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

Matheus Sanches Quessada

Mecanismo Meta-heurístico para Alocação de Recursos em Fogs Veiculares

São José do Rio Preto

2022

Matheus Sanches Quessada

Mecanismo Meta-heurístico para Alocação de Recursos em Fogs Veiculares

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

Orientador: Prof. Dr. Rodolfo Ipolito Meneguette

São José do Rio Preto
2022

Q5m

Quessada, Matheus Sanches
Mecanismo meta-heurístico para alocação de recursos em fogs veiculares / Matheus Sanches Quessada. -- São José do Rio Preto, 2022
64 f. : il., tabs., mapas

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

1. Ciência da computação. 2. Computação em fog. 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.

Matheus Sanches Quessada

Mecanismo Meta-heurístico para Alocação de Recursos em Fogs Veiculares

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

Comissão examinadora

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

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

Prof. Dr. José Remo Ferreira Brega
UNESP - Campus Bauru – SP

São José do Rio Preto

11 de novembro 2022

Dedico este trabalho aos meus pais que sempre me apoiaram na jornada para adquirir conhecimento.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus que me iluminou e me guiou nessa jornada que por muitas vezes pensei em desistir.

Aos meus pais, João e Eni, que sempre me apoiaram nos estudos e fizeram de tudo para que eu pudesse seguir esse caminho.

Agradeço aos meus colegas de mestrado e companheiros de pesquisa Douglas Lieira, Joahannes Bruno e Thiago Gomides, que me auxiliaram diretamente no desenvolvimento do meu projeto e, por muitas vezes, se reuniram comigo afim de me ajudar a solucionar problemas. Agradeço também aos outros colegas e amigos do grupo de pesquisa, que de alguma forma tiveram participação durante o período de mestrado (Eyclides, Leal e Rickson).

À minha namorada Marcela, que me apoia e conforta nos momentos de desespero e ao amigo que o Canadá me trouxe, Matheus Frölich.

Ao Prof. Robson E. De Grande, que aceitou me orientar no Canadá e me permitiu vivenciar uma oportunidade que antes era apenas um sonho.

Finalmente, ao meu orientador, o Prof. Rodolfo Ipólito Meneguette, por ter aceitado me orientar desde a graduação até aqui no mestrado, pelas oportunidades fornecidas e pela amizade criada.

"*You step into the Road, and if you don't keep your feet, there is no knowing where you might be swept off to*"

(TOLKIEN, 1991, p. 97)

RESUMO

O crescimento no número de veículos circulando e o avanço da tecnologia empregada nesses veículos trazem uma grande quantidade de recursos computacionais que podem ser utilizados de diversas formas. Nas VANETs, os veículos realizam entre eles (V2V) e com estruturas físicas (V2I) a comunicação e troca de informações. Com isso, o uso de Nuvens Veiculares vem crescendo como alternativa nos cenários de Sistemas de Transporte Inteligentes. Com o auxílio de Nuvens Veiculares, os veículos dentro da rede compartilham seus recursos individuais formando um *pool* de recursos partilhados. O uso de um novo paradigma alinhado com as Nuvens Veiculares, chamado de *fog computing*, estende os recursos da nuvem até a borda da rede como forma de garantir baixa latência, escalabilidade e mais segurança. Dessa forma, algoritmos para o processo de tomada de decisão na questão de qual recurso alocar a qual serviço dentro da rede se fazem necessários para otimizar esse processo, maximizando o número de recursos alocados e minimizando os recursos subutilizados. Diversas publicações investigaram a aplicação de algoritmos meta-heurísticos para resolver vários problemas de otimização, incluindo alocação eficiente de recursos, agrupamento, alocação de tarefas e comunicações de rede. Nesse trabalho são apresentados alguns dos tipos de *Cloud* e métodos para decisão no processo de alocação de recursos, sendo eles divididos em métodos tradicionais e métodos meta-heurísticos. Foi realizado um levantamento trazendo os trabalhos do atual estado da arte acerca do tema de técnicas utilizadas para alocação de recursos. Sendo assim, como forma de melhorar o gerenciamento da alocação de recursos e tarefas, implementou-se um mecanismo meta-heurístico baseado no comportamento de caça dos morcegos para realizar o processo de tomada de decisão na alocação de recursos em *Fogs* Veiculares para aproveitar melhor os recursos disponíveis dos veículos, alocar mais tarefas e maximizar a utilização dos recursos. O algoritmo do morcego foi selecionado como base para o mecanismo desenvolvido, dado ser considerado uma evolução de outros algoritmos meta-heurísticos, ser baseado em população e gerar múltiplas soluções e pelo seu processo de busca. Como forma de validar o mecanismo proposto, a avaliação foi conduzida através do uso do framework veicular e simulador de rede VEINS e OMNet++, juntamente com diversas mobilidades veiculares aleatórias geradas pelo Simulador de Mobilidade Urbana - SUMO. Foram avaliados as métricas de tarefas alocadas, utilização de recursos e tarefas não alocadas. O mecanismo foi comparado com técnicas tradicionais e meta-heurísticas encontradas na literatura, demonstrando-se mais eficiente que as mesmas.

Palavras-chave: Alocação de recursos. Nuvens veiculares. Fog Computing. Algoritmos. Meta-heurísticos.

ABSTRACT

The growth in the number of vehicles circulating and the advancement of technology used in these vehicles bring a large number of computational resources that can be used in different ways. In VANETs, vehicles communicate and exchange information with each other (V2V) and with physical structures (V2I). As a result, the use of Vehicular Clouds has been growing as an alternative in Intelligent Transport Systems scenarios. With the help of Vehicular Clouds, vehicles within the network share their individual resources forming a pool of shared resources. The use of a new paradigm aligned with Vehicular Clouds, called fog computing, extends cloud resources to the edge of the network as a way to ensure low latency, scalability, and more security. Thus, algorithms for the decision-making process regarding which resource to allocate to which service within the network is necessary to optimize this process, maximizing the number of allocated resources and minimizing underutilized resources. Several publications have investigated the application of metaheuristic algorithms to solve various optimization problems, including efficient resource allocation, clustering, task allocation, and network communications. In this work, some of the types of Cloud and methods for decision-making in the resource allocation process are presented, and divided into traditional methods and meta-heuristic methods. A survey was carried out bringing the works of the current state of the art on the subject of techniques used for resource allocation. Therefore, as a way to improve the management of resource and task allocation, a meta-heuristic mechanism based on the hunting behavior of bats was implemented to carry out the decision-making process in the allocation of resources in vehicular fogs to make better use of available vehicle resources, allocate more tasks and maximize resource utilization. The bat algorithm was selected as the basis for the developed mechanism, as it is considered an evolution of other meta-heuristic algorithms, is based on population, and generates multiple solutions and its search process. As a way of validating the proposed mechanism, the evaluation was conducted using the VEINS and OMNet++ vehicular framework and network simulator, together with several random vehicular mobilities generated by the Simulation of Urban Mobility - SUMO. Metrics of allocated tasks, resource utilization, and unallocated tasks were evaluated. The mechanism was compared with traditional and meta-heuristic techniques found in the literature, proving to be more efficient than them.

Keywords: Resource allocation. Vehicular clouds. Vehicular fogs. Algorithms. Meta-heuristics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AHP	*Analytic Hierarchy Process*
BAT	*Bat Algorithm*
CC	*Cloud Computing*
CH	*Cluster Head*
GA	*Genetic Algorithm*
GLS	*Guided Local Search*
HS	*Harmony Search*
IaaS	*Infrastructure as a Service*
IBGE	Instituto Brasileiro de Geografia e Estatística
IoT	*Internet of Things*
ITS	*Intelligent Transportation System*
MA	*Microcanonical Annealing*
MCC	*Mobile Cloud Computing*
MEC	*Mobile Edge Computing*
NIA	*Nature Inspired Algorithms*
OBU	*On-Board unit*
OMNet++	*Objective Modular Network Testbed in C++*
PaaS	*Plataform as a Service*
PSO	*Particle Swarm Optimization*
QoS	*Quality of Service*
RSU	*Roadside Unit*
SaaS	*Software as a Service*
SUMO	*Simulation of Urban Mobility*
TS	*Tabu Search*

V2I	*Vehicle-to-Infrastructure*
V2V	*Vehicle-to-Vehicle*
VANET	*Vehicular Ad Hoc Network*
VC	*Vehicular Cloud*
VCC	*Vehicular Cloud Computing*
VEC	*Vehicular Edge Computing*
VEINS	*Vehicular Network Simulation Framework*
WOA	*Whale Optimization Algorithm*

SUMÁRIO

1 INTRODUÇÃO

No decorrer dos últimos anos o número de veículos vendidos aumentou de forma significativa, cerca de 1.949.892 automóveis foram emplacados no Brasil no ano de 2020 (AUTOO, 2021). Dados do Instituto Brasileiro de Geografia e Estatística (IBGE) indicam que, no ano de 2020, a frota brasileira era de 107.948.371 veículos, passando para aproximadamente 111.446.870 em 2021 (IBGE, 2021). A alta no número de veículos que circula nas cidades tem consequências socioeconômicas como o aumento da emissão de CO_2 pela queima do combustível e congestionamento e engarrafamentos nas vias urbanas (MENEGUETTE; DE GRANDE; LOUREIRO, 2018).

Algumas alternativas têm surgido como forma de tentar sanar esses problemas ou até mesmo diminuir seus impactos, sendo uma delas o uso dos Sistemas de Transporte Inteligentes (*Intelligent Transportation Systems* - ITS). Nos ITSs, os veículos utilizam sensores e dispositivos de comunicação para realizar a troca de informações e recursos entre os membros da rede, a fim de solucionar os tipos de problemas abordados (MENEGUETTE; DE GRANDE; LOUREIRO, 2018). Essa comunicação é feita através de uma Rede Veicular (*Vehicular Ad Hoc Network* - VANET) pela qual os veículos podem se comunicar de duas formas diferentes: Veículo-para-Veículo (*Vehicle-to-Vehicle* - V2V) ou Veículo-para-Infraestrutura (*Vehicle-to-Infrastructure* - V2I). Contudo, algumas particularidades são encontradas nessas redes, como alta mobilidade dos nós e topologia dinâmica (alta entrada/saída). Dessa forma, faz-se necessária a ampliação da infraestrutura da rede para um melhor atendimento e comunicação entre os nós, através o uso das Nuvens Veiculares (Vehicular Clouds - VCs) que trazem flexibilidade por meio de serviços sob demanda de baixo custo (ZHANG; GRANDE; BOUKERCHE, 2015; BOUKERCHE; DE GRANDE, 2018).

As Nuvens Veiculares utilizam dos paradigmas de *Cloud* como qualidade de serviço (*Quality of Service* - QoS), recursos sob demanda e escalabilidade para compartilhar recursos computacionais de forma dinâmica em uma rede formada por um conjunto de veículos e dispositivos (MENEGUETTE; DE GRANDE; LOUREIRO, 2018). Esses recursos são compartilhados entre os membros da rede e devem ser alocados de forma efetiva para que o desempenho da VC seja bem sucedido e os mesmos aproveitados da melhor forma possível. Para isso, é fundamental uma boa tomada de decisão na hora de selecionar, alocar e direcionar esses recursos entre os membros, do contrário eles podem ser subutilizados e não atingirem seu potencial (ZHANG; GRANDE; BOUKERCHE, 2015; MENEGUETTE; BOUKERCHE; DE GRANDE, 2016).

As soluções fornecidas pela computação em nuvem podem prover uma variedade

de serviços, mas as demandas impostas a elas criam novos problemas (MENEGUETTE; BOUKERCHE, 2017; ZHANG et al., 2022). No entanto, um novo paradigma conhecido como *Fog Computing* estende os recursos de computação distribuídos acessíveis na nuvem até a borda da rede para garantir critérios como baixa latência, escalabilidade, estabilidade e segurança (PEREIRA et al., 2019; HAMDI; HUSSAIN; HUSSAIN, 2022). *Fog computing* é uma extensão da ideia de *edge computing*, permitindo o uso e execução de serviços em nuvem. Com isso, é possível combinar os recursos tanto dos equipamentos quanto dos veículos que compõem uma *fog* (SON et al., 2021; KUMAR et al., 2021).

Os serviços têm baixa latência ou mesmo baixa flutuação na latência, porque as *Fogs* operam na borda. Dessa forma, ganham um componente posicional e podem ser utilizados/executados mais próximos do usuário (CHEN et al., 2021; LAKHAN et al., 2022). Esses serviços oferecidos pela rede vão desde uma simples troca de mensagens e um aviso sobre um acidente, até um serviço de streaming e sugestões de rotas alternativas (SHRESTHA; BAJRACHARYA; NAM, 2018; PAUL et al., 2017). Para executar essas tarefas são necessários recursos computacionais como processamento, memória RAM e armazenamento. Um mecanismo eficaz de alocação e gerenciamento de recursos computacionais é crucial para realizar esse balanceamento de carga e atender melhor às necessidades desses serviços e aplicativos (MAO et al., 2022; LI et al., 2022).

O uso de algoritmos no processo de alocação de recursos em VCs já é algo conhecido e estudado. Esses algoritmos têm como função a decisão no processo de alocação de qual recurso é melhor ser alocado em determinada estrutura e quando. Os algoritmos meta-heurísticos se enquadram na categoria de otimização com base estocástica, sendo que uma grande porção deles segue conceitos da natureza e esses podem ser chamados de Algoritmos Inspirados na Natureza (*Nature Inspired Algorithms* - NIA) ou bio-inspirados (WONG; MING, 2019; JHA; JAIN; THENMALAR, 2018).

Os algoritmos meta-heurísticos podem ser divididos em três principais categorias: algoritmos evolutivos, de inteligência de enxame e inspirados na natureza (FARIS et al., 2016; WONG; MING, 2019). As principais inspirações são o comportamentos de animais, como por exemplo o de caça de determinada espécie ou em grupo, fenômenos físicos e princípios evolutivos. A utilização desses algoritmos e o interesse neles se dão devido a sua natureza estocástica e não determinística para resolução de problemas complexos (VIKHAR, 2016).

1.1 Motivação

O aumento dos veículos, investimentos e novas pesquisas nas áreas descritas, movimentam uma série de dúvidas acerca das melhores abordagens para resolução desses problemas. A escolha apropriada de determinada abordagem é necessária para que

recursos sejam poupados e melhor utilizados na rede.

Diversos trabalhos da literatura buscam resolver os problemas relacionados a alocação de recursos e tarefas, como Mohanty et al. (2018), Klaimi, Senouci e Messous (2018), Pereira et al. (2019), Pereira et al. (2020), Sarubbi et al. (2017), Li et al. (2019), Kong et al. (2019), Sun et al. (2019), Jayswal (2020), Midya et al. (2018), Lieira et al. (2021). Alguns deles utilizam métodos tradicionais, outros meta-heurísticos e alguns até mesmo a combinação de meta-heurísticos com tradicionais, aplicados em diferentes tipos de estruturas de alocação com as mais variadas características.

Neste trabalho, busca-se lidar com o problema de alocação de recursos e da otimização da seleção correta dos veículos através do desenvolvimento do ICARUS, baseado no algoritmo meta-heurístico bio-inspirado do morcego, para realização da tomada de decisão no processo de alocação de tarefas em *Fogs* veiculares, a fim de maximizar os recursos alocados e alocar uma maior quantidade de tarefas.

1.2 Objetivos

Nesta monografia tem-se como objetivo geral implementar e avaliar uma técnica meta-heurística para realização do gerenciamento no processo de alocação de recursos em Fogs Veiculares, bem como compará-la com outras técnicas conhecidas. O objetivo, então, é otimizar esse processo ao visar maximizar o número de recursos alocados, diminuir o número de recursos subutilizados e aumentar o número de veículos atendidos.

Desta forma, tem-se os seguintes objetivos específicos:

- Desenvolver o ICARUS para tomada de decisão no processo de alocação de recursos em um cenário com *Fogs* veiculares, baseado no algoritmo meta-heurístico bio-inspirado do morcego;
- Implementar diferentes algoritmos para comparação em termos de desempenho no processo de alocação de recursos;
- Realizar simulações em diferentes cenários utilizando simuladores de rede como forma de validar a técnica proposta e avaliar o desempenho dos algoritmos, considerando métricas relevantes a rede;
- Maximizar o número de recursos alocados através do mecanismo proposto; e
- Aumentar o número de tarefas alocadas e diminuir a quantidade de tarefas não alocadas.

1.3 Organização do trabalho

A dissertação está organizada da seguinte forma:

No **Capítulo 2** é apresentado a fundamentação teórica do trabalho. São apresentados os tipos de *Cloud*, abordando a *Cloud Computing*, *Mobile Cloud Computing* e *Vehicular Cloud Computing*. Também apresenta diferentes métodos que são utilizados na alocação de recursos, sendo classificados em tradicionais e meta-heurísticos.

No **Capítulo 3** são apresentados alguns trabalhos do atual estado da arte relacionados com o tema de alocação de recursos utilizando métodos tradicionais e meta-heurísticos.

No **Capítulo 4** é apresentado o desenvolvimento do ICARUS, com a definição do problema, modelagem do cenário, técnicas de comparação e métricas utilizadas, bem como o resultado obtido através das avaliações realizadas por simulação.

