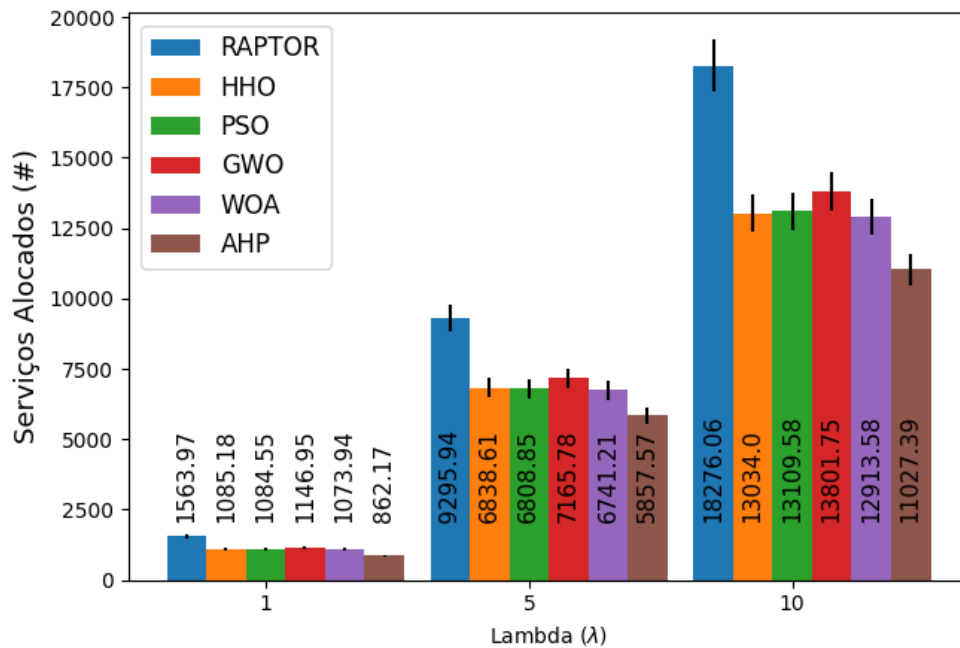
Nesta seção são apresentados os resultados obtidos a partir das simulações, elas utilizaram todos os parâmetros, métricas e algoritmos de comparação apresentados na seção 5.3, contra o algoritmo apresentado na seção 5.2 dentro da estrutura definida na seção 5.1.

Foram variados dois tipos diferentes de parâmetros: classes de serviços e volume de chegada desses serviços (por meio da variação de λ); para cada uma das três métricas definidas: Serviços Alocados, Serviços Não Alocados e Quantidade de Recursos. Logo, cada Figura que será apresentada é de uma “métrica” com uma “classe de serviços”, e cada gráfico dessa figura, representa um volume de chegada desses serviços (λ).

O cenário com classe de serviços “Baixo” é representado pelas Figuras 28, 29 e 30. Por meio da Figura 28, pode-se observar a quantidade de serviços corretamente alocados em cada uma das três variações dos valores de λ . Adicionalmente é possível afirmar que o RAPTOR (em azul) teve uma melhor performance que seus concorrentes, em todas as variações. Com $\lambda = 1$, o RAPTOR conseguiu cerca de 36,35% mais serviços alocados que o segundo colocado, o GWO no caso, seguido por HHO, PSO, WOA e AHP, em que o AHP conta com a pior performance. Uma diferença um pouco menor ocorre no cenário de $\lambda = 5$, em que o RAPTOR performou aproximadamente 29,72% melhor que o GWO, novamente segundo colocado, seguido pela mesma ordem da primeira variação. Para o último cenário, de “Serviços Alocados”, com a classe de serviços “Baixo”, com $\lambda = 10$, o RAPTOR ultrapassou o segundo colocado, GWO mais uma vez, em 32,41%, seguido por PSO, HHO, WOA e AHP.

Complementarmente à Figura 28, na Figura 29 é possível observar a quantidade média de serviços não alocados, aqueles em que o veículo escolhido não responde de volta

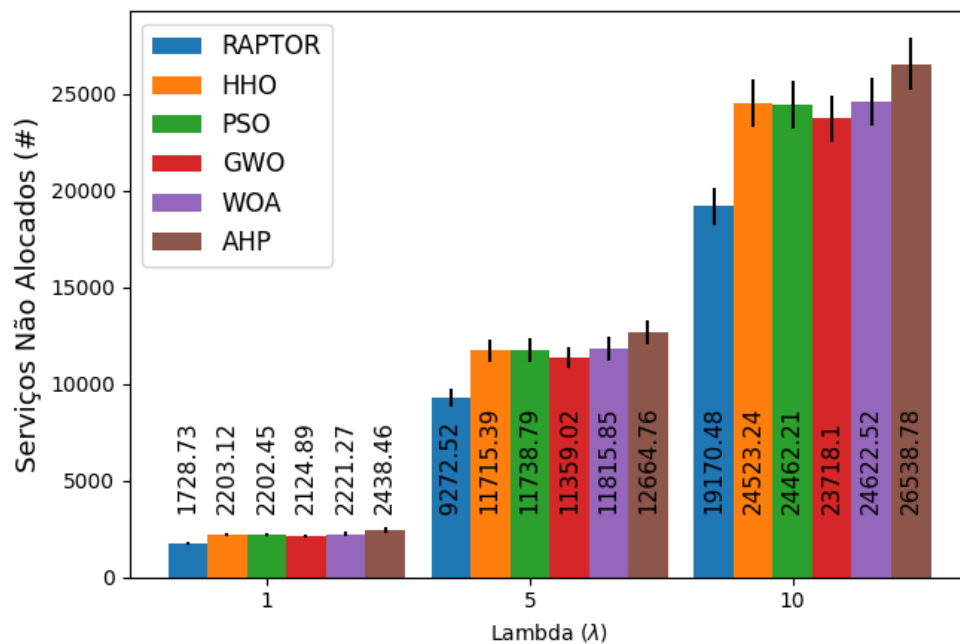
Figura 28 – Gráfico de Serviços Alocados com Classe de Serviços Baixo



Fonte: Produzido pelo autor.

ao Líder sobre a execução do serviço ou em que o veículo não têm recursos suficientes para atender aos requisitos do serviço. Em $\lambda = 1$, o RAPTOR teve 18,64% menos falhas em alocações de serviços, contra o GWO. Para $\lambda = 5$, o RAPTOR conseguiu 18,36% menos erros que o segundo colocado e para $\lambda = 10$ essa diferença foi de 19,17%.

