UNIVERSIDADE ESTADUAL PAULISTA - UNESP

Instituto de Biociências, Letras e Ciências Exatas - campus de São José do Rio Preto

DOUGLAS DIAS LIEIRA

Um Framework Para Gerenciamento de Alocação de Tarefas em Computação de Borda Veicular com Prioridade por Colaboração e Sistema Billing

São José do Rio Preto

2026

Douglas Dias Lieira

Um Framework Para Gerenciamento de Alocação de Tarefas em Computação de Borda Veicular com Prioridade por Colaboração e Sistema Billing

Tese, apresentada à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas , São José do Rio Preto, para obtenção do título de Doutorem Ciência da Computação.

Área de Concentração: Computação aplicada

Orientador: Profº Dr. Rodolfo Ipolito Meneguette

São José do Rio Preto
2026

L719f

Lieira, Douglas Dias
Um Framework Para Gerenciamento de Alocação de Tarefas em Computação de Borda Veicular com Prioridade por Colaboração e Sistema Billing / Douglas Dias Lieira. -- São José do Rio Preto, 2026
176 f. : il., tabs.

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

1. Ciência da computação. 2. Computação de borda veicular. 3. Alocação de tarefas. 4. Colaboração. 5. Billing. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

IMPACTO POTENCIAL DESTA PESQUISA

A pesquisa contribui para o avanço da Computação de Borda Veicular ao propor um framework de alocação de tarefas que integra otimização, colaboração entre veículos e um mecanismo de *billing*. A abordagem apresenta potencial para melhorar a utilização de recursos computacionais distribuídos, a eficiência da execução de tarefas e a colaboração em ambientes veiculares dinâmicos. Seus resultados podem subsidiar novas pesquisas e aplicações em Computação de Borda, Sistemas de Transporte Inteligentes e Cidades Inteligentes, contribuindo para o desenvolvimento científico, tecnológico e educacional e para a investigação de modelos sustentáveis de compartilhamento de recursos computacionais.

POTENTIAL IMPACT OF THIS RESEARCH

This research contributes to the advancement of Vehicular Edge Computing by proposing a task allocation framework that integrates optimization, collaboration among vehicles, and a billing mechanism. The proposed approach has the potential to improve the utilization of distributed computational resources, task execution efficiency, and collaboration in dynamic vehicular environments. Its results may support further research and applications in Edge Computing, Internet of Vehicles, Intelligent Transportation Systems, and Smart Cities, contributing to scientific, technological, and educational development, as well as to the investigation of sustainable models for computational resource sharing.

DOUGLAS DIAS LIEIRA

UM FRAMEWORK PARA GERENCIAMENTO DE ALOCAÇÃO DE TAREFAS EM COMPUTAÇÃO DE BORDA VEICULAR COM PRIORIDADE POR COLABORAÇÃO E SISTEMA BILLING

Tese apresentado(a) à Universidade Estadual Paulista (UNESP), Instituto de Biociências, Letras e Ciências Exatas , São José do Rio Preto, para obtenção do título de Doutor em Ciência da Computação.

Área de Concentração: Computação aplicada

Data de defesa: 04/09/2026

BANCA EXAMINADORA

_____

Profº Dr. Rodolfo Ipolito Meneguette
UNESP – Instituto de Biociências, Letras e Ciências Exatas – Campus de São José do Rio Preto

_____

Profº Dr. José Rodrigues Torres Neto
Departamento de Computação - Universidade Federal do Piauí

_____

Profº Dr. André Luis Olivete
Instituto Federal de São Paulo - Câmpus Presidente Epitácio

_____

Profº Dr. Geraldo Francisco Donegá Zafalon
UNESP / Câmpus de São José do Rio Preto - Ibilce

_____

Profº Dr. Danilo Medeiros Eler
UNESP / Câmpus de Presidente Prudente - FCT

AGRADECIMENTOS

A Deus e a Nossa Senhora das Graças;
Aos meus pais por toda estrutura familiar para que eu pudesse me dedicar aos estudos;
À minha família, minha namorada e a todos que me incentivaram durante esta jornada;
Em especial, à minha amada filha Maria Eduarda Lieira, minha maior motivação para tudo;
Ao Matheus Quessada pelos auxílios no desenvolvimento dos trabalhos científicos;
Ao Instituto Federal de São Paulo por possibilitar o afastamento para licença qualificação.
Aos professores e servidores administrativos da UNESP pelo apoio e colaboração durante estes anos.
Ao professor Robson Eduardo De Grande pela supervisão durante o meu período de doutorado sanduíche na Brock University no Canadá;
E por fim, mas não menos importante, ao meu orientador Prof. Dr. Rodolfo Ipolito Meneguette pela sua orientação desde que iniciei meus trabalhos nas pesquisas, durante o mestrado e o doutorado. Muito obrigado por todos esses anos de orientação;
O presente trabalho foi parcialmente financiado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES).

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

(FAGLIONI, 2020)

RESUMO

A evolução das tecnologias associadas aos sistemas de transporte inteligentes tem impulsionado o desenvolvimento e a expansão do mercado de veículos conectados e autônomos em cidades inteligentes. Nesse contexto, a Computação de Borda Veicular (do inglês, *Vehicular Edge Computing* – VEC) surge como uma abordagem promissora para viabilizar serviços com requisitos de baixa latência e comunicação em tempo real entre veículos e infraestruturas rodoviárias. Entretanto, esse cenário introduz novos desafios para a comunidade científica. Entre os principais desafios destacam-se a elevada dinamicidade dos veículos e as constantes alterações na topologia da rede, fatores que afetam diretamente a disponibilidade de recursos computacionais oferecidos pelos serviços. Embora diversas técnicas de alocação de tarefas tenham sido propostas para ambientes de Computação de Borda Veicular, a maioria das abordagens concentra-se na maximização da taxa de atendimento ou no desempenho do processamento das tarefas, negligenciando a implementação de mecanismos de incentivo capazes de estimular o compartilhamento de recursos computacionais pelos veículos. Ademais, a incorporação de incentivos financeiros associados aos recursos compartilhados, fundamentados no nível de colaboração dos usuários, permanece amplamente inexplorada na literatura. Com o objetivo de superar essas limitações, neste trabalho apresenta-se um *framework* para gerenciamento da alocação de tarefas em ambientes de Computação de Borda Veicular com priorização baseada na colaboração e sistema de *billing*, denominado PRIOBILL. O PRIOBILL implementa um mecanismo de seleção de nós de borda fundamentado no Algoritmo de Otimização do Leão (do inglês, *Lion Optimization Algorithm* – LOA), visando priorizar o atendimento de tarefas de acordo com o histórico de colaboração de cada veículo. Adicionalmente, é implementado um modelo de cobrança baseado no uso *usage-based billing*, destinado a recompensar os veículos que disponibilizam seus recursos computacionais para a rede VEC. O *framework* desenvolvido foi avaliado por meio de extensos experimentos e comparado com técnicas de referência da literatura. Os resultados demonstram que o PRIOBILL é capaz de maximizar o processamento de tarefas, alcançando até 30,98% no número de tarefas processadas e minimizar a taxa de rejeição até 18,98% em relação ao segundo melhor método. Além disso, o método apresentou elevada eficiência na utilização dos recursos computacionais disponibilizados, alcançando vantagem de até 31,78% em relação ao métodos subsequentes, além de atingir altas taxas de transferência e um equilíbrio adequado entre a latência e a capacidade de atendimento das demandas. Por fim, o PRIOBILL também fornece estimativas de remuneração aos veículos colaboradores por meio do sistema de *billing* implementado.

Palavras-chave: alocação de tarefas; billing; computação de borda veicular; metaheurísticos bioinspirados; v2i.