Por fim, no **Capítulo 5** são apresentados as conclusões finais deste trabalho, as produções e contribuições técnicas científicas e diretrizes para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentado a fundamentação teórica do trabalho, onde são abordados primeiramente os tipos de computação em nuvem, passando pela computação em nuvem tradicional, computação em nuvem móvel e computação em nuvem veicular. Após isso, são abordados diferentes técnicas para tomada de decisão e gerenciamento de alocação de recursos, divididas em métodos tradicionais e meta-heurísticos.

O capítulo está organizado da seguinte forma: na Seção 2.1 são abordados os tipos de computação em nuvem e na Seção 2.2 são abordados os tipos de métodos para alocação de recursos.

2.1 Tipos de Computação em nuvem

Com o avanço do movimento de Internet das Coisas (*Internet of Things* - IoT), o número de dispositivos conectados a internet aumentou significantemente. Esse aumento gerou uma demanda muito alta em termos de processamento de dados, armazenamento, gerenciamento e alocação desses recursos. A procura por formas alternativas de lidar com essa grande quantidade de dados e o poder computacional necessário para isso, abriram caminho para a Computação em Nuvem (*Cloud Computing* - CC) (BELLO et al., 2021; SHIJU GEORGE; SUJI PRAMILA, 2021).

O avanço da Computação em Nuvem trouxe outros avanços consigo, sendo um deles o conceito de Computação em Nuvem Móvel (*Mobile Cloud Computing* - MCC). O aumento no número de *smartphones* e a evolução desses dispositivos em termos de capacidade de armazenamento, processamento, acessibilidade, e outros, cresceu nos últimos tempos (FELLAH; MEZIOUD; BATOUCHE, 2020). Esse modelo considera os dispositivos móveis pontos de acesso a nuvem e tem como objetivo superar os problemas de recursos limitados que os dispositivos móveis possuem em relação a vida da bateria, largura de banda baixa, armazenamento e outros, utilizando a nuvem para realizar o processamento de funções que utilizam uma carga maior de recursos computacionais (FELLAH; MEZIOUD; BATOUCHE, 2020; KHALEDI; KHALEDI; KASERA, 2016).

Junto com o desenvolvimento das tecnologias de Computação em Nuvem e da Computação em Nuvem Móvel, um novo paradigma em junção às VANETs, surge como a Computação em Nuvens Veiculares (*Vehicular Cloud Computing* - VCC). Na VCC, os veículos pertencentes a rede compartilham seus recursos computacionais entre os membros, podendo esses recursos serem utilizados para execução de serviços e tarefas custosas em termos computacionais (BOUKERCHE; DE GRANDE, 2018).

2.1.1 Computação em Nuvem (*Cloud Computing* - CC)

A Computação em Nuvem, ou *Cloud Computing* CC, é um paradigma da computação que busca fornecer recursos computacionais, sejam eles hardware ou software, em qualquer lugar e a qualquer momento. Esse paradigma leva em consideração alguns pontos, sendo eles: entrega de recursos computacionais sobre demanda, Qualidade do Serviço (*Quality of Service* - QoS), disponibilidade a todo momento (24/7/365), entre outros.

Na Computação em Nuvem, os recursos computacionais dos membros existentes na rede são considerados uma *pool* compartilhada que pode ser configurada para ser acessada através da rede. Dessa forma, qualquer um com acesso a essa rede pode partilhar seus recursos e utilizar recursos de outros membros (BELLO et al., 2021; SHIJU GEORGE; SUJI PRAMILA, 2021; VASANTHAAZHAGU; GNANASEKAR, 2016).

A Computação em Nuvem fornece três principais tipos de serviços, sendo eles (VASANTHAAZHAGU; GNANASEKAR, 2016):

1. Infraestrutura como Serviço (*Infrastructure as a Service* - IaaS): toda infraestrutura necessária para computação, podendo ser desde um espaço no *datacenter*, componentes de rede, até servidores;

2. Plataforma como Serviço (*Plataform as a Service* - PaaS): fornece um ambiente de serviço para seus usuários. Esse ambiente pode ser tanto um serviço de hardware ou software, que pode ser usado para armazenamento, sistemas gerenciadores de banco de dados, sistemas operacionais, etc;

3. Software como Serviço (*Software as a Service* - SaaS): empréstimo e gerenciamento de softwares para clientes ou provedores de forma remota.

2.1.2 Computação em Nuvem Móvel (*Mobile Cloud Computing* - MCC)

Nos últimos anos a Computação em Nuvem Móvel, ou *Mobile Cloud Computing* MCC, vem sendo um assunto crescente de interesses dos pesquisadores devido ao desenvolvimento das redes móveis e da Computação em Nuvem (FELLAH; MEZIOUD; BATOUCHE, 2020). A Computação em Nuvem Móvel considera o uso de dispositivos móveis, sendo eles *smartphones*, *tablets*, notebooks, e sua conexão com a internet através de redes móveis 4G, 5G, e até WiFi para realizar a conexão com a *Cloud* (ALAKBAROV; ALAKBAROV, 2017).

Na MCC, o armazenamento e o processamento dos dados acontecem fora do dispositivo móvel, sendo executados em uma *Cloud*. Dessa forma, a aplicação é rodada em um computador com capacidade computacional suficiente para executar determinado serviço, sendo os dispositivos móveis vistos como clientes nesse cenário. Com isso, a MCC supera as limitações dos dispositivos móveis como poder computacional, armazenamento,

memória e consumo de energia (FELLAH; MEZIOUD; BATOUCHE, 2020; ALAKBAROV; ALAKBAROV, 2017).

2.1.3 Computação em Nuvem Veicular (*Vehicular Cloud Computing* - VCC)

O avanço da indústria automotiva e as tecnologias hoje empregadas nos veículos pelas montadoras, trazem aos motoristas um novo leque de opções relacionadas aos Veículos Inteligentes (*Smart Vehicles*) (BOUKERCHE; DE GRANDE, 2018). Um *framework* de Vehicular Clouds foi proposto pela primeira vez em Olariu, Khalil e Abuelela (2011), quando foram fornecidas soluções eficientes para problemas de segurança, disponibilidade, confiabilidade, bem como os principais problemas.

As Nuvens Veiculares (*Vehicular Clouds* - VCs), consistem em diversos veículos e dispositivos que compartilham seus recursos através do paradigma da Computação em Nuvem (MENEGUETTE; DE GRANDE; LOUREIRO, 2018). A VC, busca maximizar a utilização dos recursos ociosos dos veículos e segue conceitos como QoS, disponibilidades e escalabilidade, o que o torna um processo árduo devido a volatilidade dos recursos. Como a rede veicular (VANET) na qual a VC é implantada é uma rede de topologia dinâmica (entrada e saída de veículos rápida) e alta mobilidade dos nós, diversos problemas frente a alocação de recursos são encontrados (BOUKERCHE; DE GRANDE, 2018; MENEGUETTE; BOUKERCHE; DE GRANDE, 2016).

Veículos perto uns dos outros podem ser agrupados e considerados membros de uma mesma *Cloud* e compartilham um *pool* de recursos entre si e entre a rede. Esses recursos são utilizados em serviços, como troca de mensagens, informações de estacionamento, *streaming* de dados, etc. Dessa forma, a rede consegue fornecer recursos de forma dinâmica aos membros, suprindo as necessidades de recursos computacionais dos serviços e utilizando os recursos dos veículos que antes seriam subutilizados (BOUKERCHE; DE GRANDE, 2018).

2.2 Métodos para Decisão de Alocação de Recursos

Nesta seção são abordados diversos algoritmos utilizados para decisão no processo de alocação de recursos. O processo de alocação de recursos pode ser considerado a ação de encontrar uma solução ótima ou o processo de otimização da alocação de um recurso, que nada mais é do que escolher a melhor opção para alocar o recurso (DIVYA; JAYANTHI, 2020). Nas seções seguintes estão apresentados algoritmos que podem ser utilizados para solucionar o problema de otimização na decisão de qual a melhor *Fogs* ou o melhor membro do cenário para alocar determinado recurso.

2.2.1 Métodos tradicionais

Nesta seção são descritos alguns métodos, aqui chamados de tradicionais, encontrados na literatura para solução de problemas de otimização/alocação de recursos. Neste trabalho, são considerados como métodos tradicionais os algoritmos que não se encaixam no conceito de métodos meta-heurísticos. Esses métodos tradicionais são métodos baseados em modelos matemáticos e de pesquisa encontrados na literatura. Consideram normalmente uma formulação matemática teórica e prática para realizar a decisão da alocação e gerenciamento de recursos.

Sendo assim, são abordados dois algoritmos tradicionais da literatura, o Processo Hierárquico Analítico e a Teoria dos Jogos.

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

O Processo Hierárquico Analítico (AHP) é um método tradicional que pode ser utilizado no processo de tomada de decisão em diversos cenários, sejam eles na CC (MAKWE; KANUNGO, 2015), Computação em Nuvem Mobile (*Mobile Edge Computing* - MEC) (PEREIRA et al., 2020), VCs (PEREIRA et al., 2019), entre outros. Proposto por Thomas L. Saaty, o AHP tem como premissa ser uma técnica estruturada para organizar e analisar decisões complexas. O AHP é um método que se baseia em critérios múltiplos para tomada de decisão. Saaty recomenda a utilização de cerca de 9 níveis de critérios para a tomada de decisão. Outro ponto considerado pelo método para a decisão são os pesos dos critérios utilizados, sendo que de acordo com o peso estipulado, determinado critério tem uma maior relevância (GOLDEN; WASIL; HARKER, 1989; SAATY, 1987; SAATY, 1991). Na Tabela 1 é possível observar os estágios de importância definidor por Saaty com seus valores de importância e descrição.

De acordo com Jaiswal (1997), a metodologia do AHP pode ser descrita em 7 passos:

- Passo 1: o sistema é estudado em detalhes com uma visão de identificar o objetivo, as alternativas e critérios/subcritérios para comparação das alternativas. As informações são organizadas em uma estrutura hierárquica. Nessa hierarquia, o objetivo final do exercício é definido e os critérios e comparações são especificados;
- Passo 2: é realizado a coleta de dados de especialistas no assunto de acordo com a estrutura hierárquica. As comparações aos pares são feitas em relação aos critérios do nível anterior, ou seja, os critérios são comparados em relação à sua importância para o objetivo final, utilizando a os critérios da Tabela 1.
- Passo 3: As comparações dos pares de vários critérios gerados na Etapa 2 são organizadas em uma matriz quadrada. O tamanho da matriz corresponde ao número

Tabela 1 – Critérios de importância definidos por Saaty.

Classificação	**Valor de Importância**	**Descrição**
Mesma importância	1	Ambas premissas contribuem para o objetivo da mesma forma
Importância pequena	3	Uma premissa possuí levemente uma vantagem em relação a outra.
Importância grande	5	Uma premissa possuí fortemente uma vantagem em relação a outra.
Importância muito grande	7	Uma premissa possuí muito fortemente uma vantagem em relação a outra.
Importância extremamente forte	9	É absoluta a evidência de uma premissa em relação a outra.
Valores intermediários	2, 4, 6, 8	Indicam compromisso entre as premissas.
Uma designação razoável	Recíprocos	Usados para demonstrar dominância entre a segunda premissa quando comparada com a primeira.

Fonte: (JAISWAL, 1997; GOLDEN; WASIL; HARKER, 1989), adaptado pelo autor.

de critérios. É realizado o mesmo processo em outros níveis da hierarquia, gerando matrizes com relação a cada critério ou subcritério.

- Passo 4: O autovalor principal é avaliado e o autovetor de cada matriz é avaliado. Os elementos dos vetores são normalizados e a estrutura das matrizes é organizada de acordo com a avaliação realizada em cima dos critérios/subcritérios comparados na matriz.

- Passo 5: A consistência das matrizes CI são avaliadas através da fórmula $CI = (\lambda_{max} - N)/(N - 1)$ sendo λ_{max} o valor máximo de autovalor e N a ordem da matriz. Um valor de Índice de Inconsistência Aleatório RI é dado por Saaty. O CI é comparado com RI. Se $CI < 0,1$ a matriz é considerada consistente, caso contrário há um tipo de inconsistência e é pedido ao especialista que reveja sua opinião.

- Passo 6: Devido a inconsistências que as opiniões de cada especialista pode gerar no sistema, a mesma matriz é refeita por outro especialista e a média geométrica das duas é tirada como forma de balancear esse problema.

- Passo 7: As classificações obtidas de cada alternativa são multiplicadas pelos pesos de cada critério e ao final são geradas as classificações globais de cada alternativa.

2.2.1.2 Teoria dos Jogos

Outra abordagem tradicional é a teoria dos jogos. A teoria dos jogos foi proposta em 1944 por John Von Neumann em um trabalho intitulado *Theory of Games and Eco-*

nomic Behavior (Teoria dos Jogos e Comportamento Econômico) (NEUMANN, 1944). A teoria dos jogos é comumente utilizada quando há a necessidade de um processo de tomada de decisão na qual existem entidades "jogadores" no cenário que interagem entre si e a decisão desses jogadores pode influenciar nos outros membros (SAJJAD; SHARMA; GERDES, 2017). Na maior parte dos casos quando há a utilização de teoria dos jogos, as estratégias buscam o que pode ser chamado de Equilíbrio de Nash. Esse equilíbrio consiste em um cenário onde os jogadores estabilizam suas estratégias não havendo a necessidade de mudá-las para tentar ganhar algo a mais (NASH, 1951; SAJJAD; SHARMA; GERDES, 2017).

Um jogo pode ser classificado de acordo com o número de membros que participa do mesmo, podendo ser um jogo com um jogador, dois jogadores até n jogadores, considerando o grupo de membros como $J = [j_1, j_2, ..., j_n]$. Em um jogo completo há alguns componentes principais, os agentes (jogadores), a função útil (função de recompensa, *payoff*) e o equilíbrio. O nível de satisfação que deriva das ações dos jogadores é dado de acordo com a função útil, que também pode ser chamada de "pagamento" ou recompensa ao jogador (BAHAMOU; OUADGHIRI; BONNIN, 2016).

De forma geral, o equilíbrio pode ser encontrado através de dois principais elementos, o Equilíbrio de Nash e o Ótimo de Pareto (DAS; GOSWAMI; KONAR, 2019):

- Equilíbrio de Nash: é um resultado em que cada jogador está fazendo o melhor que pode, dadas as escolhas dos outros jogadores. Portanto, nenhum jogador pode se beneficiar com a mudança unilateral de sua escolha.
- Ótimo de Pareto: é um resultado a partir do qual qualquer tentativa de beneficiar alguém, desviando-se para algum outro resultado, necessariamente resultará em uma perda de satisfação para outra pessoa.

Sendo assim, o Ótimo de Pareto significa que não há maneira possível de deixar qualquer jogador melhor sem prejudicar outro jogador (o que é eficiente), enquanto o equilíbrio de Nash é um conceito de equilíbrio (estrategicamente viável).

Uma outra abordagem que ajuda a explicar de forma mais visual e conceitual a teoria dos jogos é o dilema do prisioneiro (KUHN, 2019). O dilema do prisioneiro consiste em 2 prisioneiros sendo acusados de um crime. O promotor diz a cada um deles, que estão separados em celas isoladas uns dos outros, que eles tem duas opções, ou eles confessam ou permanecem em silêncio. Dessa forma as seguintes premissas são estabelecidas:

1. Se você confessar e seu cúmplice permanecer em silêncio, todas as acusações contra você serão retiradas e você irá testemunhar contra o seu cúmplice que irá pegar um longo tempo de cadeia;

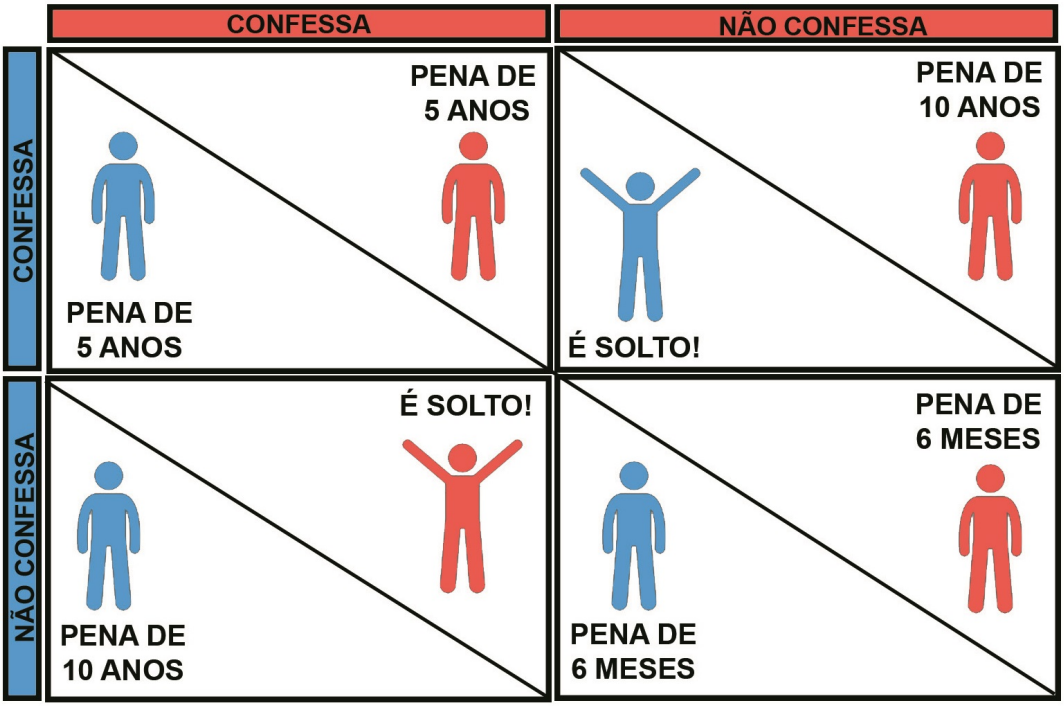
2. Caso você permaneça em silêncio e o seu cúmplice confessar, ele sairá livre e você irá para a cadeia;

3. Se ambos confessarem, os dois vão presos, porém cumprem uma pena reduzida;

4. Se ambos permanecerem em silêncio, os dois vão presos por uma acusação mais leve e pegarão pouco tempo de pena.

Sendo assim, o dilema que os prisioneiros tem que refletir sobre é: é melhor confessar do que permanecer em silêncio. Contudo, se ambos confessarem, o resultado é pior para cada um do que se os dois permanecessem em silêncio. Dessa forma, o dilema ilustra um conflito de interesses individuais e em grupo, sendo que caso os membros do grupo considerem apenas o interesse próprio, o resultado pode ser pior do que se eles considerarem o interesse do grupo. Uma representação visual do dilema do prisioneiro pode ser vista na Figura 1. Foi considerado que para a premissa um as recompensas (penas) seriam de 10 anos para quem tivesse confessado e quem não confessou sairia livre. Para a premissa 2, o inverso da premissa 1. Para a premissa 3, ambos os prisioneiros recebem uma pena reduzida de 5 anos. Já na última premissa, a premissa 4, nenhum dos dois confessam e recebem uma pena branda de 6 meses.