Figura 29 – Gráfico de Serviços Não Alocados com Classe de Serviços Baixo

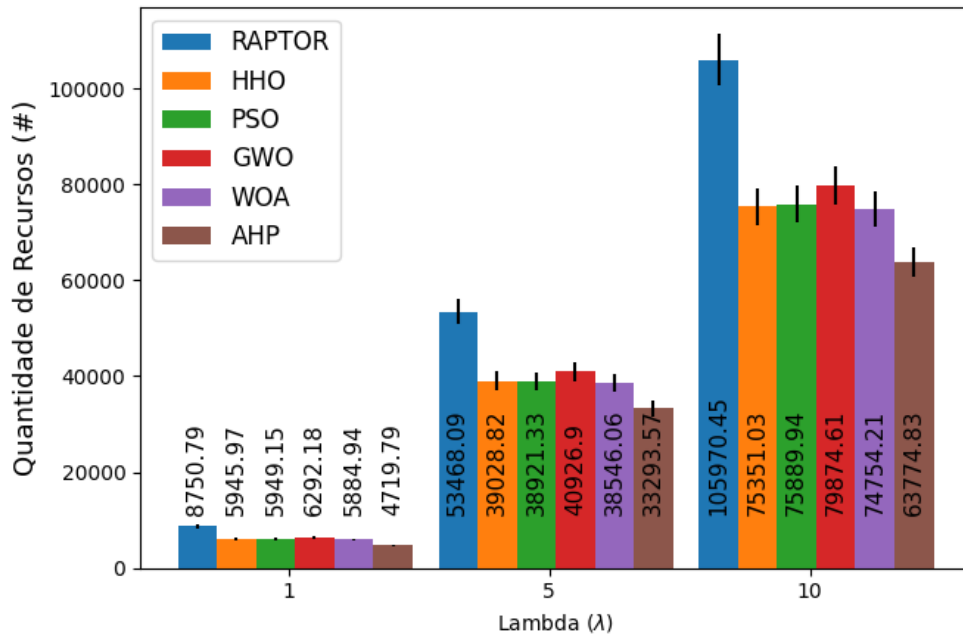


Fonte: Produzido pelo autor.

Na Figura 30, é apresentado a quantidade de recursos que foram alocados por

meio dos serviços atendidos representados na Figura 28. Para $\lambda = 1$, o RAPTOR foi capaz de alocar 39,07% mais recursos que o seu concorrente direto, o GWO. No cenário de $\lambda = 5$, foram 30,64% mais recursos alocados que o segundo colocado. E por fim, com $\lambda = 10$, foram cerca de 32,67% mais alocações de recursos para o RAPTOR em relação ao GWO, seu concorrente direto nos três gráficos apresentados.

Figura 30 – Gráfico de Quantidade de Recursos com Classe de Serviços Baixo



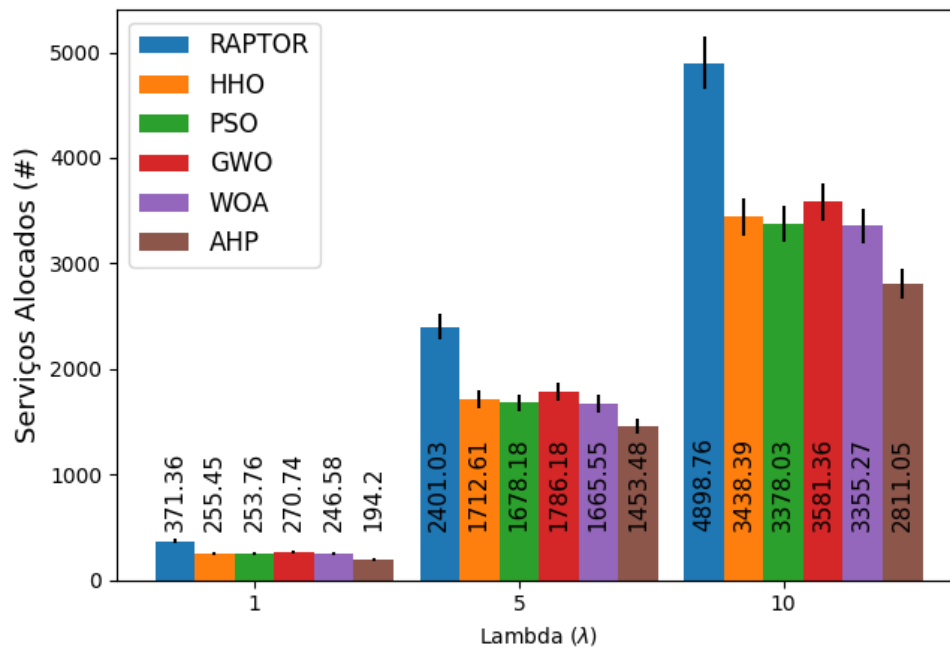
Fonte: Produzido pelo autor.

Por meio das Figuras 31, 32 e 33 são apresentadas os dados das simulações para a classe de serviços “Médio”. Na Figura 31 é possível observar a superioridade do RAPTOR também nessa classe de serviços. Com $\lambda = 1$ ele foi capaz de alocar 37,16% mais serviços que o GWO, o segundo colocado. Para $\lambda = 5$ essa diferença cai para 34,42% mais serviços alocados para o RAPTOR, e para $\lambda = 10$ a diferença fica em 36,78%. Com relação ao algoritmo dito tradicional, que teve o pior desempenho em todos os quesitos, as diferenças são de 91,22%, 65,19% e 74,26% respectivamente.

Na Figura 32, observamos o gráfico complementar à Figura 31. Com $\lambda = 1$ o RAPTOR alocou de forma errônea cerca de 2,96% menos serviços em relação ao GWO. Para $\lambda = 5$, a diferença foi para 3,52% contra o segundo colocado. E por fim, para $\lambda = 10$, o RAPTOR errou menos 3,80% das alocações contra o GWO novamente.

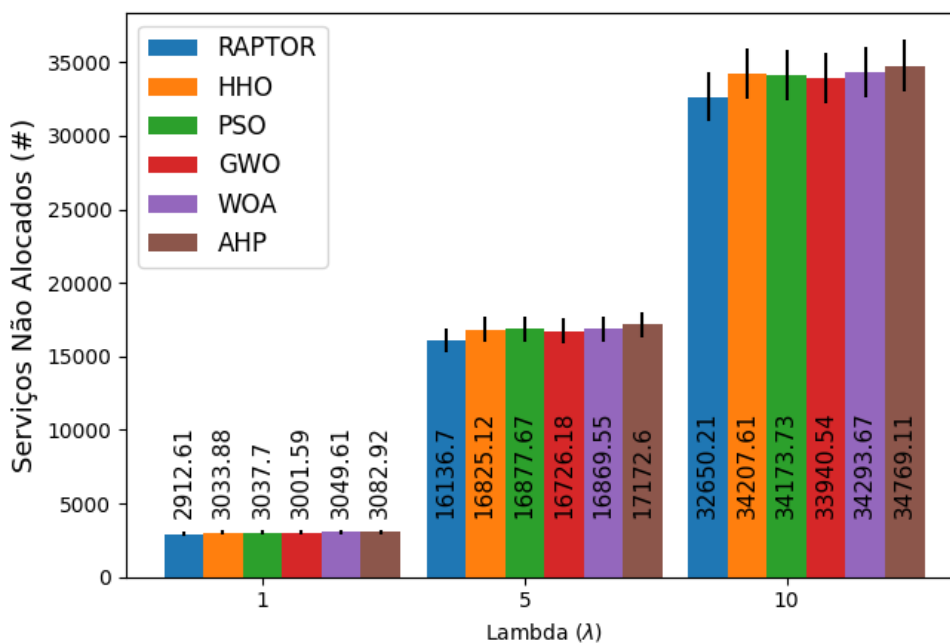
Por meio da Figura 33, é possível observar a superioridade em alocações, agora de recursos, indicada na Figura 31. Com $\lambda = 1$, o RAPTOR alocou cerca de 36,76% mais recursos ociosos que seu concorrente direto, o GWO. Em $\lambda = 5$, essa diferença recuou para 33,89% contra o segundo colocado. E para $\lambda = 10$, a diferença voltou a crescer e atingiu 36,57% mais recursos alocados nas simulações em que o RAPTOR foi utilizado. Para a classe de serviços “Médio”, assim como na classe de serviços “Baixo”, a segunda

Figura 31 – Gráfico de Serviços Alocados com Classe de Serviços Médio



Fonte: Produzido pelo autor.