ABSTRACT

The evolution of technologies associated with intelligent transportation systems has driven the development and expansion of the connected and autonomous vehicle market in smart cities. In this context, Vehicular Edge Computing (VEC) emerges as a promising approach to enable services requiring low latency and real-time communication between vehicles and road infrastructure. However, this scenario introduces new challenges for the scientific community. Key challenges include the high mobility of vehicles and constant changes in network topology—factors that directly affect the availability of computational resources offered by services. Although various task allocation techniques have been proposed for VEC environments, most approaches focus on maximizing service rates or task processing performance, neglecting the implementation of incentive mechanisms to encourage vehicles to share computational resources. Moreover, the incorporation of financial incentives for shared resources—based on user collaboration levels—remains largely unexplored in the literature. To address these limitations, this work presents a framework for managing task allocation in VEC environments, featuring collaboration-based prioritization and a billing system, named PRIOBILL. PRIOBILL implements an edge node selection mechanism based on the Lion Optimization Algorithm (LOA), aiming to prioritize task servicing according to each vehicle's collaboration history. Additionally, a usage-based billing model is implemented to reward vehicles that make their computational resources available to the VEC network. The developed framework was evaluated through extensive experiments and compared with benchmark techniques from the literature. The results demonstrate that the PRIOBILL is capable of maximizing task processing—achieving up to a 30.98% increase in the number of processed tasks—and minimizing the rejection rate by up to 18.98% compared to the second-best method. Furthermore, the method exhibited high efficiency in utilizing available computational resources, outperforming subsequent methods by up to 31.78%, while also achieving high throughput and an appropriate balance between latency and demand-handling capacity. Finally, the framework also provides remuneration estimates for participating vehicles through the implemented billing system.

Keywords: task allocation; billing; vehicular edge computing; bioinspired metaheuristics; v2i.

LISTA DE FIGURAS

Figura 1 – Representação da comunicação entre serviços ITS em cidades inteligentes. 25
Figura 2 – Comunicação em VANETs. 26
Figura 3 – Comunicação Veículo-para-Infraestrutura. 28
Figura 4 – Exemplo de Computação de Borda Móvel. 30
Figura 5 – Exemplo de VCC. 31
Figura 6 – Exemplo de Computação em Névoa Veicular. 32
Figura 7 – Exemplo de VEC. 33
Figura 8 – Exemplo de seleção do método Guloso. 34
Figura 9 – Exemplo de seleção do método Randômico. 35
Figura 10 – Exemplo de movimentação do algoritmo PSO. 36
Figura 11 – Pirâmide de hierarquia no GWO. 37
Figura 12 – Exemplo de movimentação da baleia. 38
Figura 13 – Visão geral do modelo de sistema V2I em VEC. 50
Figura 14 – Exemplo de cenário de transmissão de RSU para nós de borda. 55
Figura 15 – Leão caça. 66
Figura 16 – Princípio da Aprendizagem Baseada em Oposição. 66
Figura 17 – Escape. 67
Figura 18 – Movimento caça. 68
Figura 19 – Vagar. 70
Figura 20 – Nomade 1. 71
Figura 21 – Nomade 2. 72
Figura 22 – Defesa. 72
Figura 23 – Representação dos processos de migração e equilíbrio populacional. . . 73
Figura 24 – Movimento dos nós de borda do PRIOBILL para atacar a tarefa alvo. . 74
Figura 25 – Etapa de migração do nó de borda 75
Figura 26 – Midtown Manhattan com representação gráfica de RSUs. 78
Figura 27 – Trecho da Midtown Manhattan utilizado para simulação. 79
Figura 28 – Posicionamento das RSUs no trecho de Midtown Manhattan no SUMO. 81
Figura 29 – Número de tarefas menores processadas no horário da madrugada. . . . 86
Figura 30 – Número de tarefas menores processadas no horário da manhã. 87
Figura 31 – Número de tarefas menores processadas no horário de pico. 88
Figura 32 – Número de tarefas intermediárias processadas no horário da madrugada. 89
Figura 33 – Número de tarefas intermediárias processadas no horário da manhã. . . 90
Figura 34 – Número de tarefas intermediárias processadas no horário de pico. . . . 91
Figura 35 – Número de tarefas pesadas processadas no horário da madrugada. . . . 92

Figura 36 – Número de tarefas pesadas processadas no horário da manhã. 94
Figura 37 – Número de tarefas pesadas processadas no horário de pico. 95
Figura 38 – Número de tarefas com pesos aleatórios processadas no horário da madrugada. 96
Figura 39 – Número de tarefas com pesos aleatórios processadas no horário da manhã. 97
Figura 40 – Número de tarefas com pesos aleatórios processadas no horário de pico. 98
Figura 41 – Número de tarefas menores não processadas no horário da madrugada. 100
Figura 42 – Número de tarefas menores não processadas no horário da manhã. . . . 101
Figura 43 – Número de tarefas menores não processadas no horário de pico. 102
Figura 44 – Número de tarefas intermediárias não processadas no horário da madrugada. 103
Figura 45 – Número de tarefas intermediárias não processadas no horário da manhã.104
Figura 46 – Número de tarefas intermediárias não processadas no horário de pico. . 105
Figura 47 – Número de tarefas maiores não processadas no horário da madrugada. 106
Figura 48 – Número de tarefas maiores não processadas no horário da manhã. . . . 107
Figura 49 – Número de tarefas maiores não processadas no horário de pico. 108
Figura 50 – Número de tarefas com pesos aleatórios não processadas no horário da madrugada. 109
Figura 51 – Número de tarefas com pesos aleatórios não processadas no horário da manhã. 110
Figura 52 – Número de tarefas com pesos aleatórios não processadas no horário de pico. 110
Figura 53 – Quantidade de recursos utilizados com tarefas menores no horário da madrugada. 112
Figura 54 – Quantidade de recursos utilizados com tarefas menores no horário da manhã. 113
Figura 55 – Quantidade de recursos utilizados com tarefas menores no horário de pico. 114
Figura 56 – Quantidade de recursos utilizados com tarefas intermediárias no horário da madrugada. 115
Figura 57 – Quantidade de recursos utilizados com tarefas intermediárias no horário da manhã. 116
Figura 58 – Quantidade de recursos utilizados com tarefas intermediárias no horário de pico. 117
Figura 59 – Quantidade de recursos utilizados com tarefas maiores no horário da madrugada. 118
Figura 60 – Quantidade de recursos utilizados com tarefas maiores no horário da manhã. 119

Figura 61 – Quantidade de recursos utilizados com tarefas maiores no horário de pico. 121
Figura 62 – Quantidade de recursos utilizados com tarefas com pesos aleatórios no horário da madrugada. 121
Figura 63 – Quantidade de recursos utilizados com tarefas com pesos aleatórios no horário da manhã. 123
Figura 64 – Quantidade de recursos utilizados com tarefas com pesos aleatórios no horário de pico. 124
Figura 65 – Latência média para processamento de tarefas menores no horário da madrugada. 126
Figura 66 – Latência média para processamento de tarefas menores no horário da manhã. 127
Figura 67 – Latência média para processamento de tarefas menores no horário de pico. 128
Figura 68 – Latência média para processamento de tarefas intermediárias no horário da madrugada. 129
Figura 69 – Latência média para processamento de tarefas intermediárias no horário da manhã. 131
Figura 70 – Latência média para processamento de tarefas intermediárias no horário de pico. 132
Figura 71 – Latência média para processamento de tarefas maiores no horário da madrugada. 133
Figura 72 – Latência média para processamento de tarefas maiores no horário da manhã. 134
Figura 73 – Latência média para processamento de tarefas maiores no horário de pico. 136
Figura 74 – Latência média para processamento de tarefas com pesos aleatórios no horário da madrugada. 137
Figura 75 – Latência média para processamento de tarefas com pesos aleatórios no horário da manhã. 138
Figura 76 – Latência média para processamento de tarefas com pesos aleatórios no horário de pico. 140
Figura 77 – Throughput para tarefas menores no horário da madrugada. 142
Figura 78 – Throughput para tarefas menores no horário da manhã. 143
Figura 79 – Throughput para tarefas menores no horário de pico. 144
Figura 80 – Throughput para tarefas intermediárias no horário da madrugada. . . . 145
Figura 81 – Throughput para tarefas intermediárias no horário da manhã. 146
Figura 82 – Throughput para tarefas intermediárias no horário de pico. 147
Figura 83 – Throughput para tarefas maiores no horário da madrugada. 148