Figura 1 – Representação do dilema do prisioneiro



Fonte: (SOUSA, 2016), adaptado pelo autor.

Trazendo esse método para o cenário de alocação de recursos em Nuvens Veiculares, é possível abstrair esse conceito sendo considerado um conjunto de recursos

$R = [arm_i, process_i, memor_i]$ podendo ser armazenamento, processamento e memória. Esse recursos são utilizados em um serviço S na VC, podendo ser esse serviço pode ser monitoramento de tráfego, troca de mensagens, sugestão de rota, entre outros. Cada serviço é um jogador que foi solicitado pelo usuário. Esses serviços possuem dois *status*, atendidos e negados. Considerando dois serviços A e B, se ambos forem atendidos, ambos terão um custo de valor 4. Se o serviço A for atendido e o serviço B for negado, A terá custo de 1 e B terá custo de 5. Da mesma forma caso o contrário seja aplicado. Já a outra opção é a qual ambos os serviços serem negados e dessa forma A e B terão custo de 2.

Considerando um conjunto J composto por A e B, sendo que cada jogador possuí um conjunto de estratégias u_i, tendo:

$$S_A = (Atendido, Bloqueado)$$
$$S_B = (Atendido, Bloqueado)$$

é possível observar na Tabela 2 as possíveis combinações.

Tabela 2 – Representação das estratégias dos serviços A e B

		B	
	u_i	Atendido	Negado
A	Atendido	4, 4	1, 5
	Negado	5, 1	2, 2

Fonte: Produzido pelo autor.

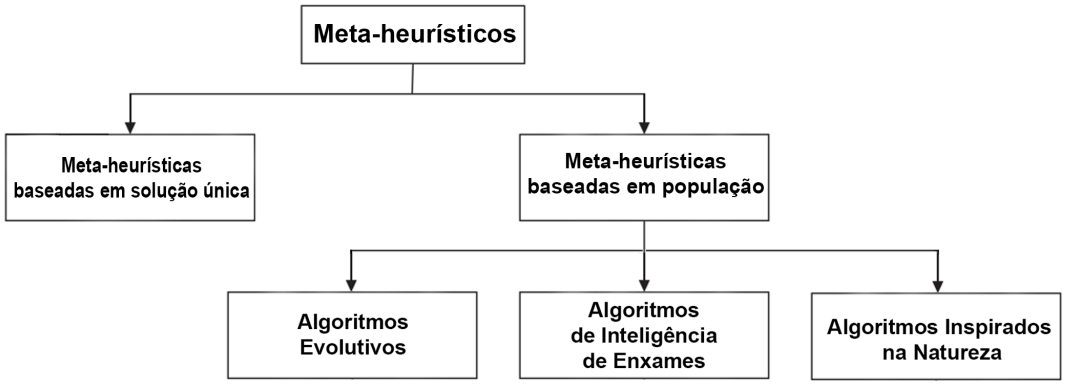
Dessa forma, o equilíbrio de Nash é encontrado onde ambos os serviços são negados e recebem o pagamento de valor 2. No caso em que ambos são atendidos e recebem o pagamento no valor de 4, é encontrado o Ótimo de Pareto.

2.2.2 Métodos meta-heurísticos

Nesta seção são descritos e abordados alguns métodos/algoritmos meta-heurísticos. Os algoritmos meta-heurísticos são utilizados para solução de problemas complexos da vida real em diferentes campos de pesquisa, como economia, engenharia, política e gerenciamento, tendo como elemento chave a intensificação e diversificação (KATOCH; CHAUHAN; KUMAR, 2020). A maior parte dos algoritmos meta-heurísticos são de inspiração na natureza (bio-inspirados) e no comportamento de enxames, podendo ser chamados de Algoritmos Inspirados na Natureza (*Nature Inspired Algorithms* - NIA), devido ao fato de levarem em consideração o comportamento de animais e de outros processos naturais (JHA; JAIN; THENMALAR, 2018; KATOCH; CHAUHAN; KUMAR, 2020).

Esses algoritmos podem ser divididos em duas principais categorias: solução única e algoritmos baseados em população. É possível observar essas categorias na Figura 2. Os de solução única, utilizam uma única solução e a melhoram utilizando busca local, contudo, a solução encontrada pode ficar presa no ótimo local. Alguns exemplos de algoritmos de solução única são a Pesquisa Tabu (*Tabu Search* - TS), Recozimento Microcanônico (*Microcanonical Annealing* - MA) e Busca Local Guiada (*Guided Local Search* - GLS). Os algoritmos baseados em população utilizam múltiplos candidatos a solução durante o processo de busca, buscando evitar ficarem estagnados no ótimo local. Alguns exemplos são o *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO) e *Bat Algorithm* (BAT) (KATOCH; CHAUHAN; KUMAR, 2020).

Figura 2 – Classificação dos algoritmos meta-heurísticos



Fonte: (KATOCH; CHAUHAN; KUMAR, 2020; WONG; MING, 2019), adaptado pelo autor.

Buscando superar a deficiência das técnicas e algoritmos baseados em métodos matemáticos padrões, as técnicas de otimização meta-heurísticas vem para buscar boas soluções (próximas das soluções ótimas) encontradas de forma sustentável do ponto de vista de recursos de computação e processamento e com uma complexidade menor de implementação desses algoritmos (GEEM; KIM; LOGANATHAN, 2001).

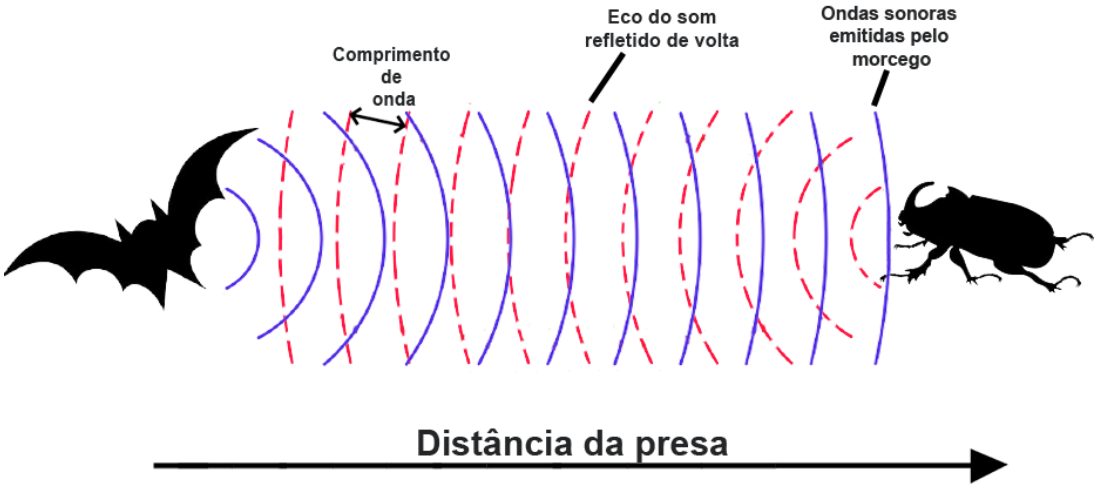
Sendo assim, são trazidos alguns métodos/algoritmos meta-heurísticos utilizados em processos de otimização, sendo eles o algoritmo do morcego (BAT), algoritmo genético (GA), otimização por enxame de partículas (PSO) e busca harmônica (HS).

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

O algoritmo do morcego, é baseado no comportamento de caça dos micro-morcegos. Os micro-morcegos são na sua maioria insetívoros (alimentam-se de insetos). Essa espécie utiliza um método de ecolocalização, muito parecido com um sonar, para detectar suas

presas, localizar fendas e evitar a colisão com obstáculos no escuro. Esse pulso sonoro emitido por eles é muito alto e ecoa até encontrar uma superfície, assim que ele bate na superfície, ele retorna até o morcego e ele consegue se localizar no ambiente. De acordo com a aproximação da presa na caça, o morcego pode emitir mais ou menos pulsos, variando de 10 a 20 pulsos/segundo em uma abordagem considerada normal, podendo chegar a cerca de 200 pulsos/segundo quando estão caçando a presa (YANG, 2010). Uma demonstração de como funciona esse efeito pode ser observado na Figura 3.

Figura 3 – Demonstração do efeito de ecolocalização dos micro-morcegos



Fonte: (SRIVASTAVA; SAHANA, 2019), adaptado pelo autor.

É possível separar a abordagem dos morcegos em duas ações, mesmo eles não seguindo um padrão específico de caça, sendo elas: buscando a presa e avançando em direção a presa. Quando estão buscando a presa, seu comportamento é o de reconhecimento e identificação da presa. Quando estão avançando em direção à presa, seu comportamento é o de ataque (YANG, 2010).

Alguns fatores são levados em consideração para encontrar a melhor solução através desse algoritmo, são eles: o movimento dos morcegos, a velocidade v_i o volume A_i e a emissão de pulso r_i e a frequência f_i. Do contrário de outras espécies que utilizam padrões de caça, os morcegos voam pelo ambiente de forma aleatória, sem um comportamento previamente definido (exceto o descrito acima). A velocidade dos morcegos é atualizada de acordo com as posições/soluções encontradas. O volume do pulso de ecolocalização também varia de acordo com a atividade que o morcego está desempenhando na caça, sendo mais alto quando estão buscando a presa e mais baixo quando avançam na direção dela para apanhá-la. A emissão do pulso aumenta conforme o morcego se aproxima da sua presa. E por fim, a frequência varia na região de 25kHz a 150kHz (YANG, 2010; YANG, 2013).

Da mesma forma que os outros algoritmos deste nicho, o BAT também utiliza

o cálculo do valor do *fitness* como forma de avaliar as soluções geradas até encontrar a melhor solução, ou o individuo mais apto. O *fitness* é um valor calculado através de uma função de otimização, na qual é buscado sempre o menor valor, sendo esse menor valor o valor considerado melhor para a solução.

2.2.2.2 Algoritmo Genético (*Genetic Algorithm* - GA)

O algoritmo genético foi proposto por J. H. Holland e é um algoritmo de solução ótima baseado em pesquisa de população que tem como princípio o processo de evolução biológica, inspirado na teoria de Darwin na qual o mais adaptado na natureza é quem irá sobreviver (Teoria da Evolução das Espécies) (THENGADE; DONDAL, 2012; KATOCH; CHAUHAN; KUMAR, 2020).

O algoritmo GA possuí algumas funções básicas, sendo elas: seleção, cruzamento e mutação (THENGADE; DONDAL, 2012; KATOCH; CHAUHAN; KUMAR, 2020). A seleção é responsável por selecionar os candidatos que irão se reproduzir e passar para uma próxima geração, assim como os que serão excluídos por não apresentarem melhoria ou resultado pior. O cruzamento é um processo que pode ou não ocorrer (60% a 70%), caso não ocorra o indivíduo é copiado diretamente para a próxima geração. Um dos tipos de cruzamento que são mais utilizados é o de ponto único, em que o "filho" pega um "cromossomo" de cada um dos pais para se formar. Após esses dois processos (seleção e cruzamento), é realizado a mutação. A mutação é realizada para garantir que os indivíduos não sejam exatamente os mesmos. Esse processo tem a probabilidade de 1 a 2 décimos de um por cento. No entanto, é considerada vital para garantir a diversidade dos membros da população (THENGADE; DONDAL, 2012).

De forma geral, o GA seleciona os candidatos que podem se reproduzir. Desses candidatos, várias cópias deles são geradas, mas não cópias perfeitas, sendo algumas alterações introduzidas durante o processo de cópia. Essas cópias "filhos" vão para a próxima geração sendo considerados os novos candidatos. Os candidatos considerados piores que os "pais" ou que não apresentaram melhorias são descartados e os que sobraram são sujeitados ao próximo passo e assim por diante. É esperado que a cada iteração, a população de candidatos seja melhorada chegando ao fim com a melhor solução para o problema (THENGADE; DONDAL, 2012).

Uma visão mais específica do GA clássico é descrita a seguir. Uma população Y com x cromossomos é criada de forma aleatória. O valor do *fitness* de cada população em Y é computado. De acordo com o valor do *fitness* dessa população, são selecionados dois cromossomos $C1$ e $C2$. A operação de cruzamento de ponto único é realizada nos cromossomos $C1$ e $C2$ dada uma probabilidade Cp para gerar uma prole O. Após isso o processo de mutação é aplicado a O com uma probabilidade Mp de gerar uma nova prole O'. Dessa forma, esse processo é repetido diversas vezes (seleção, cruzamento e mutação)

até que a nova população seja completa e as melhores soluções (soluções ótimas) sejam geradas (KATOCH; CHAUHAN; KUMAR, 2020).

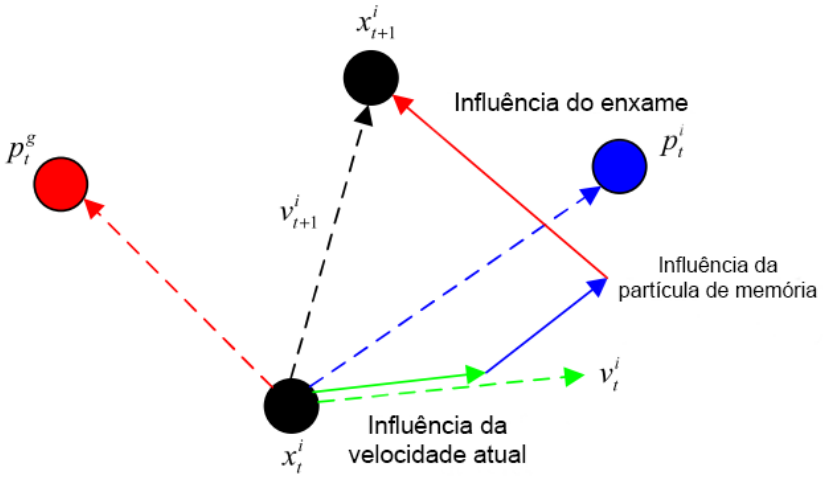
2.2.2.3 Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO)

O algoritmo de otimização por enxame de partículas é um algoritmo estocástico baseado em população, baseado na inteligência coletiva do comportamento de alguns animais, especialmente cardumes de peixes e bandos de pássaros (WANG; TAN; LIU, 2018). O PSO foi apresentado em 1995 (EBERHART; KENNEDY, 1995) e desde então vem sofrendo alterações para que seu desempenho possa ser melhorado.

O algoritmo PSO simula o comportamento social dos animais incluindo insetos, peixes e aves, sendo cada elemento considerado uma partícula (*particle*). O grupo desses animais podem ser considerados "enxames" (*swarms*), em que possuem um comportamento cooperativo para encontrar comida de acordo um padrão que vai sendo alterado conforme as experiências de aprendizado (WANG; TAN; LIU, 2018).

Pode-se considerar que o PSO possuí duas versões, sendo elas a versão global e a local. Na versão global, as partículas rastreiam suas próprias posições ótimas *pbest* e a melhor posição do enxame *gbest*. Na versão local, também é rastreado a própria posição *pbest*, contudo a *gbest* não é rastreada, do contrário, é rastreado a posição ótima de todas as partículas *nbest* vizinhas na topologia (WANG; TAN; LIU, 2018). A cada geração, o processo de iteração das partículas pode ser observado na Figura 4.

Figura 4 – Iteração do esquemas de partículas



Fonte: (WANG; TAN; LIU, 2018), traduzido pelo autor.

Analisando a Figura 4, é possível observar que a primeira parte é a influência da velocidade da partícula anterior. A segunda parte depende da distância entre a partícula "posição atual" e sua própria posição ótima *pbest*, chamada de item cognitivo. A ter-

ceira parte depende da distância entre a partícula "posição atual" e a posição ótima do enxame, chamada de fator social (WANG; TAN; LIU, 2018). Refere-se à colaboração e compartilhamento de informações entre as partículas, especificamente o movimento da partícula com base na experiência de outras partículas no enxame.