Figura 32 – Gráfico de Serviços Não Alocados com Classe de Serviços Médio

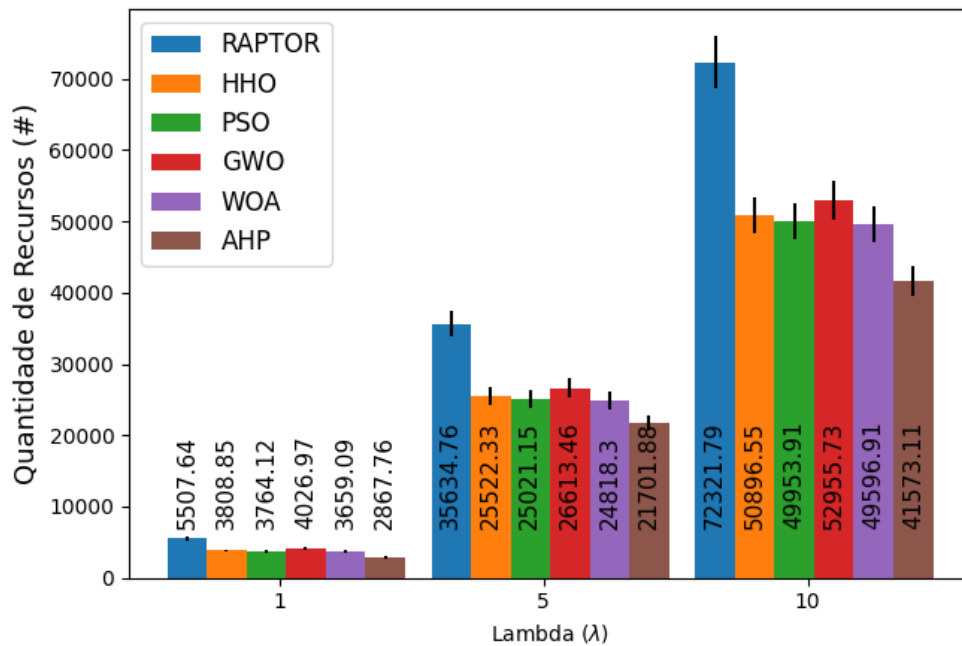


Fonte: Produzido pelo autor.

posição é predominante para o GWO, porém com margens bem próximas ao HHO, que desaparecem se considerado o intervalo de confiança.

Nas Figuras 34, 35 e 36, são apresentados os dados das simulações que contém a classe de serviços “Alto” e seus respectivos valores de lambda. Com o valor de $\lambda = 1$, o RAPTOR alocou corretamente apenas 6,25% mais serviços que o segundo colocado,

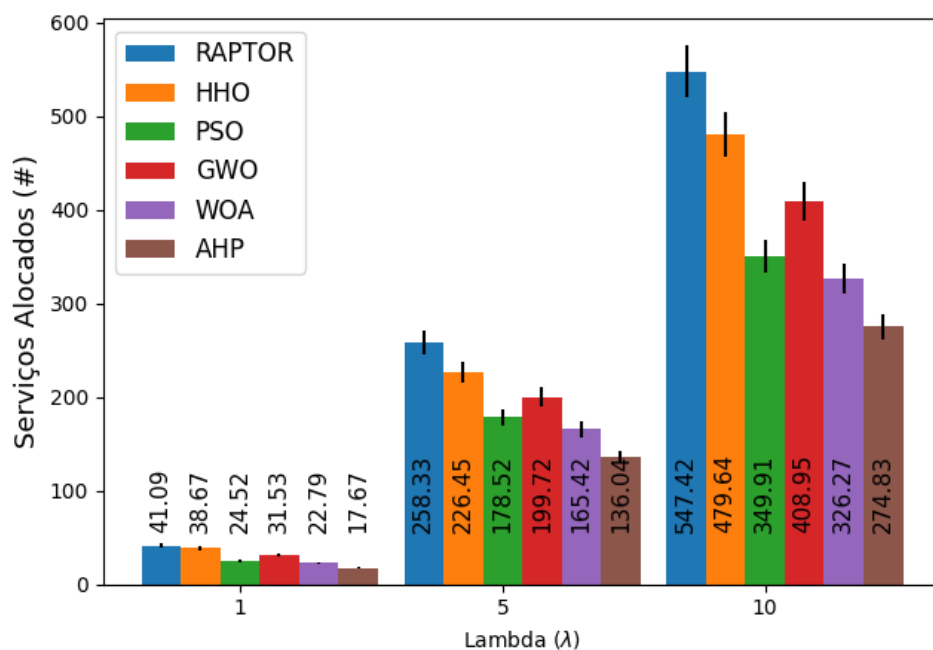
Figura 33 – Gráfico de Quantidade de Recursos com Classe de Serviços Médio



Fonte: Produzido pelo autor.

o HHO. Para $\lambda = 5$ essa diferença cresceu para 14,07%. E com $\lambda = 10$ essa diferença se manteve, em que o RAPTOR alocou 14,13% mais serviços que o HHO, que foi seu concorrente direto em todos esses casos. Se comparado com o AHP, que obteve a pior performance em todos os três valores de lambda, o RAPTOR conseguiu 132,54%, 89,89% e 99,18% mais serviços atendidos, respectivamente.

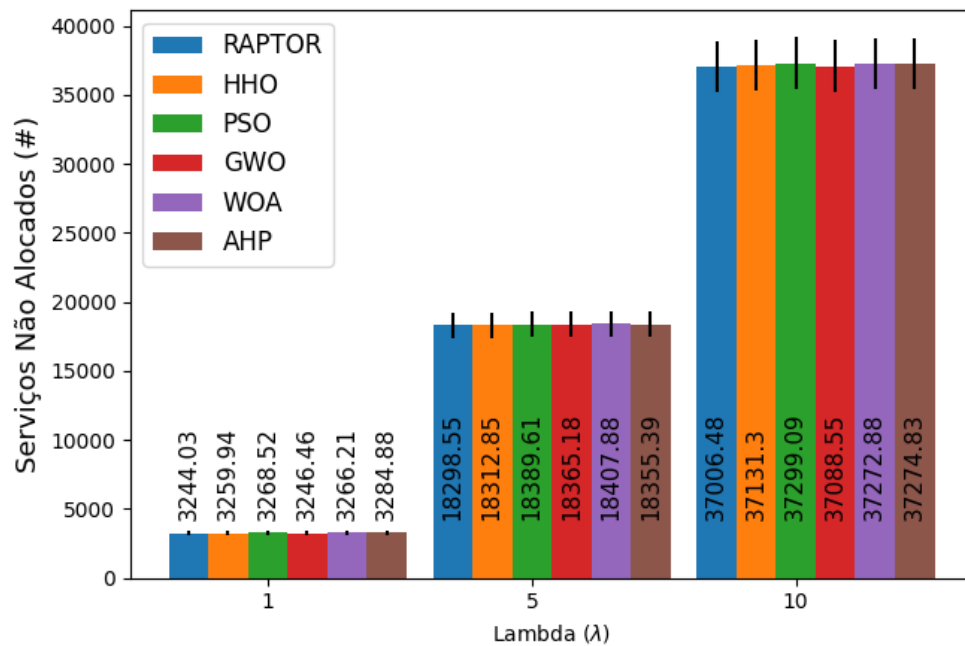
Figura 34 – Gráfico de Serviços Alocados com Classe de Serviços Alto



Fonte: Produzido pelo autor.