Figura 84 – Throughput para tarefas maiores no horário da manhã. 150
Figura 85 – Throughput para tarefas maiores no horário de pico. 151
Figura 86 – Throughput para tarefas com pesos aleatórios no horário da madrugada. 152
Figura 87 – Throughput para tarefas com pesos aleatórios no horário da manhã. . . 153
Figura 88 – Throughput para tarefas com pesos aleatórios no horário de pico. . . . 154
Figura 89 – Quantidade de recursos utilizados na simulação com tarefas menores. . 156
Figura 90 – Quantidade de recursos utilizados na simulação com tarefas intermediárias. 156
Figura 91 – Quantidade de recursos utilizados na simulação com tarefas maiores. . 157
Figura 92 – Quantidade de recursos utilizados na simulação com tarefas com pesos aleatórios. 158

LISTA DE TABELAS

Tabela 1 – Abstração do estado da arte. 47
Tabela 2 – Sumário de Notações 49
Tabela 3 – Parâmetros de mobilidades 79
Tabela 4 – Parâmetros gerais da simulação 82
Tabela 5 – Conversão de valores: Horas para segundos para Reais 83
Tabela 6 – Recursos utilizados e faturamento estimado pelo sistema billing do PRIOBILL. 158

LISTA DE ABREVIATURAS E SIGLAS

AHP *Analytic Hierarchy Process*

AWS *Amazon Web Services*

CC *Cloud Computing*

EC *Edge Computing*

GWO *Grey Wolf Optimizer*

HHO *Harris Hawks Optimization*

IA Inteligência Artificial

I2I *Infrastructure-to-Infrastructure*

IaaS *Infrastructure as a Service*

IEEE *Institute of Electrical and Electronics Engineers*

IoT *Internet of Things*

ITS *Intelligent Transportation Systems*

LOA *Lion Optimization Algorithm*

LTE *Long-Term Evolution*

MAC *Media Access Control*

MEC *Multi-access Edge Computing*

MV Máquina Virtual

NFV *Network Functions Virtualization*

NOMA *Non-Orthogonal Multiple Access*

OBU *On-Board Unit*

OMNeT++ *Objective Modular Network Testbed in C++*

PSO *Particle Swarm Optimization*

QoE *Quality of Experience*

QoS *Quality of Service*

RAN *Radio Access Network*

RSU *Roadside Unit*

SMDP *Semi-Markov Decision Process*

SUMO *Simulation of Urban MObility*

TraCI *Traffic Control Interface*

UAV *Unmanned Aerial Vehicle*

V2B *Vehicle-to-Barrier*

V2C *Vehicle-to-Cloud*

V2I *Vehicle-to-Infrastructure*

V2P *Vehicle-to-Pedestrian*

V2U *Vehicle-to-Unmanned Aerial Vehicle*

V2V *Vehicle-to-Vehicle*

VANETs *Vehicular Ad Hoc Networks*

VCC *Vehicular Cloud Computing*

VEC *Vehicular Edge Computing*

VEINS *Vehicles in Network Simulation*

VFC *Vehicular Fog Computing*

WAVE *Wireless Access in Vehicular Environments*

WOA *Whale Optimization Algorithm*

SUMÁRIO

1 INTRODUÇÃO 18
1.1 MOTIVAÇÃO **20**
1.2 QUESTÃO DE PESQUISA E HIPÓTESE **21**
1.3 OBJETIVOS **21**
1.4 ORGANIZAÇÃO DO TRABALHO **22**

2 FUNDAMENTAÇÃO TEÓRICA 24
2.1 SISTEMAS DE TRANSPORTES INTELIGENTES **24**
2.2 PARADIGMAS COMPUTACIONAIS PARA REDES VEICULARES . **28**
2.2.1 Computação em Nuvem 28
2.2.2 Computação em Névoa 29
2.2.3 Computação de Borda 29
2.2.4 Computação de Borda Móvel 29
2.2.5 Computação em Nuvem Veicular 30
2.2.6 Computação em Névoa Veicular 31
2.2.7 Computação de Borda Veicular 31
2.3 ALOCAÇÃO E GERENCIAMENTO DE TAREFAS EM VEC **32**
2.3.1 Compartilhamento de Recursos e Cooperação Veicular 38
2.3.2 Billing 39
2.4 CONSIDERAÇÕES PARCIAIS **40**

3 ESTADO DA ARTE 41
3.1 TRABALHOS RELACIONADOS **41**
3.2 CONSIDERAÇÕES PARCIAIS **45**

4 FRAMEWORK PARA GERENCIAMENTO DE ALOCAÇÃO DE TAREFAS EM COMPUTAÇÃO DE BORDA VEICULAR COM PRIORIDADE POR COLABORAÇÃO E SISTEMA BILLING 48
4.1 MODELO DE SISTEMA **50**
4.2 FORMULAÇÃO DO PROBLEMA **51**
4.3 CLUSTERIZAÇÃO **54**
4.4 INFRAESTRUTURA VEÍCULO-PARA-INFRAESTRUTURA (V2I) . **54**
4.5 GERENCIAMENTO DE TAREFAS **57**
4.6 GERENCIAMENTO NO NÓ DE BORDA **61**
4.7 *BILLING* **63**

4.8 MECANISMO DE ALOCAÇÃO DE TAREFAS PRIOBILL: BASEADO EM LOA **64**
4.9 AVALIAÇÃO E ANÁLISES **76**
4.9.1 Cenário e parâmetros da simulação 77
4.9.2 Métricas de Avaliação de Desempenho 80
4.9.3 Algoritmos avaliados 83
4.10 RESULTADOS **84**
4.10.1 Tarefas processadas 85
4.10.2 Tarefas não processadas 99
4.10.3 Quantidade de recursos utilizados 111
4.10.4 Latência média 125
4.10.5 Throughput 141
4.10.6 Billing 155
4.10.7 Considerações parciais 158

5 CONCLUSÃO 160
5.1 PRINCIPAIS CONTRIBUIÇÕES **161**
5.2 PRODUÇÃO CIENTÍFICA **161**
5.2.1 Publicação em periódico como autor principal 162
5.2.2 Publicações em congressos como autor principal 162
5.2.3 Colaborações em publicações em periódicos 162
5.2.4 Colaborações em publicações em congressos 163
5.3 TRABALHOS FUTUROS **163**
5.4 UTILIZAÇÃO DE INTELIGÊNCIA ARTIFICIAL **163**

REFERÊNCIAS 165

DADOS CURRICULARES 172

1 INTRODUÇÃO

O mercado de comunicação em redes veiculares tem apresentado crescimento constante nos últimos anos. Projeta-se que, até 2032, esse mercado alcance um valor de US$ 216,50 bilhões, impulsionado pela crescente demanda por maior segurança viária e por sistemas mais eficientes de gerenciamento do tráfego. Além disso, a pandemia da COVID-19 acelerou a adoção de tecnologias digitais e evidenciou a necessidade de uma infraestrutura de transporte mais resiliente e adaptável. Nesse contexto, as principais montadoras passaram a integrar Sistemas Inteligentes de Transporte (ITS, do inglês, *Intelligent Transportation System*) em suas linhas de veículos. Essa evolução tecnológica proporcionou maior velocidade na transmissão de dados e redução da latência, possibilitando uma comunicação mais eficiente e em tempo real entre os veículos (INSIGHTS, 2024).

As redes veiculares permitem que os elementos da rede se comuniquem e compartilhem informações, como notificações de trânsito, alertas de acidentes, bloqueios nas vias, congestionamentos, sugestões de rotas, mensagens, vídeos e outros tipos de arquivos. Entretanto, realizar a transmissão de mensagens e a troca de dados de forma confiável, eficiente e com a velocidade exigida pelas aplicações constitui um dos principais desafios dos ITS (CUI et al., 2022). Para viabilizar essa comunicação, podem ser utilizadas infraestruturas de apoio, como as Unidades de Beira de Estrada (RSU, do inglês, *RoadSide Unit*), responsáveis pelo gerenciamento da comunicação e da troca de dados com os veículos da rede. Esse modelo de comunicação é conhecido como veículo-para-infraestrutura (V2I, do inglês, *Vehicle-to-Infrastructure*), no qual a interação entre os veículos e as RSUs ocorre por meio da troca de mensagens (SANDESH; SANTHOSH, 2025).