2.2.2.4 Busca Harmônica (*Harmony Search* - HS)

O algoritmo de Busca Harmônica foi desenvolvido por Zong Woo Geem et al., desde então é aplicado para resolver diversos problemas de otimização (YANG, 2009; GEEM; KIM; LOGANATHAN, 2001). O HS é um algoritmo de otimização meta-heurístico baseado na música, sendo inspirado no fenômeno da harmonia musical, imitando o efeito de improvisação que os músicos utilizam (LI; WANG, 2018; GEEM; KIM; LOGANATHAN, 2001).

A harmonia musical é a combinação de sons considerados agradáveis do ponto de vista estéticos. O estudo de harmonia na natureza é estudado desde 582 AC pelo matemático e filósofo *Pitágoras*, sendo considerado uma relação especial entre as ondas sonoras que possuem diferentes frequências. Também foi estudado por um compositor e musicista chamado *Jean-Philippe Rameau* que estabeleceu a teoria da harmonia clássica (GEEM; KIM; LOGANATHAN, 2001).

A qualidade de um instrumento musical é determinada pelo seu tom (frequência), timbre (qualidade do som) e amplitude (volume). Cada nota musical possuí uma frequência diferente. O timbre é determinado na sua maioria pelo conteúdo harmônico que é a modulação do sinal ou a formação de ondas do som. Contudo, de acordo com o intervalo de frequência de cada instrumento musical, os harmônicos gerados são diferentes (YANG, 2010).

A processo de improvisação que um músico experiente utiliza, pode ser divido em três possíveis escolhas (YANG, 2010): 1. Tocar um pedaço de uma música famosa (uma série de arranjos em harmonia) de acordo com sua memória; 2. Tocar algo similar a uma música conhecida (ajustando o arranjo levemente); 3. Compor uma nova música ou tocar notas aleatórias. Sendo elas nomeadas como uso de memória harmônica, ajuste do arranjo e randomização.

A memória harmônica é parecida com a escolha dos melhores indivíduos utilizados no Algoritmo Genético GA. Isso assegura que a melhor harmonia vai ser carregada na nova memória harmônica. Para isso, é utilizado um parâmetro para que a memória harmônica possa ser utilizada de forma mais efetiva $r_{accept} \in [0, 1]$, sendo $r_{accept} = 0,7 \sim 0,95$, chamado taxa de consideração (YANG, 2009).

O segundo componente é o ajuste do arranjo determinado pela largura de banda do arranjo b_{range} e a taxa de ajuste r_{pa}. Para o ajuste, é utilizado um ajuste linear

$x_{new} = x_{old} + b_{range} * \varepsilon$, sendo $_{xold}$ a solução existente da memória harmônica, x_{new} o novo arranjo após o ajuste. Um número aleatório no intervalo de [-1,1] é utilizado para ε como forma de variar a solução (YANG, 2009).

O terceiro componente é a randomização, que aumenta a diversidade das soluções. O ajuste de arranjo é similar a randomização, mas limitado ao ajuste local do arranjo, correspondendo a uma busca local. A randomização pode levar o modelo a explorar diversas soluções para encontrar o ótimo global. Sendo a probabilidade de uma randomização $P_{random} = 1 - r_{accept}$ e a probabilidade atual de ajuste do arranjo sendo $P_{pitch} = r_{accept} * r_{pa}$ (YANG, 2009).

2.3 Considerações parciais

Neste capítulo é apresentado a fundamentação teórica do trabalho. São abordados os principais modelos de Computação em Nuvem, desde a própria Computação em Nuvem (CC), até a Computação em Nuvem Móvel (MCC) e Computação em Nuvem Veicular (VCC). Também é discutido sobre os métodos para decisão de alocação de recursos, divididos entre tradicionais e meta-heurísticos. Na Tabela 3 é possível observar as características principais dos três tipos de *clouds*.

Tabela 3 – Principais características da computação em nuvem, computação em nuvem móvel e computação em nuvem veicular

Características	**CC**	**MCC**	**VCC**
Suporte a recursos móveis	Não	Sim	Sim
Capacidade computacional	Alta	Baixa	Média
Limitação de bateria	Não	Sim	Não
Capacidade de armazenamento	Alta	Baixa	Média
Flexibilidade de recursos	Estática	Estática	Altamente dinâmica
Arquitetura de rede	CS*	CS*	P2P** ou CS*
Recursos físicos	Local ou SR***	Dispositivos móveis ou SR***	Veículos ou SR***

Fonte: (BOUKERCHE; DE GRANDE, 2018), adaptado e traduzido pelo autor.

Nota: *CS - Cliente Servidor, **P2P - *Peer to Peer*, ***SR - Servidores Remotos

Levando em consideração o foco principal do trabalho sendo as Nuvens Veiculares e a Computação em Nuvem Veicular com o uso do paradigma de *Fogs*, é possível identificar algumas características desse modelo, sendo elas: uma alta flexibilidade dos recursos, capacidade computacional mediana, suporte a recursos móveis e não limitação de bateria.

Em relação aos métodos para decisão de alocação de recursos, os métodos classificados aqui como tradicionais são métodos difundidos a mais tempo na literatura e baseados em formulações matemáticas para tomada de decisão. Já os métodos meta-heurísticos, se baseiam no comportamento de animais, teoria evolutiva, inspiração no

ambiente, etc. Esses métodos buscam atuar em cenários nos quais os métodos tradicionais não conseguem obter um bom desempenho, o que faz com que sua utilização seja necessária.

Dessa forma, é possível considerar que os métodos meta-heurísticos abrem novas oportunidades de pesquisa em diversos cenários da computação, especialmente em cenários nos quais os problemas encontrados são considerados problemas de otimização, como no caso da alocação de recursos em Nuvens Veiculares (VCs) com o uso de *Fogs*.

3 ESTADO DA ARTE

Neste capítulo são abordados trabalhos do atual estado da arte relacionados com o tema de alocação e gerenciamento de recursos, tanto com a utilização de algoritmos tradicionais da literatura, quanto com algoritmos meta-heurísticos.

3.1 Trabalhos Relacionados

Mohanty et al. (2018) aplicaram um algoritmo baseado em teoria dos jogos para realizar a otimização da alocação de recursos de nuvem. Os autores consideraram cada veículo no cenário um membro do jogo e possuindo uma série de estratégias. Esse cenário é composto por uma *Cloud* de três níveis sendo uma *Cloud Central*, uma *Roadside Cloud* formada por *Roadside Unit* - RSUs e uma *Vehicular Cloud*. Já em Klaimi, Senouci e Messous (2018), os autores consideram uma aplicação de teoria dos jogos para gerenciamento de recursos em uma *Vehicular Cloud* com a aplicação de uma *Vehicular Fog*. Dessa forma, foi desenvolvido um algoritmo baseado em teoria dos jogos que visa minimizar a latência enquanto otimiza os recursos de computação e utilização de recursos de energia dos veículos.

Em Pereira et al. (2019) e Pereira et al. (2020) foram utilizadas abordagens do AHP para alocação de recursos. Em Pereira et al. (2019), os autores consideraram a alocação de recursos em Nuvens Veiculares utilizando *Fogs* em um cenário de rodovias. Os recursos disponíveis para os veículos no cenário eram largura de banda, consumo de tempo de serviço, armazenamento e processamento. Em Pereira et al. (2020) um mecanismo baseado no AHP para alocação de recursos em uma Rede 5G utilizando Computação Móvel de Borda. O trabalho tem como foco maximizar a quantidade de serviços atendidos e mitigar problemas relacionados a alocação de recursos.

Em Kong et al. (2019), os autores propuseram uma estratégia de alocação de tarefas para múltiplos robôs em *Cloud* utilizando um algoritmo PSO adaptado juntamente com um algoritmo Guloso (*Greedy*). Para busca da solução foi utilizado o PSO com algumas adaptações nas equações de busca e para o cálculo do valor do *fitness* para avaliação das soluções, foi utilizado um algoritmo Guloso em junção ao cálculo da distância euclidiana entre os nós dos robôs e os nós das tarefas.

Sarubbi et al. (2017) desenvolveram um algoritmo híbrido baseado no algoritmo genético e no *Delta Network* chamado Delta-GA2. O algoritmo tem como base os métodos do algoritmo genético, incluindo o conceito de elitismo, mas também se baseia na premissa do *Delta Network* que consiste em avaliar a conectividade entre veículos. O algoritmo foi

aplicado em uma VANET para alocação de RSUs e foi avaliado o quesito de alcance da comunicação desses RSUs com os veículos da rede. No trabalho de Li et al. (2019) os autores desenvolveram um algoritmo melhorado do Algoritmo Genético IGA (*Improved Genetic Algorithm*) para alocação de recursos em redes com dispositivos IoT. Foi considerado a alocação de recursos heterogêneos providos de diversas dispositivos IoT utilizando o paradigma de computação em *Fog* com foco em minimizar o consumo de energia do sistema.

Nos trabalhos de Sun et al. (2019), Jayswal (2020), foram realizadas abordagens utilizando o método meta-heurístico do Algoritmo do Morcego (BAT). Em Jayswal (2020) um método inspirado no algoritmo do morcego foi proposto para alocação de tarefas em uma infraestrutura de nuvem. O trabalho tem como objetivo melhor a performance da nuvem em termos de tempo de execução e tempo de início. Em Sun et al. (2019), os autores utilizaram o algoritmo do morcego modificado como forma de alocar tarefas em uma rede veicular de borda (*Vehicular Edge Computing* - VEC). Como métrica de avaliação utilizaram o tempo de execução, capacidade de *offloading* e peso de sucesso das tarefas. Para comparação, utilizaram um algoritmo baseado nos recursos disponíveis apenas dos veículos chamado de computação local; outro chamado alocação justa em que o algoritmo manda para o *offloading* quando o tempo de execução da tarefa é maior que o veículo pode processar, e por último um chamado alocação por prioridade, em que é levado em consideração o peso da tarefa para solicitação do *offloading*.

Uma abordagem utilizando o algoritmo meta-heurístico de Otimização do Enxame de Partículas (PSO) para alocação de recursos foi desenvolvida por Midya et al. (2018). Os autores consideraram um cenário com uma arquitetura de três camadas em um sistema de transporte inteligente, sendo essas camadas formadas por veículos na camada mais baixa, seguidos por outra de *cloudlets* na camada do meio e na última uma nuvem centralizada. Também foi proposto uma função de cálculo de *fitness* própria pelos autores. Como forma de avaliação, foi utilizado um cenário de simulação com SUMO, *Network Simulator* 3 (NS3) e Matlab, considerando a curva de convergência dos algoritmos avaliados, erro quadrático médio e atraso de entrega de serviço.

Outra abordagem que utilizou um algoritmo baseado em um método meta-heurístico foi em Lieira et al. (2021). Os autores propuseram um algoritmo baseado no comportamento de caça dos lobos cinzentos (Algoritmo GWO). Sua aplicação foi em um um cenário com quatro estruturas de borda se comunicando através de redes móveis 5G. O trabalho buscou maximizar o atendimento dos usuários e realizar a alocação dos recursos dos dispositivos IoT na rede.

3.2 Considerações Parciais

Como forma de relacionar todos os trabalhos, foi desenvolvido uma tabela para resumir os principais pontos abordados pelos trabalhos citados na Seção 3.1. Sendo assim, as principais características como métodos utilizados, tipo do método, estrutura de alocação e problema abordado em cada um dos trabalhos são descritos na Tabela 4.

Tabela 4 – Comparação dos trabalhos abordados no estado da arte

Trabalho	**Método**	**Tipo Método**	**Estrutura de alocação**	**Problema**
Mohanty et al. (2018)	Teoria dos Jogos	Tradicional	*Cloud*	Alocação de recursos
Klaimi, Senouci e Messous (2018)	Teoria dos Jogos	Tradicional	*Cloud* Veicular	Alocação de recursos
Pereira et al. (2019)	AHP	Tradicional	Fog	Alocação de recursos
Pereira et al. (2020)	AHP	Tradicional	MEC	Alocação de recursos
Kong et al. (2019)	PSO + Guloso	Ambos	*Cloud*	Alocação de tarefas
Sarubbi et al. (2017)	IGA	Meta-heurístico	*VANET*	Alocação de RSUs
Li et al. (2019)	GA	Meta-heurístico	Rede IoT	Alocação de recursos
Sun et al. (2019)	BAT	Meta-heurístico	VEC	Alocação de tarefas
Jayswal (2020)	BAT	Meta-heurístico	*Cloud*	Alocação de tarefas
Midya et al. (2018)	PSO	Meta-heurístico	*Cloud*	Alocação de recursos
Lieira et al. (2021)	GWO	Meta-heurístico	*Edge*	Alocação de recursos
ICARUS	BAT	Meta-heurístico	*Fog*	Alocação de recursos

Fonte: Produzido pelo autor.

Os trabalhos relacionados acima apresentam métodos tradicionais para alocação de recursos Mohanty et al. (2018), Klaimi, Senouci e Messous (2018), Pereira et al. (2019), Pereira et al. (2020), métodos meta-heurísticos Sarubbi et al. (2017), Li et al. (2019), Sun et al. (2019), Jayswal (2020), Midya et al. (2018), Lieira et al. (2021) e até mesmo a junção de um tradicional com um meta-heurístico Kong et al. (2019).

Todos os trabalhos utilizam diferentes cenários de rede, desde uma *Cloud* comum, até uma MEC, porém não com o mesmo foco da ideia central deste trabalho. No trabalhos de Pereira et al. (2019), também foi utilizada a *Fog* como estrutura de alocação. Em Pereira et al. (2019) os autores utilizaram um número fixo de *Fogs* (apenas 4), diferente do cenário proposto no ICARUS onde são geradas várias *Fogs* de acordo com o agrupamento dos veículos. Quando se trata da utilização do algoritmo do morcego como base, os trabalhos de Sun et al. (2019), Jayswal (2020) realizaram esse feito. Em Sun et al. (2019) os autores realizaram uma alteração na inicialização da população de morcegos, a qual foi

gerada através do algoritmo K-means, para evitar que os morcegos gerados não estejam todos no mesmo lugar. Já em Jayswal (2020) os autores definiram uma quantidade fixa de iterações (100) para a tentativa de busca da melhor solução. No ICARUS, também foi realizada uma mudança na inicialização da população através do cálculo da distância euclidiana, mas com o intuito de normalizar e indicar a distância que os morcegos estão da presa. Outra diferença em relação ao modelo proposto por Jayswal (2020), é que no ICARUS a quantidade de iterações é determinada pela quantidade de veículos no *cluster*, dessa forma é otimizada o processo de busca pela melhor solução.

Dessa forma, o trabalho proposto vem com a abordagem de utilizar uma técnica baseada no método meta-heurístico do BAT para tomada de decisão no processo de alocação de recursos em uma Fog Veicular. O foco do trabalho é aumentar a quantidade de tarefas alocadas, maximizando o número de recursos alocados e diminuindo o número de tarefas não alocadas.

A utilização do BAT se da devido a sua fácil adaptabilidade a diversos cenários, em específico o desse trabalho no qual há vários veículos para atender uma determinada tarefa, e a geração de múltiplas possíveis soluções (morcegos) considerando todos os veículos disponíveis no *cluster* que são aperfeiçoadas de acordo com o cálculo do valor do *fitness* o que auxilia na escolha do melhor veículo da *Fog* para realizar a alocação da tarefa.

4 MECANISMO META-HEURÍSTICO PARA ALOCAÇÃO DE RECURSOS EM FOGS VEICULARES

Neste capítulo são apresentados os métodos utilizados na pesquisa e no desenvolvimento do mecanismo proposto. O mecanismo consiste na utilização do algoritmo meta-heurístico do morcego (BAT) para realizar a alocação de recursos em um cenário com *Fogs* veiculares.

A utilização do algoritmo BAT como base para o mecanismo proposto se da devido ao fato de que o cenário apresentado utiliza uma metodologia que pode ser abstraída facilmente para sua utilização, sendo que há várias *Fogs* e é necessário selecionar o melhor membro da *Fog* para alocar determinada tarefa/recurso. O BAT gera a população de morcegos (membros da *Fog*) que buscam sua presa (tarefa). Dessa forma, o membro da população considerado melhor (melhor solução), é definido com base no cálculo do valor do *fitness* (função de aptidão) para eleger qual o indivíduo mais adequado.

A organização do capítulo se dá da seguinte forma: na Seção 4.1 é descrito o problema encontrado na alocação de recursos, visão geral do sistema e as duas abordagens utilizadas neste trabalho, *Fogs* sem infraestrutura (comunicação V2V) e *Fogs* com infraestrutura (comunicação V2I). Na Seção 4.2 é descrito o funcionamento do algoritmo para gerenciamento e alocação de recursos. Na Seção 4.3 são descritos os parâmetros da simulação, métricas utilizadas para avaliação e os algoritmos utilizados na comparação. Por último, na Seção 4.4 , são discutidos os resultados obtidos através das avaliações das simulações.

4.1 Gestão de Recursos orientada a Fogs

Para o cenário, foi definido um conjunto de quarteirões ligados por segmentos de estrada formando uma topologia complexa. Nesses segmentos de estrada, os veículos inteligentes equipados cada um com uma unidade de bordo (*On-Board unit* - OBU) circulam com uma localização de partida, destino e trajeto aleatório e individual.