Por meio da Figura 35, observamos o gráfico complementar do de “Serviços Alocados”, “Serviços Não Alocados”. Com diferenças mínimas, para $\lambda = 1$, o RAPTOR falhou 0,07% menos que o GWO, segundo colocado aqui; também 0,07% em $\lambda = 5$ agora contra o HHO que foi o segundo colocado; e 0,22% em $\lambda = 10$ contra o GWO novamente. É importante observar a alternância entre o algoritmo que está em segundo lugar, isso se dá pela restrição do cenário que projeta um alto volume de “Serviços Não Alocados” e deixa em evidência as pequenas diferenças de quantidade máxima de serviços gerados na simulação.

Figura 35 – Gráfico de Serviços Não Alocados com Classe de Serviços Alto

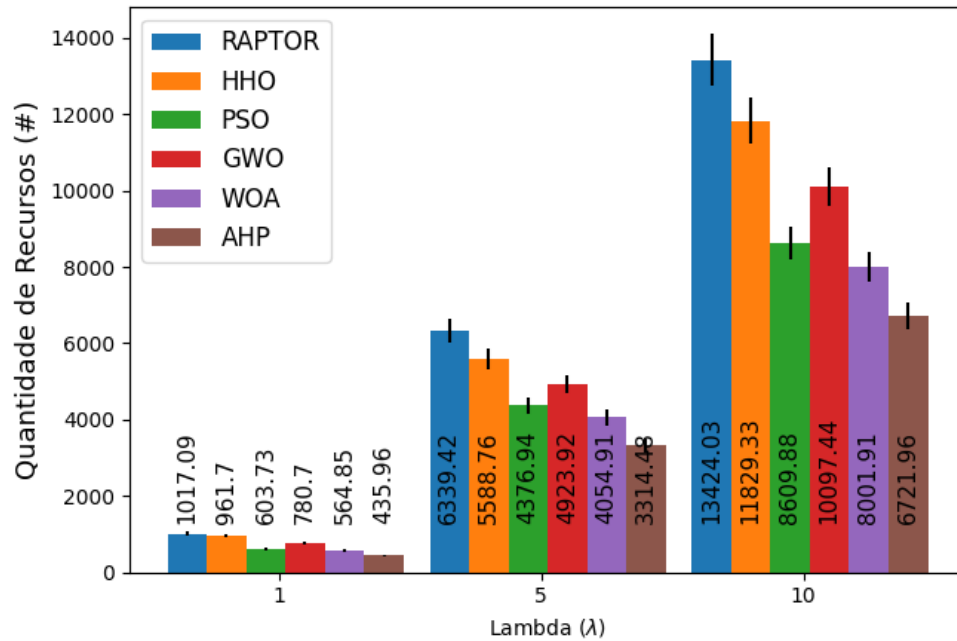


Fonte: Produzido pelo autor.

Para a Figura 36, é reforçado os resultados obtidos a partir do “Gráfico de Serviços Alocados”, em que o RAPTOR obteve o melhor desempenho em todos os valores de lambda com o HHO em segundo colocado. Com $\lambda = 1$ sua performance foi 5,75% melhor que do segundo colocado. Para $\lambda = 5$, essa diferença sobe para 13,43%, e para $\lambda = 10$ a diferença é mantida com o valor de 13,48% mais recursos alocados para o RAPTOR.

De forma geral, é possível observar o melhor desempenho do algoritmo RAPTOR em todos os cenários apresentados. A menor diferença se encontra dentro da classe de serviços “Alto” devido a restrição imposta pelo cenário. Ao considerar todas as representações da métrica de “Serviços Atendidos” (Figuras 28, 31, 34), a única situação em que o intervalo de confiança declara um empate é para a classe de serviços “Alto” em $\lambda = 1$, em todos os demais cenários o RAPTOR obteve uma performance superior. O mesmo se aplica à métrica “Quantidade de Recursos” (Figuras 30, 33, 36), pois essa métrica acompanha proporcionalmente a de “Serviços Atendidos”.

Figura 36 – Gráfico de Quantidade de Recursos com Classe de Serviços Alto



Fonte: Produzido pelo autor.

O que também fica evidente, é a superioridade dos algoritmos meta-heurísticos em relação ao tradicional AHP. Mesmo que sozinho para representar a categoria de algoritmos tradicionais, ele não conseguiu superar nenhum dos métodos meta-heurísticos, inclusive os adaptados unicamente para esses testes, como o PSO e WOA. Essa superioridade se deve ao uso do valor de *fitness* como balizador da rotina iterativa de aprimoramento da população, inerente de todos os meta-heurísticos de Inteligência de Enxame, como explicado anteriormente.

Outro ponto observado, é o RAPTOR considerar os recursos ociosos de todos os membros da VEC e também os requisitos mínimos para a execução dos serviços, com o uso da distância euclidiana para inicializar a população no algoritmo. O seu uso teve um impacto positivo nos resultados durante a fase de projeto e testes do RAPTOR, por isso foi mantido.

O outro ponto a ser lembrado é a capacidade de adaptação que o HHO proporciona ao RAPTOR, as duas diferentes estratégias na fase de exploração e quatro diferentes estratégias na fase de aproveitamento abrangem a volatilidade de uma rede veicular, e com a correta configuração de suas variáveis balizadoras (q , $|E|$ e r), somada as alterações e melhorias estruturais do algoritmo, o RAPTOR foi capaz de manter uma melhor performance em todas as possibilidades aplicadas, contra todos os seus concorrentes, em um cenário de mobilidade real.

O algoritmo proposto, RAPTOR, foi comparado a outros quatro algoritmos meta-heurísticos (HHO, PSO, GWO e WOA) e a um tradicional (AHP), com a utilização de um

mapa com mobilidade (*TraCI*) real para as simulações. Como comparação, foram preparados três diferentes valores de λ , que impactam diretamente na chegada de requisições de serviços ao Líder, e três diferentes classes de serviços. A combinação desses dois meios de comparação, criaram nove diferentes complexidades para que os algoritmos pudessem ser avaliados em diferentes intensidades durante a simulação, o que reforçou a superioridade do algoritmo proposto.

6 CONCLUSÃO

No estudo realizado sobre os métodos de alocação de recursos em Computação de Borda Veicular, foi apresentado o estado da arte de métodos meta-heurísticos baseados em Inteligência de Enxame. Também foi explorado o ambiente de Computação de Borda, Computação de Borda de Múltiplos Acessos e Computação de Borda Veicular, que mostrou serem áreas promissoras para o desenvolvimento de pesquisas. A crescente demanda por serviços de rede faz com que as estruturas existentes sejam aprimoradas e métodos mais eficientes sejam desenvolvidos.