No contexto das cidades inteligentes, torna-se necessário que os serviços de rede sejam capazes de atender às exigências das redes veiculares, caracterizadas por um ambiente distribuído, altamente dinâmico e com requisitos de processamento em tempo real, conectividade ubíqua e latência ultrabaixa. Nesse cenário, surgiu o paradigma da Computação de Borda Veicular (VEC, do inglês, *Vehicular Edge Computing*), com o objetivo de fornecer comunicação em tempo real e serviços com reconhecimento de localização (NING et al., 2019). A VEC é composta por um conjunto de veículos equipados com computadores de bordo que, em conjunto com as RSUs, comunicam-se e compartilham aplicações e serviços com os usuários dentro de uma determinada área de cobertura. Os veículos e as RSUs constituem os nós da borda veicular, cuja comunicação e troca de dados devem ocorrer de forma segura, eficiente e com latência ultrabaixa (MENEGUETTE et al., 2021).

Pesquisadores têm buscado mecanismos capazes de garantir elevados níveis de Qualidade de Serviço (QoS, do inglês, *Quality of Service*) e Qualidade de Experiência do Usuário (QoE, do inglês, *Quality of Experience*) para enfrentar os desafios inerentes à Computação de Borda Veicular. Entre esses desafios destacam-se a complexidade da arquitetura das redes, a mobilidade dos veículos, o gerenciamento de recursos, a latência, a escalabilidade e a segurança (MENEGUETTE et al., 2021). Dentre esses fatores, o gerenciamento de recursos exerce impacto direto sobre a QoS e a QoE oferecidas aos usuários. A escassez de recursos, quando não tratada de forma eficiente e prioritária, aliada à elevada mobilidade dos veículos e ao atraso na transmissão das mensagens, compromete o desempenho dos serviços disponibilizados pela rede. Dessa forma, o desenvolvimento de mecanismos capazes de utilizar, de maneira eficiente, os recursos computacionais disponíveis tem se tornado um importante tema de pesquisa (EHSAN et al., 2026; QUESSADA et al., 2026; CHEN et al., 2025; LIEIRA et al., 2024).

Além disso, a incorporação de critérios que incentivem os usuários a disponibilizar seus recursos computacionais pode representar uma alternativa para reduzir os efeitos da escassez de recursos. Nesse sentido, o desenvolvimento de algoritmos que priorizem determinadas tarefas durante o processo de seleção pode estimular os usuários a compartilharem seus recursos com a rede (LIEIRA et al., 2025). Outro mecanismo de incentivo consiste na remuneração dos usuários pelo compartilhamento de seus recursos computacionais, de forma semelhante ao modelo adotado por grandes provedores de serviços em nuvem, como Serviços Web da Amazon (AWS, do inglês, *Amazon Web Service*), Azure e Google *Cloud*, que utilizam o modelo de Infraestrutura como Serviço (IaaS, do inglês, *Infrastructure as a Service*). Essa abordagem pode ser implementada por meio de um sistema de *billing*, responsável pelo gerenciamento da cobrança e da remuneração pelo uso dos recursos computacionais (STRIPE, 2025).

Atualmente, a literatura apresenta diversas técnicas voltadas à alocação eficiente de recursos computacionais. Essas técnicas podem ser classificadas em: *i*) métodos de otimização, fundamentados em modelos matemáticos (COSTA et al., 2020b); *ii*) mecanismos adaptativos, capazes de ajustar seu comportamento de acordo com as informações recebidas (ALFARRAJ et al., 2019); e *iii*) algoritmos meta-heurísticos, inspirados em fenômenos naturais, no comportamento de animais ou em processos observados na natureza (MIRJALILI; MIRJALILI; LEWIS, 2014; YAZDANI; JOLAI, 2016). Nos últimos anos, algoritmos meta-heurísticos têm sido amplamente empregados na resolução de problemas de alocação de tarefas e recursos computacionais, conforme demonstrado em (LIEIRA et al., 2025; LIEIRA et al., 2024; GOYAL et al., 2021; LIEIRA et al., 2022b; LIEIRA et al., 2022a; KUMAR et al., 2022; QUESSADA et al., 2022b; SING et al., 2022; AZIZI et al., 2022). Essa crescente adoção deve-se à sua capacidade de produzir soluções eficazes em cenários dinâmicos, complexos e altamente mutáveis, como aqueles encontrados nas redes veiculares.

No entanto, esses trabalhos concentram-se em cenários específicos de mobilidade, sem considerar ambientes mais complexos, como aqueles baseados em comunicação V2I e em cenários de simulação mais próximos das condições reais. Nesses ambientes, é possível formar agrupamentos mais amplos de veículos que compartilham seus recursos computacionais dentro de uma mesma área de cobertura, constituindo um *pool* de recursos. Como consequência, a elevada dinamicidade da rede torna o gerenciamento desses recursos significativamente mais complexo, uma vez que os veículos entram e saem da VEC continuamente durante sua mobilidade, disponibilizando ou retirando seus recursos computacionais da rede. Além disso, esses veículos podem tanto requisitar quanto compartilhar aplicações e serviços no contexto dos Sistemas Inteligentes de Transporte.

Nesse contexto, esta tese apresenta um framework dinâmico para alocação de tarefas em Computação de Borda Veicular, com aplicação de mecanismos de **PRIO**rização e recompensa aos usuários com **BILL**ing, baseado no Algoritmo de Otimização do Leão, denominado PRIOBILL. O *framework* desenvolvido segue os princípios do meta-heurístico LOA (do inglês, *Lion Optimization Algorithm*) e emprega um mecanismo de priorização durante o processo de seleção do nó de borda mais adequado para o atendimento das tarefas. Essa priorização é realizada a partir do histórico de colaboração dos veículos, considerando tanto a quantidade de vezes que cada veículo disponibiliza seus recursos computacionais para a rede quanto o número de tarefas que já executou. Além disso, é considerado um cenário dinâmico, no qual os nós de borda ingressam e deixam a VEC continuamente ao longo da simulação. O *framework* recebe as tarefas submetidas pelos usuários, realiza sua ordenação de acordo com os critérios de prioridade estabelecidos e otimiza sua distribuição entre os membros da VEC. Adicionalmente, o mecanismo contabiliza os recursos computacionais compartilhados por cada veículo, permitindo a implementação de um sistema de *billing* destinado à remuneração dos usuários colaboradores.

1.1 MOTIVAÇÃO

A elevada quantidade de veículos conectados simultaneamente em cidades inteligentes demanda serviços capazes de atender às aplicações de forma eficiente, considerando a limitação dos recursos computacionais disponíveis e as constantes mudanças na topologia das redes veiculares. Nesse contexto, realizar a alocação eficiente de tarefas computacionais constitui um dos principais desafios apontados pela literatura.

Diversos trabalhos abordam o problema da alocação de tarefas em ambientes de Computação de Borda Veicular, como os propostos por Azizi et al. (2022), Kumar et al. (2022), Hosseini, Nickray e Ghanbari (2022), Pereira et al. (2020), Lieira et al. (2022a), Ribeiro et al. (2023), Lieira et al. (2025) e Lieira et al. (2023b). Entretanto, parte dessas abordagens não considera cenários de simulação que reproduzam as características de

ambientes reais, incluindo a elevada dinamicidade da rede, as constantes alterações de topologia, os atrasos na comunicação e as falhas de transmissão. Além disso, muitas dessas propostas não priorizam a maximização do processamento de tarefas e a utilização eficiente dos recursos computacionais, tampouco incorporam mecanismos de priorização para usuários que colaboram com a rede ou sistemas de *billing* destinados à remuneração dos veículos colaboradores.

Diante desse contexto, este trabalho busca enfrentar o problema da limitação de recursos computacionais em ambientes de Computação de Borda Veicular por meio do desenvolvimento do *framework* PRIOBILL. O mecanismo de seleção é baseado no algoritmo meta-heurístico bioinspirado Algoritmo de Otimização do Leão e empregado para realizar a tomada de decisão durante o processo de alocação de tarefas computacionais. O *framework* foi projetado para adaptar-se às constantes mudanças do ambiente veicular, visando otimizar a utilização dos recursos computacionais disponíveis e maximizar o atendimento às demandas dos usuários.

1.2 QUESTÃO DE PESQUISA E HIPÓTESE

Nesta seção, são apresentadas a questão de pesquisa e hipótese que direcionam este trabalho.