É considerado um conjunto de tarefas T sendo $\{T = 1, ..., n\}$ em que cada tarefa representa um serviço ou necessidade computacional que os veículos podem encontrar durante seus trajetos no ambiente rodoviário, como áudio/video *streaming* ou entretenimento informativo generalizado, sugestões de rotas, mensagens de alarme e aviso, monito-

ramento de tráfego ou troca de mensagens. Para cada tarefa, uma quantidade de recursos computacionais é necessária para seu processamento e execução, sendo esses recursos processamento, memória e armazenamento. Para isso, cada tarefa possui um identificador (id_t), a quantidade de processamento (pr_t), memória (me_t) e armazenamento (st_t) necessários. Sendo essa quantidade recursos necessários diferentes de tarefa para tarefa, como forma de englobar uma gama maior de tipos de tarefas diferentes.

Todos os veículos inteligentes que circulam no cenário possuem sua quantidade de recursos computacionais que podem ou não atender a demanda solicitada pelas tarefas. Assim, é considerado um conjunto de veículos V sendo $\{V = 1, ..., n\}$ em que cada veículo possui seu próprio identificador, capacidade de processamento, memória e armazenamento, respectivamente representados por id_v, pr_v, me_v e, st_v. Todos os veículos têm diferentes quantidades de recursos disponíveis para representar a heterogeneidade de modelos, marcas e tecnologia.

4.1.1 Fogs sem infraestruturas

Uma visão geral do sistema baseado em *Fogs* sem infraestrutura pode ser visto na Figura 5. Neste cenário, os veículos próximos se comunicam através de *beacons* e formam *Fogs* (*clusters*) móveis, pelo cenário.

Figura 5 – Abstração da arquitetura de Fogs sem infraestruturas

Fonte: Produzido pelo autor.

4.1.1.1 Clusterização e seleção de líder do *cluster*

O processo de clusterização neste cenário foi conduzido utilizando apenas comunicação V2V através de um processo de *beaconing* entre veículos e para o critério da eleição de cluster o método de menor identificador (*Lowest ID* - LID). A cada segundo, cada veículo do cenário transmite uma mensagem de *beacon* que chega aos veículos no raio de comunicação, e os veículos que recebem essa mensagem são adicionados a uma tabela de vizinhos pertencente ao remetente desse *beacon*. Cada veículo tem a sua própria tabela de vizinhos que permite conhecimento dos membros que o rodeiam.

A formação do *cluster* é baseada nos vizinhos que cada veículo possui. Uma vez que um veículo escalone uma mensagem de *cluster*, a primeira verificação é se o id do veículo id_m é o id mínimo dos vizinhos, se sim, significa que este veículo é o líder do *cluster* (*cluster head* - CH). Em seguida, é feita outra verificação, se o CH atual designado for igual ao id do veículo, se positivo, o nó transmite uma mensagem informando aos demais membros que ele é o CH. Se ainda não for o CH designado, ele se tornará o CH, iniciará um novo *cluster* e transmitirá uma mensagem para os outros membros.

Quando os veículos no raio de comunicação (vizinhos) recebem uma mensagem de *cluster* (que é definida por um $messageID$), é verificado se o nó possui um CH definido $myCH$, caso contrário, esse nó se junta ao *cluster* do remetente da mensagem. Em seguida, é verificado se o CH do nó *cluster* $myCH$ é maior que o identificador do CH $chID$ da mensagem, se sim, significa que ele recebeu uma mensagem de outro CH. Se o nó é o CH, significa que o *cluster* morreu e o nó se junta ao *cluster* do remetente.

4.1.1.2 Gerenciamento de recursos/veículos

Aqui é descrito o gerenciamento de alocação de recursos/tarefas e manutenção de veículos com o cenário de comunicação V2V.

Para o gerenciamento de alocação de recursos/tarefas, assim que o *cluster* é formado, as solicitações de tarefas começam a chegar a uma taxa de Poisson com $\lambda = 1$ para o CH do *cluster* e inicia-se o processo de alocação de recursos. O CH executa o ICARUS, que é o algoritmo de tomada de decisão, para definir qual veículo será selecionado para atender aquela determinada tarefa entre os membros de seu *cluster*. Após a decisão, o CH envia uma mensagem ao veículo selecionado com a tarefa, para que o veículo possa processar aquela tarefa e retornar a resposta ao CH.

Para o gerenciamento de veículos, seguindo o processo de clusterização e eleição de *cluster heads*, os veículos podem sair do *cluster*, mudar de *cluster* e até mesmo sair do cenário. O rastreamento dessas ações é baseado na comunicação de *beacons* entre os membros do *cluster* e nas respostas que eles recebem dos membros vizinhos.

4.1.2 Fogs baseadas em infraestruturas

O cenário baseado em infraestruturas pode ser visto na Figura 6. Neste cenário, há a utilização de RSUs espalhados pelo mapa, que gerenciam e centralizam o processo de criação de *Fogs*, bem como os veículos.

Figura 6 – Abstração da arquitetura de Fogs baseadas em infraestruturas

Fonte: Produzido pelo autor.

4.1.2.1 Clusterização e manutenção dos *clusters*

Considera-e que os veículos se comunicam com as RSUs que geram múltiplos *clusters* utilizando a comunicação V2I realizada entre RSUs-veículos. Cada veículo é definido como um nó no ambiente de simulação, representado por um vértice no cluster, e as interconexões entre os vértices são consideradas arestas. Portanto, as conexões no ambiente-veículo podem ser expressas usando a teoria dos grafos com pesos. Foram utilizados os algoritmos clássicos de agrupamento/*clusterização* K-means e DBSCAN com conectividade ponderada para inicializar o processo de clusterização dos veículos (COSTA et al., 2020; YAN, 2022).

Para a manutenção dos *clusters*, são utilizados dois processos, a junção e a divisão:

- **Junção:** a rede veicular requer um algoritmo de manutenção de cluster dinâmica e de alta velocidade. O método de junção mescla dois *clusters* adjacentes e de pequeno porte para melhorar a conectividade e a estabilidade geral da rede veicular.
- **Divisão:** o método de divisão divide um cluster veicular grande e de densidade irregular para melhorar a conectividade e a estabilidade geral da rede veicular.

4.1.2.2 Gerenciamento de recursos/veículos

Dado o processo de formação do *cluster*, um primeiro intervalo de 15 segundos é utilizado como forma de esperar os *clusters* se formarem e começar sua atualização, para que assim o processo de alocação de recursos seja iniciado.

Da mesma forma que no V2V, o processo de alocação de recursos/tarefas é iniciado a partir do momento em que o *cluster* encontra-se formado, após isso, as solicitações de tarefas começam a chegar, neste cenário nos RSUs, a uma taxa de Poisson com $\lambda = 1$. Conforme as tarefas vão chegando as RSUs, o processo de seleção do melhor veículo disponível naquele *cluster* é iniciado. Dessa forma, o RSU executa o ICARUS e seleciona o melhor veículo designado para aquela tarefa em questão e envia a tarefa para que o veículo a processe.

Nesse cenário, o RSU é considerado o CH, que direciona diretamente ao veículo selecionado, a solicitação para utilização dos recursos e alocação da tarefa. Os veículos, da mesma forma que no outro cenário, podem entrar no *cluster*, trocar de *cluster* e sair do *cluster* e do cenário.

4.2 Algoritmo para gerenciamento e alocação de recursos

Nesta seção é apresentado o ICARUS, um mecanismo para alocação e gerenciamento de recursos em *Fogs* veiculares. Dado os avanços no desenvolvimento dos Sistemas de Transportes Inteligentes e nas VANETs, novos serviços vem sendo criados e necessitam ser atendidos. Dessa forma, novos mecanismos para realização e gerenciamento da alocação de recursos nessas redes de forma efetiva tornam-se necessários, buscando diminuir a perda de recursos e realizando uma melhor distribuição dos mesmos.

Neste trabalho, utiliza-se o algoritmo meta-heurístico do morcego (BAT), como base do mecanismo de alocação e gerenciamento de recursos, o qual busca identificar o melhor membro da *Fog* para realizar a solicitação de recursos como forma de atender um determinado serviço/tarefa solicitado pela rede. O BAT foi escolhido devido devido a sua fácil adaptabilidade a diversos cenários, principalmente ao de VCs e *Fogs*, a geração de múltiplas possíveis soluções o que abrange uma maior gama de possibilidades (algoritmo baseado em população), e a sua constante verificação no processo de busca da melhor solução, utilizando o valor do cálculo do *fitness*. Dessa forma, a probabilidade de encontrar o membro correto da *Fog* para alocar a tarefa é mais alta dada a constante verificação do melhor *fitness*. Assim, considera-se que para este trabalho os morcegos são os membros do *cluster* (*Fog*) que possuem recursos computacionais (memória, processamento e armazenamento) e suas presas são as tarefas que necessitam serem alocadas/atendidas.

O ICARUS foi baseado/modelado no Algoritmo do Morcego (YANG, 2010; YANG;

GANDOMI, 2012) e no *framework* Evolopy (FARIS et al., 2016), baseado na linguagem de programação Python[1] para algoritmos evolutivos/meta-heurísticos. Dessa forma, ele vem como alternativa aos algoritmos convencionais construídos sobre modelos matemáticos, já que atualmente existem opções baseadas em algoritmos meta-heurísticos que visam propor uma estratégia de busca de problemas mais eficaz, para facilitar a modelagem do problema.

Neste caso, o ICARUS assume que a tarefa é a ("presa") e os ("morcegos") são os membros do cluster da *Fog*. Como resultado, os membros do cluster ("morcegos") usam a ecolocalização para procurar ("caçar") a tarefa mais apropriada ("presa") em um esforço para alocar a tarefa. Ao procurar o membro mais apto da população, conhecido como a melhor solução, o algoritmo realiza um esforço constante para diminuir o valor de aptidão (*fitness)* usando o efeito de ecolocalização, assim como o eco reflete nas paredes da caverna e os morcegos continuam a utilizá-lo para guiá-los até a tarefa. A tentativa de melhorar o valor do *fitness* e atualizar as variáveis de base do algoritmo seguem o mesmo padrão no algoritmo. As seguintes informações sobre a pesquisa ("ecolocalização") são fornecidas:

1. Todos os membros do *cluster* usam o efeito de ecolocalização para pesquisar e localizar sua tarefa;

2. Os membros do *cluster* voam aleatoriamente com velocidade v_i e posição x_i com uma frequência fixa f_{min} variando o comprimento de onda λ e o volume de pulso A_0 para procurar a tarefa. A frequência dos pulsos emitidos e a taxa desses pulsos ($r \in [0, 1]$) podem ser ajustadas de acordo com a proximidade do alvo.

Seguindo a abstração na qual é assumida que os membros da *Fog* são um grupo de morcegos e cada membro (veículo) representa um único morcego no grupo, o efeito de ecolocalização é aplicado com uma frequência específica f, velocidade v, e posição no espaço x dos morcegos correspondentes à presa (tarefa). Esta frequência f é proposta em um intervalo sendo $f \in [0, 2]$, em que $f_{max} = 2$ corresponde ao morcego atacando a presa, e $f_{min} = 0$ em que os morcegos (membros do grupo) estão simplesmente voando (procurando, preparando e esperando a presa). Os valores de frequência f, velocidade v e a posição atual do morcego x são atualizados durante a busca pela solução ótima, também conhecida como o melhor membro do *cluster* (morcego) para alocar a tarefa em questão, é conhecido como movimento de ataque. Os valores de aptidão (*fitness*) servem como único critério para escolher a opção ideal.

A velocidade v é aumentada durante o processo de busca, juntamente com os valores da frequência f dado que o processo de busca iterativa das soluções ótimas já

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

começou. Com a frequência da atualização mais recente, esta função aumenta o valor da melhor solução atual. A posição no espaço x é então criada a partir dessas duas atualizações. Diz-se que o movimento de caça dos morcegos é aleatório, pois eles não têm um padrão de movimento específico, permitindo que eles explorem uma área maior. As seguintes equações foram usadas para calcular os valores atualizados de frequência $(f), velocidade$(v) e as novas soluções (x):

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{4.1}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, \tag{4.2}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{4.3}$$

onde $\beta \in [0, 1]$ é um vetor aleatório formado por distribuição uniforme. A melhor posição (solução) atual é x_*, que será selecionada como definitiva entre as demais após a comparação com todas as soluções possíveis.

De acordo com a Equação 4.4, o algoritmo gera a primeira população usando a distância euclidiana entre os valores dos recursos dos membros do *cluster* e os recursos da tarefa. A etapa inicial do algoritmo facilita a normalização dos valores de entrada do ICARUS. A melhor solução é conhecida como o melhor valor de aptidão para o algoritmo (geralmente o menor número alcançável), e os algoritmos meta-heurísticos normalmente usam uma função de referência para avaliar o valor de aptidão de uma solução. Para realizar o *benchmarking* do valor de *fitness* para o ICARUS, foi utilizado a Função de Ackley, descrita na Equação 4.5, sendo uma das funções mais populares para benchmarking (LIANG; SUGANTHAN; DED, 2005). Em algoritmos evolucionários, esta função é frequentemente usada para cálculo do valor de aptidão. Para gerar os primeiros valores de fitness, o ICARUS utiliza a primeira geração de soluções, que são formadas pela distância euclidiana dos recursos fornecidos pelos veículos e os recursos da tarefa, como parâmetro de entrada para a função de Ackley (Equação 4.5).

$$euclidiana = \sqrt{\sum_{j=1}^{n} (s_{ij} - f_{ij})^2} \tag{4.4}$$

$$\begin{aligned} Ackley = &-20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) \\ &- \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e \end{aligned} \tag{4.5}$$

O Algoritmo 1 exibe uma abstração do ICARUS. O ICARUS usa os membros do cluster (da *Fog*) e seus parâmetros, juntamente com os parâmetros obrigatórios da tarefa para definir o melhor membro da Fog que alocará essa tarefa. Assim, como entrada para o ICARUS, ele recebe o $membrosDoCluster$, que é uma matriz$[N][3]$ (Equação 4.6) na qual cada linha representa um membro do *cluster* (veículo), e cada coluna é o número de recursos que aquele veículo tem disponível (processamento, memória, armazenamento), e a $tarefa$ que precisa ser alocada, que é um vetor[3] (Equação 4.7) com o número de recursos que aquela tarefa requer. Nas primeiras linhas do algoritmo (2 a 5) foram inicializadas as variáveis de controle e recebido os parâmetros. Na quinta linha, a variável *dim* corresponde à dimensão da busca, que é considerada como 3 devido ao número de recursos diferentes considerados (processamento, memória e armazenamento). Nas linhas 6 a 8, é gerado a primeira população/solução, usando a Distância Euclidiana (Equação 4.4), entre os valores dos parâmetros $membrosDoCluster$ e os requisitos $tarefa$. Com esta primeira população, utiliza-se a função de Ackley (Equação 4.5) para gerar os primeiros valores de fitness (linhas 9 a 10). Após a geração dos primeiros valores de fitness, a variável $menorFitness$ recebe o menor fitness gerado (linha 11). Todas as abordagens meta-heurísticas buscam encontrar e minimizar o menor valor de aptidão como alvo da busca. Durante a busca, assume-se que a solução que tem o menor valor de aptidão é a mais apta, bem como a melhor entre a população de busca recém-formada.

$$membrosDoCluster[N][3] = \begin{bmatrix} pr & me & st \\ 7 & 5 & 2 \\ 10 & 5 & 6 \\ 3 & 8 & 4 \\ ... & & \\ 8 & 5 & 3 \end{bmatrix} \tag{4.6}$$

$$tarefa[3] = \begin{bmatrix} 4 & 6 & 8 \end{bmatrix} \tag{4.7}$$

O laço principal do ICARUS ocorre nas linhas (12 a 23). Esta parte do algoritmo representa o ataque dos morcegos à presa. Para isso, é utilizado dois laços, que vão do primeiro membro da população ao último, comparando cada membro com todos os demais membros da população. O algoritmo BAT emprega dois laço porque busca a melhor solução e o menor valor de aptidão para cada membro da solução criado por morcego. Na linha 14, o algoritmo usa as Equações 4.1, 4.2 e 4.3 para atualizar os valores de f, v e x , gerando uma possível solução melhor para aquele membro. Uma comparação é feita nas linhas 15 e 16 para verificar se um valor aleatório gerado naquele momento é maior que a taxa de pulso r_i, se sim, a nova solução daquela iteração é modificada pelo melhor tempo de solução e outro valor aleatório. Na linha 17, um $novoFitness$ é gerado usando a nova solução gerada $novaSolucao$ através da Função de Ackley (Equação 4.5). Uma nova

verificação é feita na linha 18 para verificar se o $novoFitness$ é menor ou igual ao fitness da iteração real $fitness[i]$ e se outro valor aleatório é menor que a intensidade do pulso A_i. Quando isso ocorre, é atualizada a solução atual e o *fitness* atual. Como o algoritmo continua tentando diminuir o valor do *fitness*, outra verificação é feita na linha 21 para verificar se o *fitness* baixou seu valor ou não, caso positivo, atualiza-se a $melhorSolucao$ e a $menorFitness$. Através do $melhorSolucao$ e do $menorFitness$, é possível encontrar o $melhorMembro$ dos membros do *cluster* que o algoritmo decidiu atribuir à tarefa. Por fim, na linha 24 é retornado o $melhorMembro$.