Com o progresso das tecnologias que envolvem a área veicular, especificamente área de ITS, têm-se as redes veiculares capazes de ofertar serviços para melhorar a qualidade de vida e reduzir o impacto ambiental e socioeconômico. Porém, parte dos desafios encontrados na prestação desses serviços incorrem da alocação de recursos computacionais, pois tendem a não possuir abundância além de serem impactados por outros problemas conhecidos desse tipo de rede, como sua dinamicidade, baixa escalabilidade e alto tempo de resposta. Mesmo que pequenos, os constantes avanços são necessários nessa área da pesquisa, para que futuramente eles possam compor algo maior e mais bem definido.

É com esse pensamento que este trabalho propõe o RAPTOR, um algoritmo meta-heurístico inspirado no método da Otimização do Falcão Harris que auxilia o Líder da VEC a alocar de maneira mais assertiva os recursos ociosos de um veículo membro (*Edge Node*), para atendimento de um serviço. Em comparação com outros algoritmos da literatura, HHO, PSO, GWO, WOA e AHP, o RAPTOR teve uma melhor performance em todas as métricas estabelecidas, “Serviços Alocados”, “Serviços Não Alocados” e “Quantidade de Recursos”, em todos os tipos de classes de serviços e para os três valores diferentes de λ que foram testados.

Essa superioridade se deu principalmente para as classes de serviços “Baixo” e “Médio”, em que a diferença para o segundo colocado chegou a ultrapassar 35% em alguns casos. Para a classe de serviços “Alto”, a diferença foi bem mais estreita, devido a natureza restritiva do teste, em que temo-se serviços com requisitos de Armazenamento, Processamento e Memória com valores entre 7 e 10, enquanto os veículos tinham cada um desses recursos com valores entre 1 e 10. Ou seja, não se tem garantia de que na VEC existirá algum veículo capaz de atender à essa demanda.

6.1 Principais contribuições

Nesta seção, as principais contribuições alcançadas durante a execução deste projeto estão listadas:

- Desenvolvimento de um algoritmo baseado em um método meta-heurístico para gerenciar o processo de tomada de decisão durante a alocação de recursos em um ambiente de Computação de Borda Veicular sem infraestrutura;
- Implantar um ambiente de simulação por meio com o uso de um simulador de redes veiculares, juntamente com o uso de um cenário com mobilidade real;
- Aumentar a quantidade de recursos alocados por meio da maximização da quantidade de serviços atendidos e, automaticamente, redução da quantidade de serviços não atendidos;

6.2 Produção Científica

- Gottsfritz *et al.* (2023b). “Marvel: Um algoritmo meta-heurístico para alocação de recursos em redes veiculares ad-hoc. In: Anais do XXVIII Workshop de Gerência e Operação de Redes e Serviços. Porto Alegre, RS, Brasil: SBC, 2023. p. 15–28. ISSN 2595-2722” (Menção Honrosa)
- Gottsfritz *et al.* (2023a). “Maltreat - a metaheuristic algorithm for allocation of resources in vehicular ad-hoc networks. In: *2023 19th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2023.”
- Guidoni, D. L.; Gottsfritz, E. N.; Meneguetto, R. I.; Silva, C. M.; Filho, G. P. R.; Souza, F. S. H. “Toward an efficient data dissemination protocol for vehicular ad-hoc networks. *IEEE Access*, v. 10, p. 123711–123722, 2022.”

6.3 Trabalhos Futuros

Por fim, como direções futuras, pretende-se implementar diferentes formas de eleição do Líder e também implementar RSUs a fim de comparar o desempenho com e sem estrutura de beira de estrada. Pretende-se também, adicionar o tempo de execução e gasto de recursos computacionais em seu funcionamento, como meio de avaliação de desempenho, assim como um sistema de recolocação de “Serviços Não Alocados”. Outro ponto a ser desenvolvido é a utilização de outros protocolos de comunicação, como o 5G e 6G, por exemplo.

REFERÊNCIAS

- ABEDINPOURSHOTORBAN, H. *et al.* Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm. *Swarm and Evolutionary Computation*, v. 26, p. 8–22, 2016. ISSN 2210-6502. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2210650215000528>.
- ALTABEEB, A. M. *et al.* Solving capacitated vehicle routing problem using cooperative firefly algorithm. *Applied Soft Computing*, v. 108, p. 107403, 2021. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494621003264>.
- ARENA, F.; PAU, G.; SEVERINO, A. A review on iee 802.11p for intelligent transportation systems. *Journal of Sensor and Actuator Networks*, v. 9, n. 2, 2020. ISSN 2224-2708. Disponível em: <https://www.mdpi.com/2224-2708/9/2/22>.
- BARTHELEMY, P.; BERTOLOTTI, J.; WIERMSMA, D. S. A lévy flight for light. *Nature*, v. 453, n. 7194, p. 495–498, May 2008. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/nature06948>.
- BHATTACHARYA, A.; CHATTOPADHYAY, P. Solving complex economic load dispatch problems using biogeography-based optimization. *Expert Systems with Applications*, v. 37, n. 5, p. 3605–3615, 2010. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417409008975>.
- BORGWARDT, K. H. *The Simplex Method: a probabilistic analysis*. Springer Science & Business Media, 2012. v. 1. Disponível em: <https://link.springer.com/book/10.1007/978-3-642-61578-8>.
- BRAHMI, I.; HAMDI, M.; ZARAI, F. Chaotic grey wolf optimization-based resource allocation for vehicle-to-everything communications. *International Journal of Communication Systems*, v. 34, n. 13, p. e4908, 2021. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4908>.
- CHAHAL, M.; HARIT, S. Optimal path for data dissemination in vehicular ad hoc networks using meta-heuristic. *Computers Electrical Engineering*, v. 76, p. 40–55, 2019. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790618315404>.
- CHAKRABORTY, A.; KAR, A. K. Swarm intelligence: A review of algorithms. In: PATNAIK, S.; YANG, X.-S.; NAKAMATSU, K. (Ed.). *Nature-Inspired Computing and Optimization: Theory and Applications*. Cham: Springer International Publishing, 2017. p. 475–494. ISBN 978-3-319-50920-4. Disponível em: https://doi.org/10.1007/978-3-319-50920-4_19.
- CODECA, L. *et al.* Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation. *IEEE Intelligent Transportation Systems Magazine*, v. 9, n. 2, p. 52–63, 2017.
- COMPUTING, E. M. E.; INITIATIVE, I. *et al.* Mobile-edge computing: introductory technical white paper. *ETSI: Sophia Antipolis, France*, p. 1–36, 2014.