Questão de Pesquisa

O emprego de um sistema eficiente de tomada de decisão para computação de borda veicular, que disponibilize seus serviços de acordo com as necessidades específicas de cada usuário e recompensa os usuários com base na cooperação com a rede, é viável em termos de utilização eficiente dos escassos recursos de rede, tempo de inferência e consumo computacional?

Hipótese

O desenvolvimento de um *framework* para o gerenciamento da alocação de tarefas em serviços de computação de borda veicular, implementando perfis de usuários durante o processo de tomada de decisão para alocação de tarefas e um sistema de *billing* baseado no compartilhamento de recursos, pode permitir a utilização eficiente dos recursos, otimizar a latência e melhorar a experiência do usuário.

1.3 OBJETIVOS

Este trabalho tem como objetivo geral o desenvolvimento e a avaliação de um *framework* para alocação de tarefas em ambientes de Computação de Borda Veicular, com priorização baseada na colaboração dos usuários e integrado a um sistema de *billing*.

O *framework* visa otimizar o processo de alocação de tarefas, maximizar a utilização dos recursos computacionais disponibilizados, reduzir o tempo de inferência e minimizar o consumo de recursos computacionais. Além disso, o *framework* incorpora um sistema de *billing* como mecanismo de incentivo para estimular os usuários a compartilharem seus recursos computacionais com a rede.

Os objetivos específicos deste trabalho são:

- Implementar um cenário de Computação de Borda Veicular em um ambiente de simulação urbana;
- Desenvolver e avaliar um mecanismo de alocação de tarefas em VEC com priorização baseada no histórico de colaboração dos usuários;
- Implementar e avaliar um mecanismo de seleção do nó de borda mais adequado para a alocação de tarefas utilizando o algoritmo meta-heurístico LOA;
- Incorporar um sistema de *billing*, baseado no modelo de Infraestrutura como Serviço (IaaS), como forma de incentivar os usuários a compartilharem seus recursos computacionais;
- Realizar experimentos por meio da linguagem de programação Python[1] e dos simuladores SUMO[2], OMNeT++[3] e Veins[4], obtendo parâmetros que permitam avaliar e comparar as métricas de desempenho do ambiente de Computação de Borda Veicular para validação do *framework*;
- Comparar o mecanismo desenvolvido com métodos apresentados na literatura, visando validar sua eficiência e eficácia;
- Maximizar a utilização dos recursos computacionais disponíveis, reduzindo sua subutilização e minimizando os custos e o consumo computacional da VEC;
- Buscar um equilíbrio entre as métricas de atendimentos, latência e *throughput*.

1.4 ORGANIZAÇÃO DO TRABALHO

A tese está organizada da seguinte forma:

No **Capítulo 2**, é apresentada a fundamentação teórica, abrangendo os conceitos de redes veiculares, Computação de Borda, Computação de Borda Veicular, técnicas de gerenciamento de recursos, mecanismos de priorização de usuários e sistemas *billing*.

[1] https://www.python.org/
[2] https://www.eclipse.org/sumo/
[3] https://omnetpp.org/
[4] https://veins.car2x.org/

No **Capítulo 3**, é apresentado o estado da arte, contendo uma revisão dos trabalhos mais recentes relacionados à alocação de tarefas computacionais, bem como uma análise comparativa entre essas abordagens e a solução desenvolvida nesta tese.

No **Capítulo 4**, é apresentado o *framework* para gerenciamento da alocação de tarefas em Computação de Borda Veicular com priorização baseada na colaboração dos usuários e integrado a um sistema *billing*, denominado PRIOBILL. Também são descritos o cenário de simulação desenvolvido para a validação do framework, a metodologia de avaliação empregada e a análise dos resultados obtidos.

No **Capítulo 5**, são apresentadas as conclusões da tese, destacando as principais contribuições do trabalho, as produções científicas decorrentes da pesquisa e as perspectivas para o desenvolvimento de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os principais conceitos que fundamentam o desenvolvimento deste trabalho. Inicialmente, são abordados os Sistemas de Transportes Inteligentes (ITS) e os conceitos relacionados utilizados ao longo da tese. Em seguida, são descritos os principais paradigmas computacionais aplicados às redes veiculares e suas variações. Por fim, são apresentados os conceitos relacionados ao processo de alocação e gerenciamento de tarefas em Computação de Borda Veicular, incluindo mecanismos de alocação de recursos e *billing*.

2.1 SISTEMAS DE TRANSPORTES INTELIGENTES

Os Sistemas de Transportes Inteligentes (ITS) consistem na aplicação de tecnologias aos sistemas de transporte, promovendo a integração entre veículos e as infraestruturas que os cercam com o objetivo de aumentar a segurança viária, reduzir congestionamentos, minimizar a ocorrência de acidentes, disponibilizar serviços aos usuários e otimizar rotas (ALAM; FERREIRA; FONSECA, 2016). Diante do crescente número de veículos em circulação nos centros urbanos, torna-se essencial que serviços relacionados à segurança, ao atendimento médico emergencial, à entrega de mantimentos e ao transporte de equipamentos médicos sejam realizados de forma rápida, eficiente e segura. Para atender a essas demandas, pesquisadores têm desenvolvido e aplicado diversas tecnologias com o objetivo de aprimorar a qualidade dos serviços de transporte e melhorar a experiência dos usuários (GARG; KAUR, 2023).

As cidades inteligentes representam uma nova geração de centros urbanos, caracterizados pela integração de tecnologias e serviços capazes de utilizar informações provenientes de diferentes fontes para apoiar a tomada de decisões e promover melhorias na qualidade de vida da população. Impulsionadas pela ampla utilização de aplicações web e redes de comunicação, essas cidades incorporam tecnologias voltadas ao gerenciamento eficiente das informações geradas, transformando grandes volumes de dados em conhecimento útil para seus usuários. Para viabilizar essa infraestrutura, são empregados recursos como redes de banda larga com e sem fio, redes metropolitanas, cabos de fibra óptica e Unidades de Beira de Estrada (RSU), que proporcionam conectividade entre os diversos serviços disponíveis (BITTENCOURT et al., 2025; KOMNINOS, 2014). Nesse contexto, os ITS constituem um dos principais pilares para a implementação de Cidades Inteligentes. Na Figura 1 apresenta-se uma representação da comunicação entre os serviços de ITS em um ambiente urbano, no qual veículos, RSUs e semáforos encontram-se interconectados.

Figura 1 – Representação da comunicação entre serviços ITS em cidades inteligentes.



Fonte: Gerada por IA.

Com a evolução das tecnologias embarcadas em veículos inteligentes, espera-se que, no futuro, esses veículos estejam altamente conectados entre si e aos diversos dispositivos distribuídos pelas cidades inteligentes. As redes veiculares, também conhecidas como VANETs (do inglês, *Vehicular Ad-Hoc Networks*), possibilitam o compartilhamento de dados e informações relevantes entre os veículos, oferecendo serviços relacionados à segurança no trânsito, monitoramento de congestionamentos, redução da poluição, aplicações de infoentretenimento, entre outros (NOOR-A-RAHIM et al., 2022; HUSSEIN et al., 2022; DAI et al., 2022). Nesse contexto, espera-se que as redes veiculares sejam capazes de fornecer serviços com elevados níveis de qualidade aos seus usuários. Para isso, é necessário que os recursos computacionais e de comunicação da rede sejam alocados e utilizados de forma eficiente.

Entretanto, diversos desafios ainda são objeto de investigação pela comunidade científica com o objetivo de aprimorar a qualidade dos serviços oferecidos pelas redes veiculares. Entre esses desafios destacam-se a latência, a largura de banda, o congestionamento dos canais de comunicação sem fio, a fragmentação e a limitação do espectro disponível, o gerenciamento de dados, a heterogeneidade das informações, a segurança e a privacidade dos dados, as limitações de *hardware*, a elevada mobilidade dos veículos, os custos associados à infraestrutura de comunicação e os efeitos provocados pelo ambiente de operação (NOOR-A-RAHIM et al., 2022; HUSSEIN et al., 2022; MENEGUETTE et

al., 2021). Grande parte desses desafios está relacionada ao processo de comunicação entre os elementos da rede, o qual pode ocorrer de diferentes formas, conforme ilustrado na Figura 2 e descrito a seguir:

Figura 2 – Comunicação em VANETs.