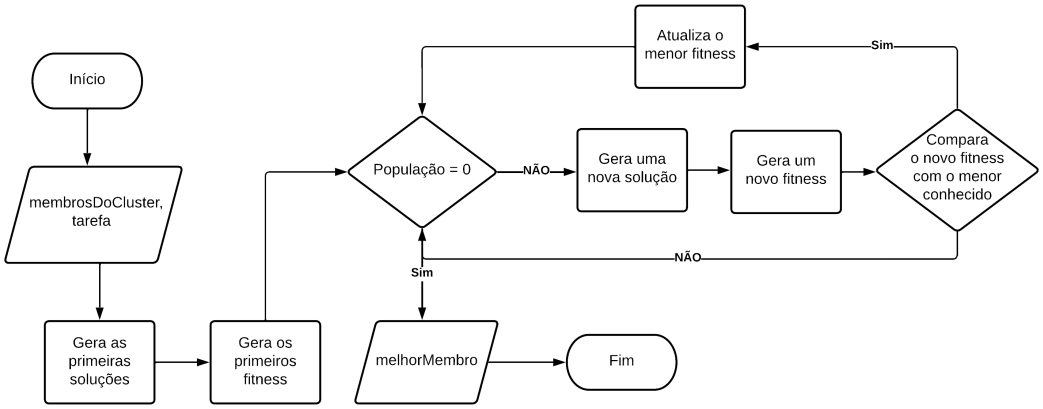
Algoritmo 1: ICARUS

```
   Entrada: membrosDoCluster, tarefa
   Saída: melhorMembro
 1 início
 2     Define a frequência f_i em x_i
 3     Inicia a taxa de pulso r_i e o volume A_i
 4     Populacao ← len(membrosDoCluster)
 5     dim ← 3
 6     for i in range(0, dim) do                    ▷ gera as primeiras soluções (população)
 7         for j in range(0, Populacao) do
 8             Solucao[j,i] ← Euclidiana(membrosDoCluster[j,i], tarefa[i]) (Equação 4.4)
 9     for i in range(0, Populacao) do                  ▷ gera os primeiros valores de fitness
10         fitness[i] ← Ackley(Solucao[i, :]) (Equação 4.5)
11     menorFitness ← min(fitness)      ▷ seleciona o menor fitness dentre os gerados
12     for t in range (0, Populacao) do        ▷ busca pelo melhor membro (ecolocalização)
13         for i in range (0, Populacao) do
14             Gera uma nova solução atualizando f_i, v_i, x_i com as Equações 4.1, 4.2, 4.3
15             se (random() > r_i) então              ▷ verificação baseada na taxa de pulso
16                 novaSolucao[i, :] ← melhorSolucao * random()
17             novoFitness ← Ackley(novaSolucao) ▷ gera um novo possível melhor fitness
18             se (novoFitness ≤ fitness[i]) e (random() < A_i) então        ▷ compara o novo
                fitness com o fitness da iteração
19                 Solucao[i, :] ← novaSolucao
20                 fitness[i] ← novoFitness
21             se (novoFitness ≤ menorFitness) então       ▷ verifica se o novo fitness é
                melhor do que o conhecido
22                 melhorSolucao ← novaSolucao
23                 menorFitness ← novoFitness
24     retorna melhorMembro
```

Como forma de tentar exemplificar melhor o Algoritmo 1, é apresentado na Figura 7 um fluxograma simplificado do ICARUS. No início do fluxograma, o ICARUS recebe os membros do *cluster* e a tarefa. Após isso, as primeiras soluções e os primeiros valores de *fitness* são gerados. Então, é iniciado um laço de repetição que utiliza a população de morcegos como base para tentar encontrar um melhor valor de *fitness*, logo uma melhor solução. Dessa forma, a cada iteração, uma nova solução é gerada, um novo *fitness* é gerado e comparado com o menor valor de *fitness* conhecido. Caso o novo *fitness* seja menor que o menor conhecido, o menor *fitness* conhecido é atualizado e esse processo

continua até o laço terminar. Após o término do laço e a escolha do menor *fitness* (melhor solução), o melhor membro do *cluster* para atender a tarefa designada é retornado.

Figura 7 – Fluxograma do ICARUS



Fonte: Produzido pelo autor.

4.3 Simulação

Nesta seção são apresentados os experimentos conduzidos para avaliação do mecanismo proposto e são discutidos as diferentes configurações de cenários de aplicação, técnicas de comparação, métricas de avaliação e os resultados obtidos.

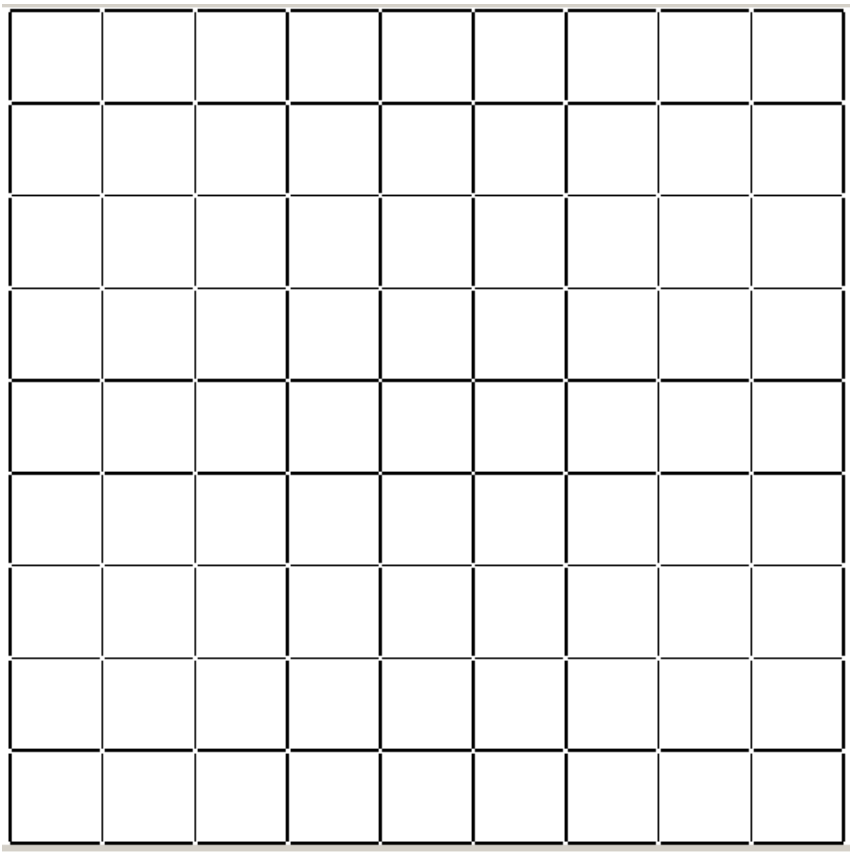
4.3.1 Configurações dos cenários e características das simulações

Os principais parâmetros das simulações podem ser observados na Tabela 5. Para a realização das simulações, foi utilizado o framework de simulação veicular (*Vehicular Network Simulation Framework* - VEINS), que combina um simulador de rede com um simulador de mobilidade. O *Objective Modular Network Testbed* em C++ (OMNet++) (VARGA, 2010) coordena toda a parte complexa de comunicação que suporta as trocas de mensagens entre os veículos e as infraestruturas. A simulação do movimento dos nós (veículos) foi feita pela Simulação de Mobilidade Urbana (*Simulation of Urban Mobility* - SUMO) (LOPEZ et al., 2018).

Como forma de representar ao máximo um cenário real na simulação, o SUMO gera uma mobilidade aleatória para todos os carros, designando a cada um uma rota, origem e destino aleatórios. Foram analisadas diferentes densidades máximas veiculares (160, 330, .., 1600) nas quais o tempo de simulação utilizado foi de 500 segundos, para cada cenário, densidade específica e algoritmo testado. As simulações foram executadas 33 vezes cada e os resultados foram gerados baseados na média aritmética dessas rodadas de simulação com um intervalo de confiança de 95%. Os veículos foram posicionados

em um cenário de uma *grid* de 2700m x 2700m (9 x 9), que representa um conjunto de quarteirões ligados entre si em que a densidade veicular varia com os valores mostrados anteriormente (Figura 8).

Figura 8 – Mapa utilizado nas simulações (Grid 9 x 9)



Fonte: SUMO.

Foram considerados 3 diferentes cenários de simulação:

- **Cenário 1:** utiliza-se comunicação V2V através de *beacons* entre os veículos com o algoritmo de clusterização LID que considera o menor identificador de cada veículo como forma de selecionar o líder do *cluster*;
- **Cenários 2 e 3:** neste cenário há 7 RSUs espalhadas pelo mapa em posições estratégicas. Cada RSU executa o algoritmo de clusterização e agrupa os veículos no seu raio de comunicação. A comunicação é feita no formato V2I entre veículos e RSUs e para a clusterização foram testados os algoritmos DBSCAN e K-means.

4.3.2 Técnicas de comparação

Como forma de validar o mecanismo proposto, o ICARUS foi comparado com outras técnicas heurísticas e meta-heurísticas utilizadas por diferentes autores na oti-

Tabela 5 – Parâmetros das simulações

Parâmetro	**Valor**
Chegada de Tarefas	Distribuição de Poisson
Recursos das Tarefas	rand (1, 10)
Recursos dos Veículos	rand (1, 10)
Densidade Veicular Máxima	160, 330, ..., 1600
Velocidade Máxima dos Veículos	13,90 m/s
Modelo Físico	IEEE 802.11p
Raio de Comunicação dos Veículos	400m
Poder de Transmissão	20mW
Tipo de Cenário	*Grid* 9 x 9
Tamanho do Cenário	2700m x 2700m
Número de *runs*	33 vezes/cada
Tempo de Simulação	500 segundos
V2V	
Política de Clusterização	Menor ID (Lowest ID - LID)
V2I	
Política de Clusterização 1	*Density-based spatial clustering of applications with noise* - DBSCAN
Política de Clusterização 2	K-means
Número de RSUs	7
Raio de Comunicação das RSUs	400m

Fonte: Produzido pelo autor.

mização/decisão de alocação de recursos, como nos trabalhos de (YANG et al., 2017), (LIEIRA et al., 2022), (LIEIRA et al., 2022) e (PEREIRA et al., 2019). Esses trabalhos utilizaram as seguintes técnicas respectivamente: Greedy, Algoritmo de Otimização da Baleia (*Whale Optimization Algorithm* - WOA), PSO e AHP. A seguir, são descritos as principais características de cada método:

- **Greedy (Guloso):** o algoritmo guloso é considerado um algoritmo de simples tomada de decisão. Dada a uma determinada *Fog*, ele seleciona o primeiro veículo disponível no *cluster*, sem levar em consideração os recursos disponíveis do veículo ou os recursos que a tarefa necessita. Seu foco principal é atender o mais rápido possível a solicitação de alocação de recursos;

- **WOA:** o algoritmo WOA tem como base o algoritmo meta-heurístico de otimização da baleia. Esse método considera o comportamento de caça das baleias jubartes, as quais utilizam uma técnica chamada de rede de bolhas que é a base para o processo de busca do algoritmo;

- **PSO:** o algoritmo PSO utiliza como base o algoritmo meta-heurístico de otimização por enxame de partículas. A base do algoritmo é o comportamento de enxames de pássaros e cardumes de peixes e seu processo de busca é baseado no comportamento

social dos animais entre si em que cada elemento do bando é considerado uma partícula.

- **NANCY (AHP):** o algoritmo NANCY é baseado no método tradicional do AHP que baseia-se em um método matemático de multicritério conhecido como estratégia de seleção elitista, em que considera uma matriz de pesos de importância definidos para cada critério da alocação.

4.3.3 Métricas de avaliação

Para a avaliação, três métricas foram consideradas como forma de analisar o ICARUS e seus concorrentes, número de tarefas alocadas ou atendidas, número de tarefas não alocadas e quantidade de recursos alocados.

Para tarefas alocadas, são consideradas as tarefas que o algoritmo de tomada de decisão selecionou um veículo com uma quantidade de recursos computacionais disponíveis equivalente a quantidade de recursos necessários da tarefas ou maior.

Para tarefas não alocadas, são consideradas as tarefas perdidas e negadas pelos veículos. As tarefas perdidas ocorrem quando, por algum motivo, seja ele colisão de pacotes, distanciamento do raio de comunicação ou saída do *cluster*, não são entregues aos veículos/infraestruturas ou suas respostas (positivas ou negativas) não são recebidas pelo controlador. Para tarefas negadas, são consideradas as tarefas nas quais o método de tomada de decisão selecionou um veículo cujo poder computacional era insuficiente para realizar o atendimento da tarefa selecionada.

Por último, para a quantidade de recursos alocados, é realizado a soma dos recursos de processamento (pr), memória (me) e armazenamento (ar) utilizados em cada tarefa considerada alocada.

4.4 Resultados

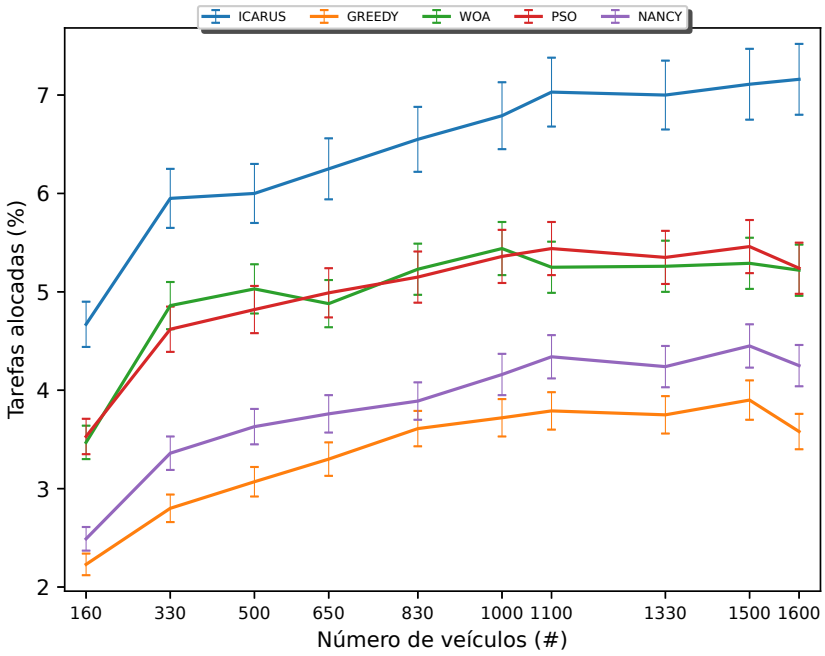
Nesta seção são apresentados os resultados obtidos nos três cenários propostos, V2V com o clusterização através do LID, V2I com clusterização utilizando o DBSCAN e por último V2I com clusterização utilizando o K-means.

4.4.1 V2V

Os resultados obtidos através do cenário V2V podem ser observados nas Figuras 9, 10 e 11. Na Figura 9 é demonstrado os resultados referentes ao número de tarefas alocadas pelos algoritmos. É possível observar que o ICARUS foi mais eficiente que seus concorrentes, superando-os no processo de alocação de tarefas. De forma significativa, o

ICARUS desempenhou um melhor papel na alocação em todas as diferentes densidades testadas, sendo que na menor densidade (≈ 160 veículos) alocou cerca de 4,67% das tarefas e na segunda densidade testada (≈ 350 veículos), já superou todos os outros algoritmos mesmo em densidades maiores (≈ 1600 veículos) com 5,95% de alocação. Os algoritmos WOA e PSO desempenharam um papel considerado similar, seguidos pelas outras duas técnicas consideradas tradicionais, NANCY e Greedy.

Figura 9 – Gráfico de tarefas alocadas (V2V)

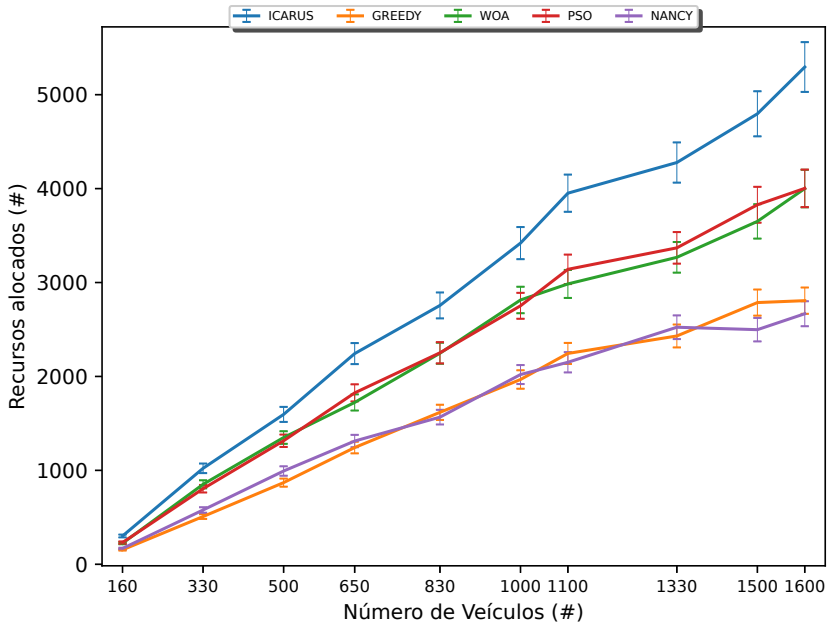


Fonte: Produzido pelo autor.