- CRONAPP, R. *Edge computing: Saiba o que é a tendência e por que é relevante*. 2020. Acessado em: 11 de nov. de 2021. Disponível em: <https://blog.cronapp.io/edge-computing/>.
- CUI, Z.; LI, F.; ZHANG, W. Bat algorithm with principal component analysis. *International Journal of Machine Learning and Cybernetics*, v. 10, n. 3, p. 603–622, Mar 2019. ISSN 1868-808X. Disponível em: <https://doi.org/10.1007/s13042-018-0888-4>.
- DEHKORDI, A. A. *et al.* Nonlinear-based chaotic harris hawks optimizer: Algorithm and internet of vehicles application. *Applied Soft Computing*, v. 109, p. 107574, 2021. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494621004956>.
- DIVYA, R.; JAYANTHI, V. E. Efficient optimal resource allocation for profit maximization in software defined network approach to improve quality of service in cloud environments. *Journal of Ambient Intelligence and Humanized Computing*, Jun 2020. ISSN 1868-5145. Disponível em: <https://doi.org/10.1007/s12652-020-02192-8>.
- 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.
- EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. IEEE, 1995. p. 39–43. Disponível em: <https://ieeexplore.ieee.org/document/494249>.
- ELER, G. *Vídeo inédito flagra jubartes usando “rede de bolhas” para caçar peixes*. 2019. Acessado em: 03 de nov. de 2021. Disponível em: <https://super.abril.com.br/ciencia/video-inedito-flagra-jubartes-usando-rede-de-bolhas-para-cacar-peixes/>.
- FARIS, H. *et al.* Grey wolf optimizer: a review of recent variants and applications. *Neural Computing and Applications*, v. 30, n. 2, p. 413–435, Jul 2018. ISSN 1433-3058. Disponível em: <https://doi.org/10.1007/s00521-017-3272-5>.
- FARIS, H. *et al.* EvoloPy: An open-source nature-inspired optimization framework in python. In: *Proceedings of the 8th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science and Technology Publications, 2016. Disponível em: <https://doi.org/10.5220/0006048201710177>.
- FISTER, I. *et al.* Bat algorithm: Recent advances. In: *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*. IEEE, 2014. p. 163–167. Disponível em: <https://ieeexplore.ieee.org/document/7028669>.
- 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>.
- GOLDEN, B. L.; WASIL, E. A.; HARKER, P. T. (Ed.). *The Analytic Hierarchy Process*. Springer Berlin Heidelberg, 1989. Disponível em: <https://doi.org/10.1007/978-3-642-50244-6>.

- GOTTSFRITZ, E. *et al.* Maltreat - a metaheuristic algorithm for allocation of resources in vehicular ad-hoc networks. In: *2023 19th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. [S.l.]: IEEE, 2023.
- GOTTSFRITZ, E. *et al.* Marvel: Um algoritmo meta-heurístico para alocação de recursos em redes veiculares ad-hoc. In: *Anais do XXVIII Workshop de Gerência e Operação de Redes e Serviços*. Porto Alegre, RS, Brasil: SBC, 2023. p. 15–28. ISSN 2595-2722. Disponível em: <https://sol.sbc.org.br/index.php/wgrs/article/view/24667>.
- GUIDONI, D. L. *et al.* Toward an efficient data dissemination protocol for vehicular ad-hoc networks. *IEEE Access*, v. 10, p. 123711–123722, 2022.
- HAO, W. A gradient descent method for solving a system of nonlinear equations. *Applied Mathematics Letters*, v. 112, p. 106739, 2021. ISSN 0893-9659. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893965920303712>.
- HASSAN, N.; YAU, K. A.; WU, C. Edge computing in 5g: A review. *IEEE Access*, v. 7, p. 127276–127289, 2019.
- HASSANIEN, A. E. *et al.* An improved moth flame optimization algorithm based on rough sets for tomato diseases detection. *Computers and Electronics in Agriculture*, v. 136, p. 86–96, 2017. ISSN 0168-1699. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168169916308225>.
- HEIDARI, A. A. *et al.* Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*, v. 97, p. 849–872, 2019. ISSN 0167-739X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167739X18313530>.
- HOLLAND, J. H. *et al.* *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992. Disponível em: <https://ieeexplore.ieee.org/book/6267401>.
- HU, Y. C. *et al.* Mobile edge computing—a key technology towards 5g. *ETSI white paper*, v. 11, n. 11, p. 1–16, 2015.
- HUSSAIN, K. *et al.* Common benchmark functions for metaheuristic evaluation: A review. In: *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION*. Politeknik Negeri Padang, 2017. v. 1, n. 4-2, p. 218–223. Disponível em: <https://doi.org/10.30630/joiv.1.4-2.65>.
- IEA. *Global CO2 emissions by sector, 2019-2022*. 2023. Disponível em: <https://www.iea.org/data-and-statistics/charts/global-co2-emissions-by-sector-2019-2022>.
- INGBER, L. Very fast simulated re-annealing. *Mathematical and Computer Modelling*, v. 12, n. 8, p. 967–973, 1989. ISSN 0895-7177. Disponível em: <https://www.sciencedirect.com/science/article/pii/0895717789902021>.
- ISHTIAQ, A. *et al.* Intelligent clustering using moth flame optimizer for vehicular ad hoc networks. *International Journal of Distributed Sensor Networks*, v. 15, n. 1, p. 1550147718824460, 2019. Disponível em: <https://doi.org/10.1177/1550147718824460>.
- JAISWAL, N. K. *Military Operations Research: Quantitative Decision Making*. Springer, 1997. 209-232 p. Disponível em: <https://doi.org/10.1007/978-1-4615-6275-7>.

- KANG, J. *et al.* Blockchain for secure and efficient data sharing in vehicular edge computing and networks. *IEEE Internet of Things Journal*, v. 6, n. 3, p. 4660–4670, 2019.
- KAUR, K.; SINGH, U.; SALGOTRA, R. An enhanced moth flame optimization. *Neural Computing and Applications*, v. 32, n. 7, p. 2315–2349, Apr 2020. ISSN 1433-3058. Disponível em: <https://doi.org/10.1007/s00521-018-3821-6>.
- KHALILPOURAZARI, S.; PASANDIDEH, S. H. R. Multi-item eoq model with nonlinear unit holding cost and partial backordering: moth-flame optimization algorithm. *Journal of Industrial and Production Engineering*, Taylor Francis, v. 34, n. 1, p. 42–51, 2017. Disponível em: <https://doi.org/10.1080/21681015.2016.1192068>.
- KHAN, W. A. *et al.* A review and comparative study of firefly algorithm and its modified versions. *Optimization Algorithms-Methods and Applications*, IntechOpen, p. 281–313, 2016.
- KUMAR, A.; PANT, S.; RAM, M. System reliability optimization using gray wolf optimizer algorithm. *Quality and Reliability Engineering International*, v. 33, n. 7, p. 1327–1335, 2017. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/qre.2107>.
- KUMAR, V.; KUMAR, D. A systematic review on firefly algorithm: Past, present, and future. *Archives of Computational Methods in Engineering*, v. 28, n. 4, p. 3269–3291, Jun 2021. ISSN 1886-1784. Disponível em: <https://doi.org/10.1007/s11831-020-09498-y>.
- LAWLER, E. L.; WOOD, D. E. Branch-and-bound methods: A survey. *Operations Research*, v. 14, n. 4, p. 699–719, 1966. Disponível em: <https://doi.org/10.1287/opre.14.4.699>.
- LI, X. *et al.* Optimizing resources allocation for fog computing-based internet of things networks. *IEEE Access*, v. 7, p. 64907–64922, 2019.
- LIEIRA, D. D. *et al.* Meta-heuristic mechanism based on whale optimization algorithm for tasks allocation in edge computing. In: *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, 2022. p. 1–6. Disponível em: <https://ieeexplore.ieee.org/document/9820067>.
- LIEIRA, D. D. *et al.* Tovec: Task optimization mechanism for vehicular clouds using meta-heuristic technique. In: *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2021. p. 358–363. Disponível em: <https://ieeexplore.ieee.org/document/9498784>.
- LIEIRA, D. D. *et al.* Mechanism for optimizing resource allocation in vanets based on the pso bio-inspired algorithm. In: *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2022. p. 283–290. Disponível em: <https://ieeexplore.ieee.org/abstract/document/9881752>.
- LIN, H. *et al.* Dynamic service migration in ultra-dense multi-access edge computing network for high-mobility scenarios. *EURASIP Journal on Wireless Communications and Networking*, v. 2020, n. 1, p. 191, Oct 2020. ISSN 1687-1499. Disponível em: <https://doi.org/10.1186/s13638-020-01805-2>.