Fonte: Gerado por IA.

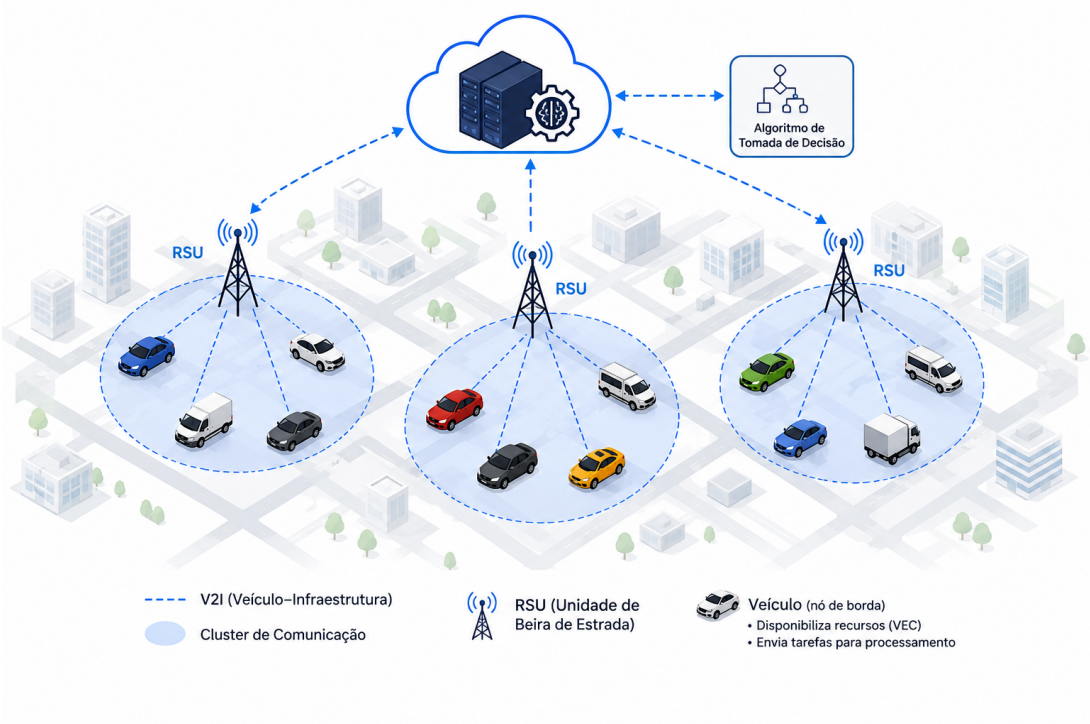
- V2V: comunicação veículo para veículo, na qual a troca de informações ocorre diretamente entre os dispositivos instalados nos veículos, sem a necessidade de infraestrutura de apoio;
- V2I: comunicação veículo para infraestrutura, em que os veículos compartilham dados com elementos da infraestrutura, como RSUs e estações rádio-base de redes celulares;
- I2I: comunicação infraestrutura para infraestrutura, na qual informações, geralmente relacionadas ao tráfego, são trocadas exclusivamente entre elementos da infraestrutura;
- V2P: comunicação veículo para pedestre, em que os dispositivos embarcados nos veículos comunicam-se com dispositivos móveis utilizados pelos pedestres;

- V2B: comunicação veículo para barreira rodoviária, na qual sensores embarcados nos veículos interagem com barreiras de proteção, contribuindo para a prevenção de acidentes;
- V2C: comunicação veículo para nuvem, em que serviços de computação em nuvem são utilizados para armazenar, compartilhar e processar os dados gerados pelos veículos;
- V2U: comunicação veículo para Veículos Aéreos Não Tripulados (UAV, do inglês, *Unmanned Aerial Vehicles*), caracterizada pela comunicação sem fio entre veículos e UAVs (HUSSEIN et al., 2022).

Neste trabalho, é implementado um cenário de comunicação V2I. Com os avanços tecnológicos, as montadoras passaram a equipar os veículos com uma Unidade de Bordo (OBU, do inglês, *OnBoard Unit*), responsável por permitir a comunicação sem fio entre os veículos e outros elementos da rede. Por meio da OBU, um veículo pode transmitir mensagens tanto para outros veículos (V2V) quanto para infraestruturas rodoviárias (V2I) (ALI; LI, 2020). Na Figura 3 ilustra-se um exemplo de comunicação V2I, no qual um veículo envia uma mensagem para uma RSU (mensagem 1, em azul). Essa mensagem pode conter informações relacionadas à rota, às condições de tráfego, à segurança viária, a aplicações de infoentretenimento, entre outras, além de representar uma solicitação de alocação de tarefas computacionais. Após receber a mensagem, a RSU a retransmite aos veículos interconectados localizados dentro de sua área de cobertura (mensagem 2, em verde).

A comunicação entre veículos em movimento e RSUs pode ser realizada por meio de mensagens *beacon*, utilizando o padrão IEEE 802.11p, também conhecido como Acesso Sem Fio em Ambientes Veiculares (WAVE, do inglês, *Wireless Access in Vehicular Environments*). O padrão WAVE opera com múltiplos canais de comunicação, permitindo o tratamento das mensagens de acordo com diferentes níveis de prioridade, por meio de distintas classes de acesso. Essa característica possibilita que mensagens críticas, como aquelas relacionadas à segurança viária, sejam priorizadas e transmitidas de forma confiável, mesmo em cenários complexos de redes veiculares (EICHLER, 2007). O padrão IEEE 802.11p concentra-se nas camadas física e de controle de acesso ao meio (MAC, do inglês, *Media Access Control*) (JIANG; DELGROSSI, 2008), sendo amplamente empregado em aplicações que utilizam comunicação V2I (SHA-MOHAMMAD; ABDEL-WAHAB; POPESCU, 2013).

Figura 3 – Comunicação Veículo-para-Infraestrutura.



Fonte: Gerada por IA.

2.2 PARADIGMAS COMPUTACIONAIS PARA REDES VEICULARES

Nesta seção, são apresentados os principais paradigmas computacionais empregados em redes veiculares, destacando seus conceitos e características de forma introdutória.

2.2.1 Computação em Nuvem

A Computação em Nuvem surgiu como um paradigma capaz de fornecer, sob demanda, recursos de tecnologia da informação de forma escalável, flexível e gerenciável, incluindo servidores, armazenamento, plataformas e aplicações para indivíduos e organizações. A infraestrutura disponibilizada por esse modelo impulsionou o desenvolvimento de importantes tendências tecnológicas, como computação móvel, Internet das Coisas, *Big Data* e Inteligência Artificial (SUNYAEV, 2024).

A Computação em Nuvem oferece elevada flexibilidade para armazenamento de dados e disponibilidade de recursos computacionais. Os provedores de serviços em nuvem são responsáveis pelo gerenciamento da infraestrutura física e disponibilizam interfaces que facilitam o acesso e a utilização dos serviços pelos usuários. Uma de suas principais características é a escalabilidade, que permite ampliar ou reduzir dinamicamente a capacidade de processamento e armazenamento conforme a demanda da aplicação. Essa flexibilidade possibilita o processamento de grandes volumes de dados e a execução de

aplicações computacionalmente intensivas de maneira eficiente (ANDRIULO et al., 2024).

2.2.2 Computação em Névoa

A Computação em Névoa surgiu como uma extensão da Computação em Nuvem, estabelecendo uma camada intermediária entre os servidores centrais e os dispositivos finais. Esse paradigma atua de forma descentralizada, aproximando os serviços tradicionalmente oferecidos pela nuvem da borda da rede e intermediando a comunicação entre os usuários e os servidores centrais (BURHAN et al., 2023). Dessa forma, a Computação em Névoa opera de maneira hierárquica em conjunto com a Computação em Nuvem, facilitando a comunicação entre os serviços centralizados e os dispositivos finais, geralmente representados por dispositivos da Internet das Coisas (IoT, do inglês, *Internet of Things*) (BELLAVISTA et al., 2019). Entre suas principais características destacam-se a baixa latência, o conhecimento da localização dos dispositivos, a arquitetura distribuída, a escalabilidade, a proximidade física em relação aos usuários, a maior disponibilidade de largura de banda, além de mecanismos voltados à segurança e à privacidade (BURHAN et al., 2023).