O número de recursos alocados por cada algoritmo pode ser observado na Figura 10. Novamente, o ICARUS desempenhou um melhor papel, aproveitando melhor os recursos das tarefas e consequentemente alocando mais recursos. Os algoritmos NANCY e Greedy tiveram performance similar utilizando menos recursos das tarefas, seguidos pelos algoritmos WOA e PSO que também tiveram performance similar. Considerando a densidade com maior número de veículos (≈ 1600 veículos), o ICARUS obteve cerca 5294,12 unidades de recurso alocadas, seguidos por PSO (4002,42), WOA (4000,39), Greedy (2806,85) e NANCY (2667,55).

Para as tarefas não alocadas, os resultados podem ser observados na Figura 11. Os algoritmos Greedy e NANCY tiveram um maior número de tarefas não alocadas respectivamente, seguidos pelos algoritmos WOA e PSO que durante o processo se alternaram em alguns momentos, e em último lugar o ICARUS, mostrando que ele obteve um desempenho melhor dado ao menor número de tarefas não alocadas. É possível observar que mesmo na densidade inicial (≈ 200 veículos), considerando uma quantidade baixa de

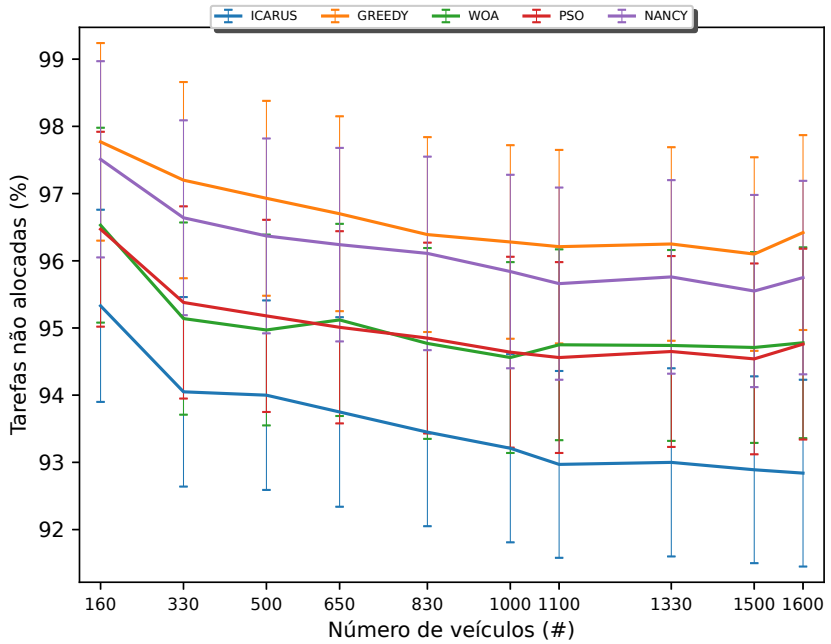
Figura 10 – Gráfico de utilização de recursos (V2V)



Fonte: Produzido pelo autor.

automóveis, o ICARUS conseguiu um resultado de apenas 95,33% tarefas não alocadas, enquanto o pior caso, o algoritmo Greedy, não alocou 97,77% das tarefas disponíveis.

Figura 11 – Gráfico de tarefas não alocadas (V2V)

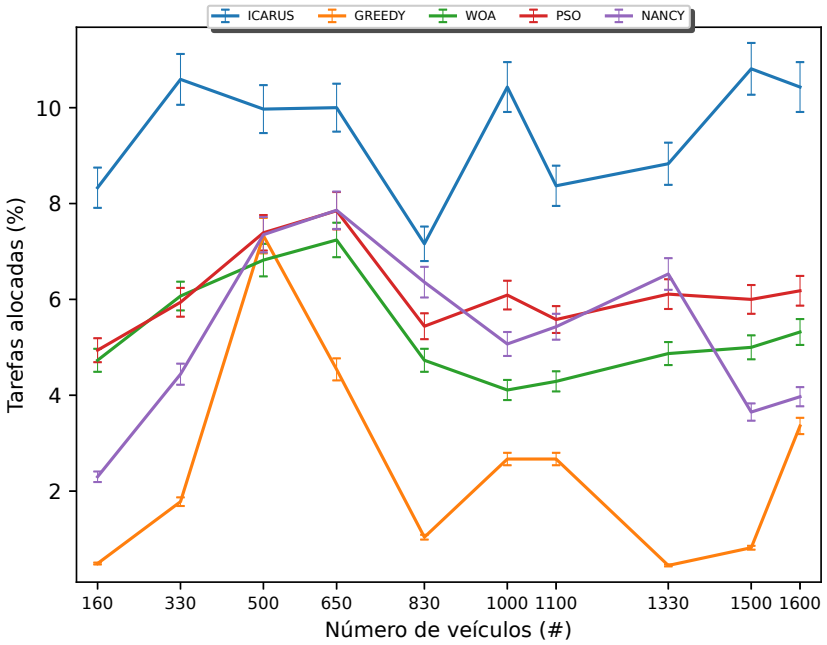


Fonte: Produzido pelo autor.

4.4.2 V2I - DBSCAN

Os resultados do cenário V2I com o algoritmo de clusterização DBSCAN são apresentados através das Figuras 12, 13 e 14. O número de tarefas alocados por algoritmo pode ser visto na Figura 12. Levando em consideração todas as densidades de veículos desse cenário, o ICARUS demonstrou-se mais eficiente, alocando mais tarefas do que os outros algoritmos. Pode-se dizer que o algoritmo que obteve o menor desempenho foi o algoritmo Greedy que desempenhou um pior papel no processo de alocação de tarefas, com exceção em uma densidade particular (≈ 500 veículos), que atingiu um valor aproximado aos outros algoritmos comparados (WOA, NANCY e PSO).

Figura 12 – Gráfico de tarefas alocadas (V2I - DBSCAN)

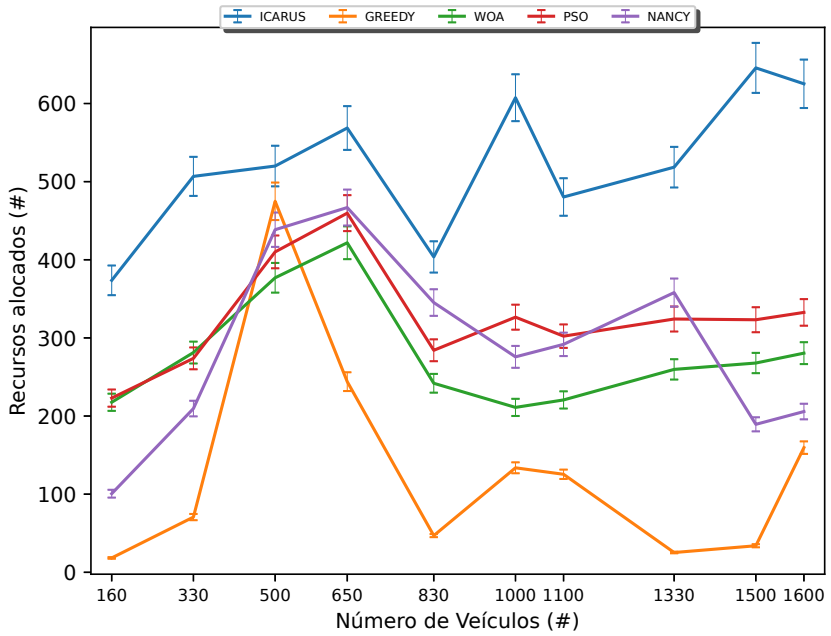


Fonte: Produzido pelo autor.

No quesito número de recursos alocados (Figura 13), o ICARUS obteve um melhor desempenho, chegando a atingir o valor 625,24 (≈ 1600 veículos), seguido do PSO (332,64), WOA (280,48), NANCY (205,75) e Greedy (159,55). Durante a maior parte das densidades, os algoritmos PSO, WOA e NANCY obtiveram resultados parecidos, com exceção do algoritmo Greedy que obteve o pior desempenho.

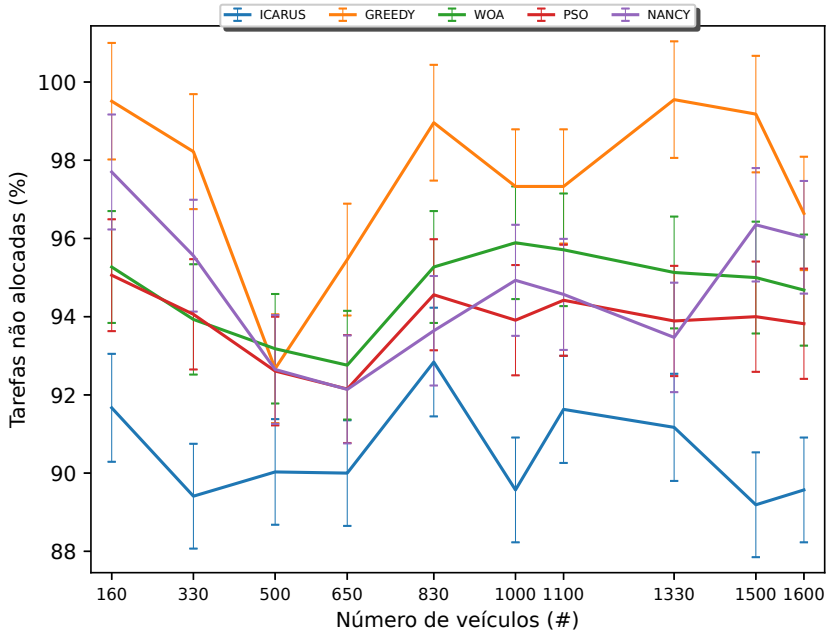
Para tarefas não alocadas, os resultados são demonstrados na Figura 14. É possível observar que o algoritmo Greedy obteve o maior número de tarefas não alocadas na maior parte das densidades, com uma média de 97,49% considerando todas as densidades, seguido pelo NANCY (94,70%), WOA (94,68%) PSO (94,68%) e o ICARUS (90,51%).

Figura 13 – Gráfico de utilização de recursos (V2I - DBSCAN)



Fonte: Produzido pelo autor.

Figura 14 – Gráfico de tarefas não alocadas (V2I - DBSCAN)



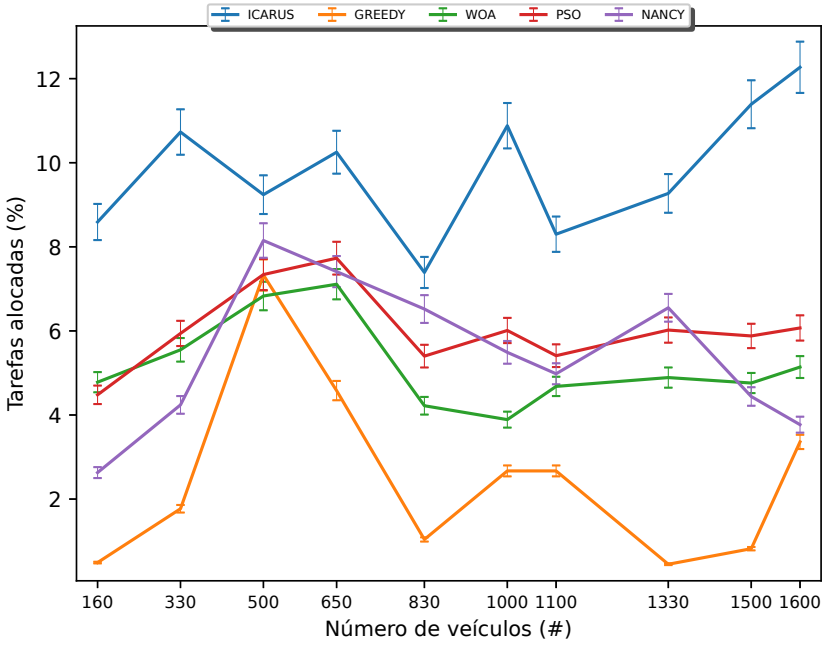
Fonte: Produzido pelo autor.

4.4.3 V2I - K-means

Os resultados com o algoritmo de clusterização K-means são demonstrados nas Figuras 15, 16 e 17. O gráfico de tarefas alocadas (Figura 15) demonstra novamente que o

ICARUS alocou mais tarefas que os outros algoritmos comparados. Em relação ao cenário com o algoritmo de clusterização DBSCAN, com o uso do K-means o ICARUS obteve um melhor desempenho que no outro cenários em algumas das densidades, em específico as maiores densidades (1,84% maior com ≈ 1600 veículos) dado a uma melhor formação nos *clusters* por parte do algoritmo.

Figura 15 – Gráfico de tarefas alocadas (V2I - K-means)



Fonte: Produzido pelo autor.

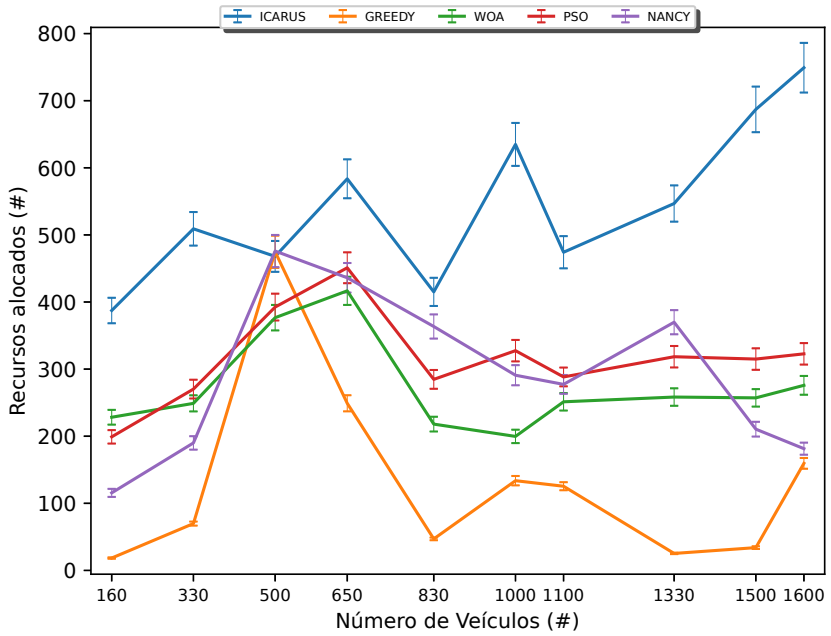
Para o número de utilização de recursos (Figura 16), o ICARUS obteve um desempenho médio de 545,50 unidades de recursos alocados, seguido pelo PSO (316,94), NANCY (291,11) WOA (273,09) e Greedy (133,71). No geral, a performance dos algoritmos foi similar quando comparado ao cenário com o DBSCAN.

Por último, para o número de tarefas não alocadas, os resultados são demonstrados na Figura 17. O ICARUS demonstrou-se mais eficiente que os outros algoritmos dado o fato de possuir um número menor de tarefas não alocados que seus concorrentes. Considerando as menores porcentagens de tarefas não alocadas pelos algoritmos ao longo das diferentes densidades, o ICARUS atingiu 87,73%, seguido do NANCY 91,85%, PSO 92,27%, Greedy 92,67% e por último o WOA 92,89%.

4.5 Considerações parciais

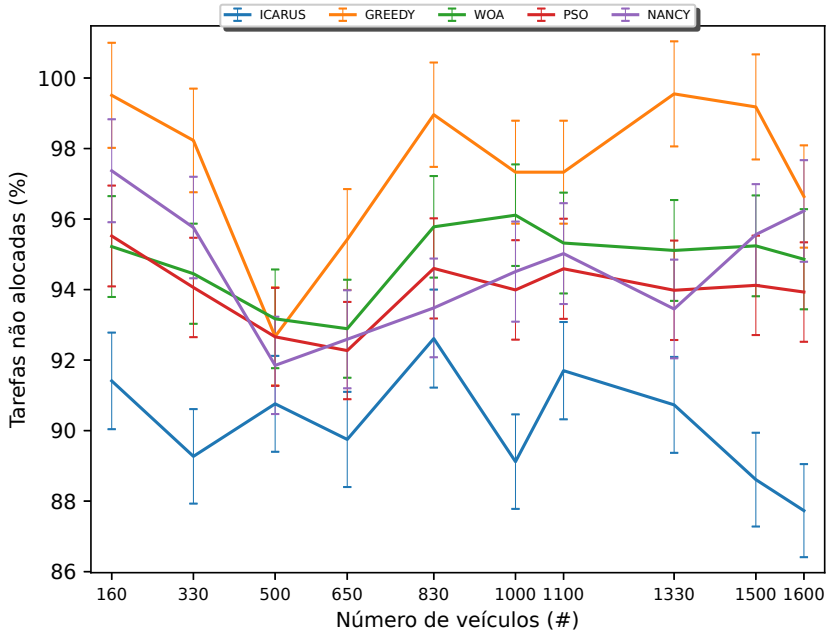
De forma geral, é possível observar um desempenho melhor do ICARUS nos 3 cenários testados. Quando considerado o cenário V2V, é possível observar uma variação constante nos valores dos resultados ao comparar com os cenários V2I que apresentam

Figura 16 – Gráfico de utilização de recursos (V2I - K-means)



Fonte: Produzido pelo autor.