- LIU, L. *et al.* Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, v. 26, n. 3, p. 1145–1168, Jun 2021. ISSN 1572-8153. Disponível em: <https://doi.org/10.1007/s11036-020-01624-1>.
- LOPEZ, P. A. *et al.* Microscopic traffic simulation using sumo. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. Disponível em: <https://elib.dlr.de/124092/>.
- Lü, X. *et al.* Energy management of hybrid electric vehicles: A review of energy optimization of fuel cell hybrid power system based on genetic algorithm. *Energy Conversion and Management*, v. 205, p. 112474, 2020. ISSN 0196-8904. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0196890420300108>.
- 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.
- MEERSCHAERT, M. M. Chapter 9 - simulation of probability models. In: MEERSCHAERT, M. M. (Ed.). *Mathematical Modeling (Fourth Edition)*. Fourth edition. Boston: Academic Press, 2013. p. 301–358. ISBN 978-0-12-386912-8. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780123869128500099>.
- MENEGUETTE, R. *et al.* Vehicular edge computing: Architecture, resource management, security, and challenges. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 55, n. 1, nov 2021. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3485129>.
- MENEGUETTE, R. I.; GRANDE, R. E. D.; LOUREIRO, A. A. F. *Intelligent Transport System in Smart Cities*. Springer, 2018. v. 1. 182 p. Disponível em: <https://link.springer.com/book/10.1007/978-3-319-93332-0>.
- MENEGUETTE, R. I.; MADEIRA, E. R. M.; BITTENCOURT, L. F. Multi-network packet scheduling based on vehicular ad hoc network applications. In: *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualization management (svm)*. [S.l.: s.n.], 2012. p. 214–218.
- MIDYA, S. *et al.* Pso based optimized resource allocation in three tier cloud architecture for vanet. In: *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE, 2018. p. 12–17. Disponível em: <https://ieeexplore.ieee.org/document/8718723>.
- MIRJALILI, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems*, v. 89, p. 228–249, 2015. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705115002580>.
- 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: <https://www.sciencedirect.com/science/article/pii/S0965997813001853>.
- MOHANAKRISHNAN, U.; RAMAKRISHNAN, B. Mctrp: An energy efficient tree routing protocol for vehicular ad hoc network using genetic whale optimization algorithm. *Wireless Personal Communications*, v. 110, n. 1, p. 185–206, Jan 2020. ISSN 1572-834X. Disponível em: <https://doi.org/10.1007/s11277-019-06720-4>.
- MOODI, Y. *et al.* Using response surface methodology and providing a modified model using whale algorithm for estimating the compressive strength of columns confined with frp sheets. *Construction and Building Materials*, v. 183, p. 163–170, 2018. ISSN 0950-0618. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950061818314727>.
- NALCACI, G.; ERMIS, M. Selective harmonic elimination for three-phase voltage source inverters using whale optimizer algorithm. In: *2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*. IEEE, 2018. p. 1–6. Disponível em: <https://ieeexplore.ieee.org/document/8391290>.
- NEMATOLLAHI, M.; GHAFARI, A.; MIRZAEI, A. Task and resource allocation in the internet of things based on an improved version of the moth-flame optimization algorithm. *Cluster Computing*, Jun 2023. ISSN 1573-7543. Disponível em: <https://doi.org/10.1007/s10586-023-04041-7>.
- NGUYEN, V. *et al.* Application of the lowest-id algorithm in cluster-based tdma system for vanets. In: *2015 International Conference on Information Networking (ICOIN)*. IEEE, 2015. p. 25–30. Disponível em: <https://ieeexplore.ieee.org/document/7057851>.
- OICA. *World Motor Vehicle Production 2022*. 2023. International Organization of Motor Vehicle Manufacturers. Disponível em: <https://www.oica.net/category/production-statistics/2022-statistics/>.
- PEARSON, E. S.; WISHART, J. *"Student's" collected papers*. [S.l.]: Biometrika Office, 1942.
- PEIXOTO, M. *et al.* Fogjam: A fog service for detecting traffic congestion in a continuous data stream VANET. *Ad Hoc Networks*, v. 140, p. 103046, 2023. ISSN 1570-8705. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1570870522002189>.
- PEREIRA, R. S. *et al.* Reliable: Resource allocation mechanism for 5g network using mobile edge computing. *Sensors*, v. 20, n. 19, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/19/5449>.
- PEREIRA, R. S. *et al.* A novel fog-based resource allocation policy for vehicular clouds in the highway environment. In: *2019 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, 2019. p. 1–6. Disponível em: <https://ieeexplore.ieee.org/document/8937912>.
- PIOTROWSKI, A. P. Review of differential evolution population size. *Swarm and Evolutionary Computation*, v. 32, p. 1–24, 2017. ISSN 2210-6502. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2210650216300268>.

- QU, C.; HE, W. A cuckoo search algorithm with complex local search method for solving engineering structural optimization problem. *MATEC Web of Conferences*, v. 40, p. 09009, 2016. Disponível em: <https://doi.org/10.1051/mateconf/20164009009>.
- QU, C. *et al.* Harris hawks optimization with information exchange. *Applied Mathematical Modelling*, v. 84, p. 52–75, 2020. ISSN 0307-904X. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0307904X2030158X>.
- QUESSADA, M. S. *et al.* A bat bio-inspired mechanism for resource allocation in vehicular clouds. In: *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. [S.l.: s.n.], 2021. p. 197–204.
- RAJA, N. S. M.; VISHNUPRIYA, R. Kapur's entropy and cuckoo search algorithm assisted segmentation and analysis of rgb images. *Indian Journal of Science and Technology*, v. 9, n. 17, p. 89936, 2016.
- RAJABIOUN, R. Cuckoo optimization algorithm. *Applied Soft Computing*, v. 11, n. 8, p. 5508–5518, 2011. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494611001670>.
- RAJPUT, N. S. Firefly inspired robust cooperative and secure qos provisioning framework for vehicular networks in multi-service environment. *Vehicular Communications*, v. 34, p. 100423, 2022. ISSN 2214-2096. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214209621000929>.
- RAMAKRISHNAN, B.; SREEDIVYA, S.; SELVI, M. Adaptive routing protocol based on cuckoo search algorithm (arp-cs) for secured vehicular ad hoc network (vanet). *International Journal of computer networks and applications (IJCNA)*, v. 2, n. 4, p. 173–178, 2017.
- RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. Gsa: A gravitational search algorithm. *Information Sciences*, v. 179, n. 13, p. 2232–2248, 2009. ISSN 0020-0255. Special Section on High Order Fuzzy Sets. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025509001200>.
- RASHEDI, E.; RASHEDI, E.; NEZAMABADI-POUR, H. A comprehensive survey on gravitational search algorithm. *Swarm and Evolutionary Computation*, v. 41, p. 141–158, 2018. ISSN 2210-6502. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2210650217303577>.
- RAZA, S. *et al.* A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions. *Wireless Communications and Mobile Computing*, Hindawi, v. 2019, p. 3159762, Feb 2019. ISSN 1530-8669. Disponível em: <https://doi.org/10.1155/2019/3159762>.
- REDDY, G. N.; KUMAR, S. P. Multi objective task scheduling algorithm for cloud computing using whale optimization technique. In: BHATTACHARYYA, P. *et al.* (Ed.). *Smart and Innovative Trends in Next Generation Computing Technologies*. Singapore: Springer Singapore, 2018. p. 286–297. ISBN 978-981-10-8657-1.
- RIBEIRO, P. *Ecolocalização*. 2017. Acessado em: 11 de nov. de 2022. Disponível em: <https://escolaeducacao.com.br/ecolocalizacao/>.