2.2.3 Computação de Borda

A Computação de Borda consiste em aproximar do usuário final os serviços tradicionalmente executados em ambientes de Computação em Nuvem, transferindo o processamento e a execução das aplicações para a borda da rede. Essa abordagem reduz o tempo de resposta das aplicações e o atraso no processamento das tarefas, tornando os serviços mais adequados para aplicações sensíveis à latência. Entretanto, os dispositivos de borda normalmente possuem menor capacidade de armazenamento e menor poder computacional quando comparados aos servidores de nuvem (KHAN et al., 2019).

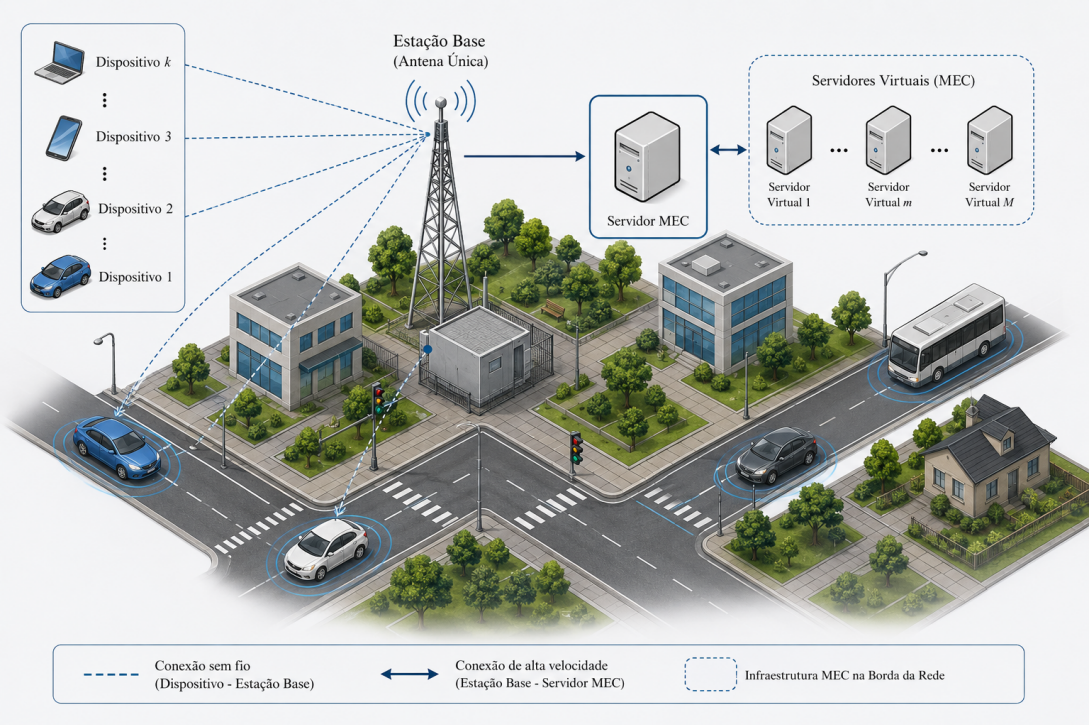
2.2.4 Computação de Borda Móvel

Entre as aplicações da Computação de Borda, destaca-se a Computação de Borda Móvel, destinada ao atendimento de dispositivos móveis por meio de serviços disponibilizados em estações rádio-base de redes celulares (KHAN et al., 2019). Esse paradigma caracteriza-se pelo processamento das aplicações diretamente na borda da rede, proporcionando baixa latência, maior disponibilidade de largura de banda e comunicação por meio da infraestrutura celular. Em contrapartida, os dispositivos de borda apresentam capacidade de armazenamento e processamento mais limitada quando comparados aos ambientes de Computação em Nuvem (HUSSEIN et al., 2022).

Na Figura 4 apresenta-se um exemplo de Computação de Borda Móvel, no qual dispositivos móveis comunicam-se com uma estação rádio-base, responsável por desem-

penhar o papel de servidor de borda para o processamento das aplicações (FU; DING; JIANG, 2022).

Figura 4 – Exemplo de Computação de Borda Móvel.



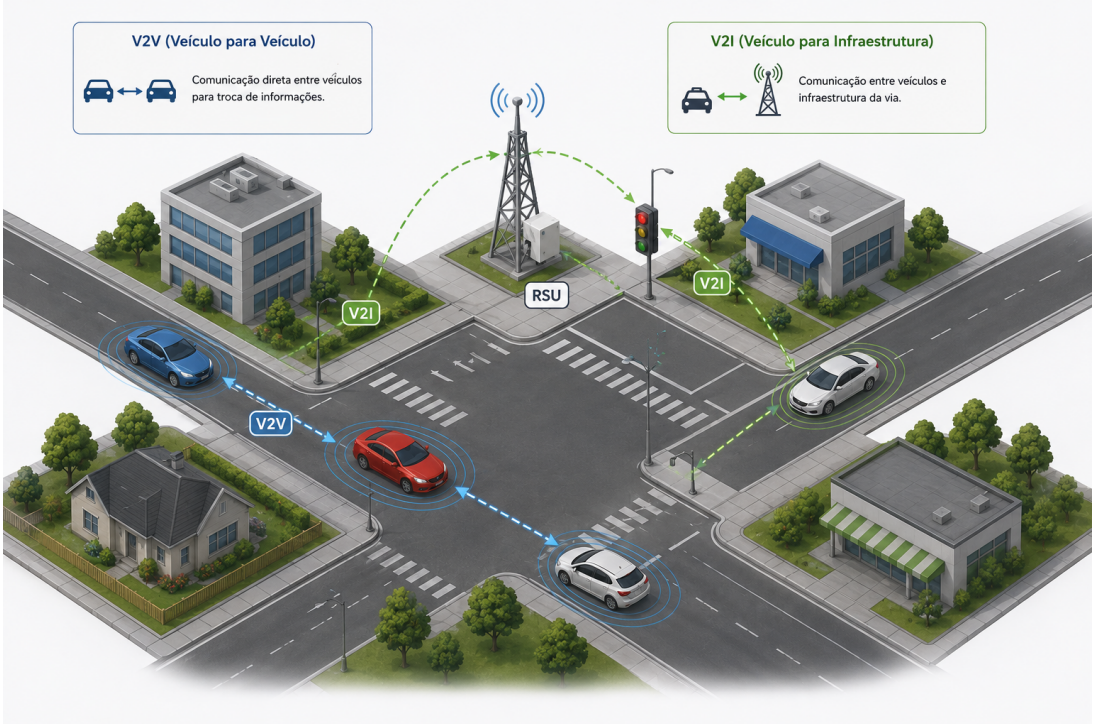
Fonte: Gerado por IA.

2.2.5 Computação em Nuvem Veicular

A Computação em Nuvem Veicular (VCC, do inglês, *Vehicular Cloud Computing*) consiste em um paradigma no qual um conjunto de veículos, equipados com dispositivos de bordo, disponibiliza seus recursos de computação e comunicação para serem compartilhados com os serviços centralizados de Computação em Nuvem. A principal proposta da VCC é aproveitar os recursos computacionais ociosos existentes em veículos em circulação nas vias urbanas, rodovias e estacionamentos, ampliando a capacidade de processamento disponível para as aplicações (COSTA et al., 2020a; WEI et al., 2022). Entre as principais características desse paradigma destacam-se a necessidade de conexão contínua com a nuvem central, a elevada latência, a menor largura de banda disponível e a capacidade de lidar com grandes volumes de dados (WAHEED et al., 2022).

Na Figura 5 apresenta-se um exemplo de uma arquitetura de Computação em Nuvem Veicular, na qual comunicações V2V e V2I são utilizadas para compartilhar recursos computacionais com a nuvem central de forma integrada.

Figura 5 – Exemplo de VCC.



Fonte: Gerado por IA.