Figura 17 – Gráfico de tarefas não alocadas (V2I - K-means)



Fonte: Produzido pelo autor.

uma oscilação maior nos resultados. Isso se da devido ao fato de no cenário V2V, os veículos criarem *clusters* móveis durante o processo de alocação, em relação aos *clusters* fixos do cenário V2I, que dependem de um raio de comunicação fixo da RSU e dessa forma problemas relacionados a colisão de pacotes e raio de comunicação podem ocorrer com uma

maior frequência. Mesmo com uma oscilação maior nos resultados dos cenários de V2I, o desempenho dos algoritmos no geral foi melhor nos cenários V2I em relação a porcentagem de tarefas alocadas e não alocadas. O ICARUS obteve um desempenho melhor do que seus concorrentes, devido ao fato de levar em consideração todos os membros do *cluster* e seus recursos para a tomada de decisão, bem como os recursos necessários da tarefa. Também, a utilização do cálculo da distância euclidiana para inicialização da população (que apresenta um impacto de fundamental importância no desempenho) e a constante tentativa de melhora do função de aptidão (valor de *fitness*) durante o processo de busca do algoritmo, foram de fundamental importância para a obtenção desses resultados.

O mecanismo proposto, ICARUS, foi comparado com outros algoritmos considerados tradicionais Greedy e NANCY, e também com outros métodos meta-heurísticos WOA e PSO. Foram utilizados 3 diferentes cenários com diferentes tipos de comunicação e de algoritmos de clusterização. Dessa forma, o ICARUS obteve um melhor desempenho superando os algoritmos testados em relação a número de tarefas alocadas, utilização de recursos alocados e número de tarefas não alocadas. Os resultados das simulações demonstram que o mecanismo meta-heurístico baseado no algoritmo do morcego, aqui chamado de ICARUS, demonstrou-se mais eficaz no processo de alocação de tarefas e recursos que os algoritmos considerados tradicionais e até mesmo alguns métodos meta-heurísticos nos diferentes cenários testados.

5 CONCLUSÃO

Com a evolução da tecnologia na área veicular, os ITSs pode oferecer novos e melhores serviços. Os desafios na prestação desses serviços vêm do problema de alocação de recursos computacionais de quais veículos ou recursos podem ser atribuídos a um determinado tipo de serviço e uma determinada tarefa. O uso de *fogs* veiculares e paradigmas de nuvem vêm para ajudar a solucionar esses problemas com a redução da latência na comunicação e compartilhamento de recursos. Um algoritmo adequado de tomada de decisão no processo de alocação de tarefas é fundamental para o funcionamento do sistema.

Neste trabalho, foi apresentado o ICARUS, um mecanismo meta-heurístico baseado no algoritmo bio-inspirado do morcego para realizar a tomada de decisão na alocação de tarefas. O ICARUS aplica uma meta-heurística baseada no efeito de ecolocalização dos morcegos para encontrar a melhor solução. Três diferentes cenários de simulação foram utilizados como forma de validar o mecanismo proposto, um cenário V2V com o algoritmo de clusterização LID, um V2I com o algoritmo de clusterização DBSCAN e por último outro cenário V2I com o algoritmo de clusterização K-means. Os resultados da simulação mostraram que, em geral, ICARUS teve um desempenho melhor do os algoritmos meta-heurísticos WOA e PSO e que os algoritmos tradicionais NANCY (AHP) e Greedy.

5.1 Principais contribuições

Nesta seção são listadas as principais contribuições atingidas deste projeto de mestrado durante o seu desenvolvimento:

- Proposta de utilização de um algoritmo meta-heurístico para o processo de tomada de decisão na alocação de recursos e tarefas em um ambiente de *fogs* veiculares, utilizando os formatos baseados em *fogs* sem infraestruturas e baseados com infraestruturas;
- Desenvolvimento do ICARUS realizar a alocação de recursos e tarefas;
- Implantar um ambiente de simulação utilizando um simulador de rede e um *framework* veicular, juntamente com mobilidades aleatórias para teste do mecanismo proposto;
- Maximização do número de tarefas alocadas e diminuição do número de tarefas não alocadas; e

- Maximização do número de recursos utilizados na alocação de tarefas.

5.2 Produção Científica

- Quessada et al. (2021). "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."
- Quessada et al. (2022a). "ARCANE: Algoritmo meta-heurístico para alocação de tarefas em nuvens veiculares. In: Anais do VI Workshop de Computação Urbana (CoUrb 2022). Sociedade Brasileira de Computação - SBC, 2022."
- Quessada et al. (2022b). "Towards bat bio-inspired decision-making for task allocation in vehicular fogs. In: 2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS). [S.l.: s.n.], 2022. p. 298–305."

5.3 Trabalhos Futuros

Para trabalhos futuros, pretende-se avaliar outros algoritmos meta-heurísticos nas análises de desempenho com o ICARUS. Incluir novas métricas de desempenho relacionadas à rede para que as consequências de diferentes densidades possam ser melhor estudadas e discutidas. Abranger diferentes mobilidades e regiões urbanas reais nas próximas simulações, considerando uma gama de parâmetros de algoritmos para análises de desempenho abrangentes em resultados mais confiáveis e melhores tomadas de decisão.

REFERÊNCIAS

ALAKBAROV, R. G.; ALAKBAROV, O. R. Mobile clouds computing: Current state, architecture and problems. In: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. [S.l.: s.n.], 2017. p. 1–6.

AUTOO, S. d. M. *Total de Emplacamentos de Veículos Leves no Brasil por Mês e Ano*. 2021. Acessado em: 12 de set. de 2021. Disponível em: https://www.autoo.com.br/emplacamentos/.

BAHAMOU, S.; OUADGHIRI, M. D. E.; BONNIN, J.-M. When game theory meets vanet's security and privacy. In: *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media*. New York, NY, USA: Association for Computing Machinery, 2016. (MoMM '16), p. 292–297. ISBN 9781450348065. Disponível em: https://doi.org/10.1145/3007120.3007168.

BELLO, S. A. et al. Cloud computing in construction industry: Use cases, benefits and challenges. *Automation in Construction*, v. 122, p. 103441, 2021. ISSN 0926-5805. Disponível em: https://www.sciencedirect.com/science/article/pii/S0926580520310219.

BOUKERCHE, A.; DE GRANDE, R. E. Vehicular cloud computing: Architectures, applications, and mobility. *Computer Networks*, v. 135, p. 171–189, 2018. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128618300057.

CHEN, F. et al. Code-based computation offloading in vehicular fog networks. In: *2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. [S.l.: s.n.], 2021. p. 1–5.

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

DAS, R.; GOSWAMI, S.; KONAR, A. Relationship between nash equilibria and pareto optimal solutions for games of pure coordination. In: *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. [S.l.: s.n.], 2019. p. 1–7.

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.

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*. [S.l.: s.n.], 1995. p. 39–43.

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.

FELLAH, H.; MEZIOUD, C.; BATOUCHE, M. C. Mobile cloud computing: Architecture, advantages and security issues. In: . New York, NY, USA: Association for Computing Machinery, 2020. (NISS2020). ISBN 9781450376341. Disponível em: https://doi.org/10.1145/3386723.3387880.

GEEM, Z. W.; KIM, J. H.; LOGANATHAN, G. A new heuristic optimization algorithm: Harmony search. *SIMULATION*, v. 76, n. 2, p. 60–68, 2001. Disponível em: https://doi.org/10.1177/003754970107600201.

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.

HAMDI, A. M. A.; HUSSAIN, F. K.; HUSSAIN, O. K. Task offloading in vehicular fog computing: State-of-the-art and open issues. *Future Generation Computer Systems*, v. 133, p. 201–212, 2022. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X22000942.

IBGE, I. B. de Geografia e E. *Frota de Veiculos*. 2021. Acessado em: 16 de ago. de 2022. Disponível em: https://cidades.ibge.gov.br/brasil/pesquisa/22/28120?ano=2021.

JAISWAL, N. K. *Military Operations Research: Quantitative Decision Making*. [S.l.]: Springer, 1997. 209-232 p.

JAYSWAL, A. K. Efficient task allocation for cloud using bat algorithm. In: *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. [S.l.: s.n.], 2020. p. 186–190.

JHA, A. A.; JAIN, S.; THENMALAR, S. Survey on grey wolf algorithm in resource allocation. *International Journal of Pure and Applied Mathematics - Special Issue*, v. 118, p. 201 – 206, 2018. ISSN 1314-3395 (on-line version).

KATOCH, S.; CHAUHAN, S. S.; KUMAR, V. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, Springer Science and Business Media LLC, v. 80, n. 5, p. 8091–8126, out. 2020. Disponível em: https://doi.org/10.1007/s11042-020-10139-6.

KHALEDI, M.; KHALEDI, M.; KASERA, S. K. Profitable task allocation in mobile cloud computing. In: *Proceedings of the 12th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. New York, NY, USA: Association for Computing Machinery, 2016. (Q2SWinet '16), p. 9–17. ISBN 9781450345040. Disponível em: https://doi.org/10.1145/2988272.2988281.

KLAIMI, J.; SENOUCI, S.-M.; MESSOUS, M.-A. Theoretical game approach for mobile users resource management in a vehicular fog computing environment. In: *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*. [S.l.: s.n.], 2018. p. 452–457.

KONG, X. et al. Multi-robot task allocation strategy based on particle swarm optimization and greedy algorithm. In: *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*. [S.l.: s.n.], 2019. p. 1643–1646.

KUHN, S. Prisoner's Dilemma. In: ZALTA, E. N. (Ed.). *The Stanford Encyclopedia of Philosophy*. Winter 2019. [S.l.]: Metaphysics Research Lab, Stanford University, 2019.

KUMAR, T. A. et al. Reinforced resource management in vehicular fog computing using deep beacon power control protocol. *International Journal of Web and Grid Services*, Inderscience Publishers (IEL), v. 17, n. 4, p. 371–388, 2021.

LAKHAN, A. et al. Cost-efficient mobility offloading and task scheduling for microservices iovt applications in container-based fog cloud network. *Cluster Computing*, Springer, v. 25, n. 3, p. 2061–2083, 2022.

LI, G.; WANG, H. Improved harmony search algorithm for global optimization. In: *2018 Chinese Control And Decision Conference (CCDC)*. [S.l.: s.n.], 2018. p. 864–867.

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

LI, Y. et al. Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-stackelberg approach. *IEEE Internet of Things Journal*, v. 9, n. 17, p. 15969–15982, 2022.

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

LIEIRA, D. D. et al. 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)*. [S.l.: s.n.], 2022. p. 1–6.

LIEIRA, D. D. et al. Algorithm for 5g resource management optimization in edge computing. ieee latin america transactions. *IEEE Latin America Transactions*, v. 19, n. 10, 2021.

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)*. [S.l.: s.n.], 2022. p. 283–290.

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/.

MAKWE, A.; KANUNGO, P. Scheduling in cloud computing environment using analytic hierarchy process model. In: *2015 International Conference on Computer, Communication and Control (IC4)*. [S.l.: s.n.], 2015. p. 1–4.

MAO, W. et al. Data-driven capacity planning for vehicular fog computing. *IEEE Internet of Things Journal*, v. 9, n. 15, p. 13179–13194, 2022.

MENEGUETTE, R.; BOUKERCHE, A.; DE GRANDE, R. SMART: an efficient resource search and management scheme for vehicular cloud-connected system. In: *2Proceedings of the IEEE Global Communications Conference: Mobile and Wireless Networks*. Washington, USA: [s.n.], 2016.

MENEGUETTE, R. I.; BOUKERCHE, A. A cooperative and adaptive resource scheduling for vehicular cloud. In: *2017 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.: s.n.], 2017. p. 398–403.

MENEGUETTE, R. I.; DE GRANDE, R.; LOUREIRO, A. A. *Intelligent Transport System in Smart Cities*. [S.l.]: Springer, 2018. 182 p.

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)*. [S.l.: s.n.], 2018. p. 12–17.

MOHANTY, P. et al. Dynamic resource allocation in vehicular cloud computing systems using game theoretic based algorithm. In: *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. [S.l.: s.n.], 2018. p. 476–481.

NASH, J. Non-cooperative games. *Annals of Mathematics*, Annals of Mathematics, v. 54, n. 2, p. 286–295, 2021/05/13/ 1951. ISSN 0003486X. Full publication date: Sep., 1951. Disponível em: http://www.jstor.org/stable/1969529.

NEUMANN, J. V. *Theory of Games and Economic Behavior*. [S.l.]: Princeton University Press, 1944. 776 p.

OLARIU, S.; KHALIL, I.; ABUELELA, M. Taking vanet to the clouds. *International Journal of Pervasive Computing and Communications*, Emerald Group Publishing Limited, v. 7, n. 1, p. 7–21, Jan 2011. ISSN 1742-7371. Disponível em: https://doi.org/10.1108/17427371111123577.

PAUL, A. et al. Chapter 2 - intelligent transportation systems. In: Paul, A. et al. (Ed.). *Intelligent Vehicular Networks and Communications*. Elsevier, 2017. p. 21–41. ISBN 978-0-12-809266-8. Disponível em: https://www.sciencedirect.com/science/article/pii/B9780128092668000028.

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)*. [S.l.: s.n.], 2019. p. 1–6.

QUESSADA, M. S. et al. ARCANE: Algoritmo meta-heurístico para alocação de tarefas em nuvens veiculares. In: *Anais do VI Workshop de Computação Urbana (CoUrb 2022)*. Sociedade Brasileira de Computação - SBC, 2022. Disponível em: https://doi.org/10.5753/courb.2022.223498.

QUESSADA, M. S. et al. Towards bat bio-inspired decision-making for task allocation in vehicular fogs. In: *2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. [S.l.: s.n.], 2022. p. 298–305.

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.

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.

SAJJAD, I.; SHARMA, R.; GERDES, R. A game-theoretic approach and evaluation of adversarial vehicular platooning. In: . New York, NY, USA: Association for Computing Machinery, 2017. (SCAV'17), p. 35–41. ISBN 9781450349765. Disponível em: https://doi.org/10.1145/3055378.3055383.

SARUBBI, J. a. M. et al. A genetic algorithm for hybrid vanets with synchronous communication. In: . New York, NY, USA: Association for Computing Machinery, 2017. (GECCO '17), p. 303–304. ISBN 9781450349390. Disponível em: https://doi.org/10.1145/3067695.3076031.

SHIJU GEORGE, S.; SUJI PRAMILA, R. A review of different techniques in cloud computing. *Materials Today: Proceedings*, 2021. ISSN 2214-7853. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214785321019015.

SHRESTHA, R.; BAJRACHARYA, R.; NAM, S. Y. Challenges of future vanet and cloud-based approaches. *Wireless Communications and Mobile Computing*, 2018.

SON, D. B. et al. Fuzzy deep q-learning task offloading in delay constrained vehicular fog computing. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2021. p. 1–8.

SOUSA, L. *O dilema do prisioneiro e a Lava-Jato*. 2016. Acessado em: 27 de set. de 2021. Disponível em: https://agcomunique.wordpress.com/2016/11/17/o-dilema-do-prisioneiro-e-a-lava-jato/.

SRIVASTAVA, S.; SAHANA, S. K. Application of bat algorithm for transport network design problem. *Applied Computational Intelligence and Soft Computing*, Hindawi, v. 2019, p. 9864090, Jan 2019. ISSN 1687-9724. Disponível em: https://doi.org/10.1155/2019/9864090.

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.

THENGADE, A.; DONDAL, R. Genetic algorithm – survey paper. *IJCA Proc National Conference on Recent Trends in Computing, NCRTC*, v. 5, 01 2012.

TOLKIEN, J. R. R. *The fellowship of the ring*. London, England: HarperCollins, 1991.

VARGA, A. Omnet++. In: _____. *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.

VASANTHAAZHAGU, A. K.; GNANASEKAR, J. M. Cloud computing overview, security threats and solutions-a survey. In: *Proceedings of the International Conference on Informatics and Analytics*. New York, NY, USA: Association for Computing Machinery, 2016. (ICIA-16). ISBN 9781450347563. Disponível em: https://doi.org/10.1145/2980258.2982046.

VIKHAR, P. A. Evolutionary algorithms: A critical review and its future prospects. In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. [S.l.: s.n.], 2016. p. 261–265.

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.

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

YAN, P. *Fog Connectivity Clustering and MDP Modeling for Software-defined Vehicular Networks*. 89 p. Dissertação (Mestrado) — Dissertação (Mestrado) - Brock University, 2022.

YANG, W. et al. Secrecy-based resource allocation for vehicular communication networks with outdated csi. In: *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. [S.l.: s.n.], 2017. p. 1–5.

YANG, X.-S. Harmony search as a metaheuristic algorithm. In: _____. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 1–14. ISBN 978-3-642-00185-7. Disponível em: https://doi.org/10.1007/978-3-642-00185-7_1.

YANG, X.-S. A new metaheuristic bat-inspired algorithm. In: _____. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 65–74. ISBN 978-3-642-12538-6. Disponível em: https://doi.org/10.1007/978-3-642-12538-6_6.

YANG, X.-S. Bat algorithm: Literature review and applications. *International Journal of Bio-Inspired Computation*, v. 5, 08 2013.

YANG, X.-S.; GANDOMI, A. H. Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations*, Emerald, v. 29, n. 5, p. 464–483, jul. 2012.

ZHANG, T.; GRANDE, R. E. D.; BOUKERCHE, A. Vehicular cloud: Stochastic analysis of computing resources in a road segment. In: . New York, NY, USA: Association for Computing Machinery, 2015. (PE-WASUN '15), p. 9–16. ISBN 9781450337595. Disponível em: https://doi.org/10.1145/2810379.2810383.

ZHANG, X. et al. Joint communication and computation resource allocation in fog-based vehicular networks. *IEEE Internet of Things Journal*, v. 9, n. 15, p. 13195–13208, 2022.