- SAATY, T. L. The analytic hierarchy process—what it is and how it is used. *Mathematical Modelling*, v. 9, n. 3, p. 161–176, 1987. ISSN 0270-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/0270025587904738>.
- SAATY, T. L. Some mathematical concepts of the analytic hierarchy process. *Behaviormetrika*, n. 29, p. 1–9, 1991. Disponível em: https://link.springer.com/content/pdf/10.2333/bhmk.18.29_1.pdf.
- SABELLA, D.; REZNIK, A.; FRAZAO, R. *Multi-Access Edge Computing in Action*. CRC Press, 2019. Disponível em: <https://www.amazon.com.br/Multi-Access-Computing-Action-Dario-Sabella/dp/0367173948>.
- SAYED, G. I. *et al.* Alzheimer’s disease diagnosis based on moth flame optimization. In: PAN, J.-S. *et al.* (Ed.). *Genetic and Evolutionary Computing*. Cham: Springer International Publishing, 2017. p. 298–305. ISBN 978-3-319-48490-7.
- SHEHAB, M. *et al.* Moth–flame optimization algorithm: variants and applications. *Neural Computing and Applications*, v. 32, n. 14, p. 9859–9884, Jul 2020. ISSN 1433-3058. Disponível em: <https://doi.org/10.1007/s00521-019-04570-6>.
- SHEHAB, M.; KHADER, A. T.; AL-BETAR, M. A. A survey on applications and variants of the cuckoo search algorithm. *Applied Soft Computing*, v. 61, p. 1041–1059, 2017. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494617301278>.
- SOLIMAN, G. M. *et al.* A hybrid whale optimization algorithm with adaptive spiral for terrorism prediction (the case of egypt). *European Journal of Scientific Research*, v. 149, n. 2, p. 165–184, 2018.
- SOMMER, C.; GERMAN, R.; DRESSLER, F. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, v. 10, n. 1, p. 3–15, 2011.
- SONG, S. *et al.* Dimension decided harris hawks optimization with gaussian mutation: Balance analysis and diversity patterns. *Knowledge-Based Systems*, v. 215, p. 106425, 2021. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705120305542>.
- STORN, R. Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. *Technical report, International Computer Science Institute*, v. 11, 1995.
- SUDABATTULA, S.; KOWSALYA, M. Optimal allocation of wind based distributed generators in distribution system using cuckoo search algorithm. *Procedia Computer Science*, v. 92, p. 298–304, 2016. ISSN 1877-0509. 2nd International Conference on Intelligent Computing, Communication Convergence, ICC3 2016, 24-25 January 2016, Bhubaneswar, Odisha, India. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050916316064>.
- SUN, J. *et al.* Joint communication and computing resource allocation in vehicular edge computing. *International Journal of Distributed Sensor Networks*, v. 15, n. 3, p. 1550147719837859, 2019. Disponível em: <https://doi.org/10.1177/1550147719837859>.

SUN, X. *et al.* Enhancing the user experience in vehicular edge computing networks: An adaptive resource allocation approach. *IEEE Access*, v. 7, p. 161074–161087, 2019.

TIKHOMIROV, V. M. *Stories about maxima and minima*. Universities Press, 1991. v. 1. Disponível em: <https://www.amazon.com.br/Stories-About-Maxima-Minima-Tikhomirov/dp/0821801651>.

TOFFOLO, T. A. M. *et al.* The sport teams grouping problem. *Annals of Operations Research*, v. 275, n. 1, p. 223–243, Apr 2019. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/s10479-017-2595-z>.

VARGA, A. Omnet++. In: WEHRLE, K.; GÜNEŞ, M.; GROSS, J. (Ed.). *Modeling and Tools for Network Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 35–59. ISBN 978-3-642-12331-3. Disponível em: https://OPTdoi.org/10.1007/978-3-642-12331-3_3.

WANG, D.; TAN, D.; LIU, L. Particle swarm optimization algorithm: an overview. *Soft Computing*, v. 22, n. 2, p. 387–408, Jan 2018. ISSN 1433-7479. Disponível em: <https://doi.org/10.1007/s00500-016-2474-6>.

WEI, L. *et al.* A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. *European Journal of Operational Research*, v. 265, n. 3, p. 843–859, 2018. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S037722171730766X>.

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)*. IEEE, 2019. p. 1–5. Disponível em: <https://ieeexplore.ieee.org/document/8843624>.

YAMAMOTO, T. Historical developments in convergence analysis for newton's and newton-like methods. *Journal of Computational and Applied Mathematics*, Elsevier, v. 124, n. 1-2, p. 1–23, 2000.

YANG, X.-S. Firefly algorithms for multimodal optimization. In: WATANABE, O.; ZEUGMANN, T. (Ed.). *Stochastic Algorithms: Foundations and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 169–178. ISBN 978-3-642-04944-6.

YANG, X.-S. Firefly algorithm. In: *Engineering Optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons, Ltd, 2010. p. 221–230. ISBN 9780470640425. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470640425.ch17>.

YANG, X.-S. *A New Metaheuristic Bat-Inspired Algorithm*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. 65–74 p. ISBN 978-3-642-12538-6. Disponível em: https://doi.org/10.1007/978-3-642-12538-6_6.

YANG, X.-S.; DEB, S. Cuckoo search via lévy flights. In: *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*. IEEE, 2009. p. 210–214. Disponível em: 10.1109/NABIC.2009.5393690.

YANG, X.-S.; KARAMANOGLU, M. Chapter 1 - nature-inspired computation and swarm intelligence: a state-of-the-art overview. In: YANG, X.-S. (Ed.). *Nature-Inspired Computation and Swarm Intelligence*. Academic Press, 2020. p. 3–18. ISBN 978-0-12-819714-1. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128197141000105>.

ZHANG, R.; JIANG, X.; LI, R. Improved decomposition-based multi-objective cuckoo search algorithm for spectrum allocation in cognitive vehicular network. *Physical Communication*, v. 34, p. 301–309, 2019. ISSN 1874-4907. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1874490717303506>.

ZHANG, S.; GAJPAL, Y.; APPADOO, S. S. A meta-heuristic for capacitated green vehicle routing problem. *Annals of Operations Research*, v. 269, n. 1, p. 753–771, Oct 2018. ISSN 1572-9338. Disponível em: <https://doi.org/10.1007/s10479-017-2567-3>.