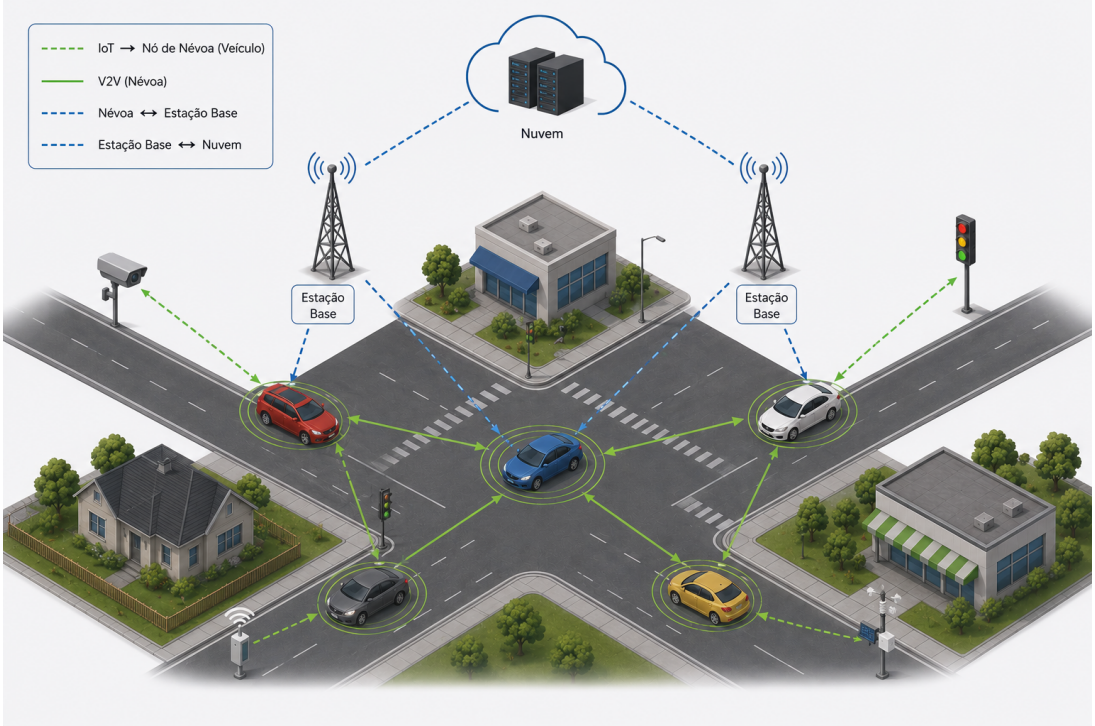
2.2.6 Computação em Névoa Veicular

A Computação em Névoa Veicular consiste em uma adaptação do paradigma de Computação em Névoa para atender às necessidades específicas das redes veiculares. Seu principal objetivo é lidar com os desafios impostos pela elevada mobilidade dos veículos, aproximando os serviços de processamento e armazenamento da borda da rede, de modo a oferecer maior largura de banda e baixa latência (KESHARI; SINGH; MAURYA, 2022). Na Figura 6 ilustra-se uma arquitetura de Computação em Névoa Veicular, na qual os veículos atuam como nós de névoa (*fog nodes*) e, em conjunto com estações rádio-base de redes celulares, intermedeiam a comunicação entre dispositivos da Internet das Coisas (IoT) e a Computação em Nuvem (XIAO; ZHU, 2017).

2.2.7 Computação de Borda Veicular

A Computação de Borda Veicular (VEC) é frequentemente considerada uma extensão da Computação em Névoa, uma vez que compartilha diversas de suas características, como a baixa latência, o processamento realizado próximo ao usuário final, a maior disponibilidade de largura de banda e a limitação dos recursos computacionais disponíveis (WAHEED et al., 2022). Entretanto, a colaboração entre veículos vizinhos e a necessidade de adaptação às constantes mudanças provocadas pela mobilidade veicular conferem características particulares à VEC, diferenciando-a dos demais paradigmas de computação distribuída. Como consequência, esse ambiente apresenta elevada complexi-

Figura 6 – Exemplo de Computação em Névoa Veicular.



Fonte: Gerado por IA.

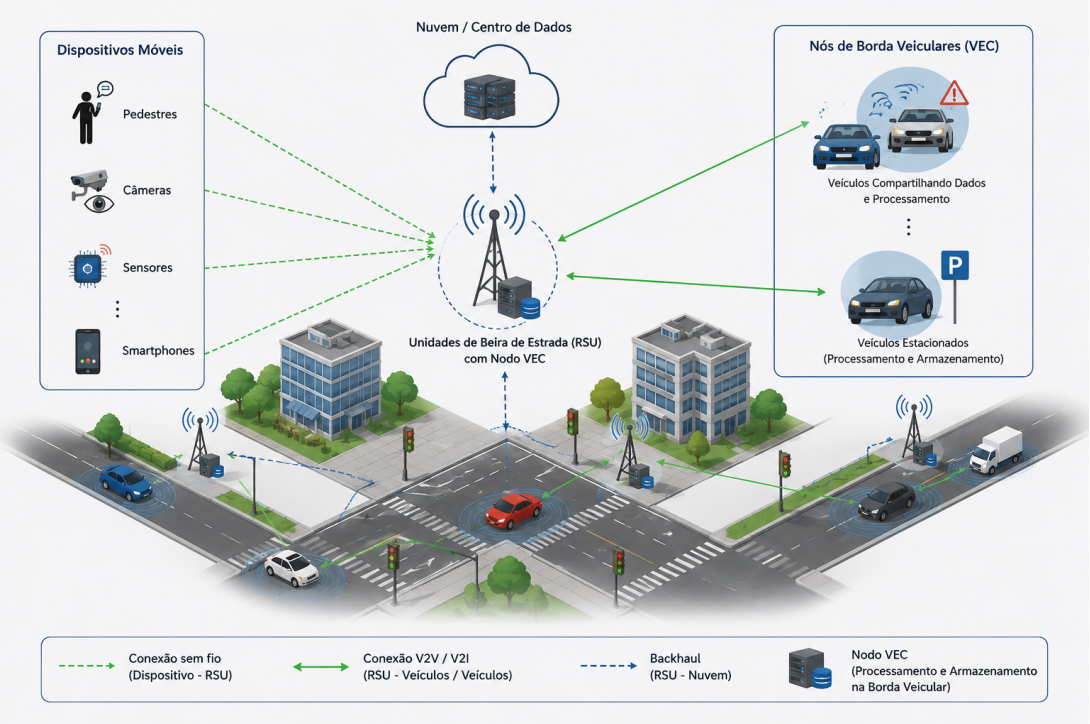
dade e exige mecanismos capazes de realizar tomadas de decisão em tempo real para o gerenciamento eficiente dos recursos computacionais (MENEGUETTE et al., 2021).

Na Figura 7 apresenta-se um exemplo de um ambiente de Computação de Borda Veicular utilizando comunicação V2I. Nesse cenário, veículos em deslocamento com baixa velocidade e veículos estacionados comunicam-se com uma RSU, compartilhando seus recursos computacionais ociosos para a execução de aplicações distribuídas. Simultaneamente, dispositivos IoT estabelecem comunicação com a RSU para solicitar a alocação de serviços e tarefas computacionais na rede (BOUKERCHE; SOTO, 2020). Nesse paradigma, os veículos atuam como nós de borda, disponibilizando seus recursos para atender às demandas das aplicações. Entretanto, a elevada mobilidade dos veículos e a constante alteração da topologia da rede tornam o gerenciamento da comunicação e dos recursos computacionais um desafio significativo, motivando o desenvolvimento de novas soluções por parte da comunidade científica (MENEGUETTE et al., 2021; BOUKERCHE; SOTO, 2020).

2.3 ALOCAÇÃO E GERENCIAMENTO DE TAREFAS EM VEC

O crescente número de veículos equipados com computadores de bordo, conectados entre si e integrados à infraestrutura de Computação de Borda Veicular, torna

Figura 7 – Exemplo de VEC.



Fonte: Gerado por IA.

indispensável que os serviços oferecidos sejam processados de forma eficiente e com baixa latência (MENEGUETTE et al., 2021). Nesse contexto, garantir a troca eficiente de informações, bem como a alocação e o processamento de tarefas de maneira a atender às necessidades dos usuários, constitui um dos principais desafios dos Sistemas de Transportes Inteligentes. Assim, busca-se explorar de forma eficiente os recursos compartilhados na rede, visando otimizar seu desempenho e melhorar a qualidade dos serviços oferecidos (LIU et al., 2021).

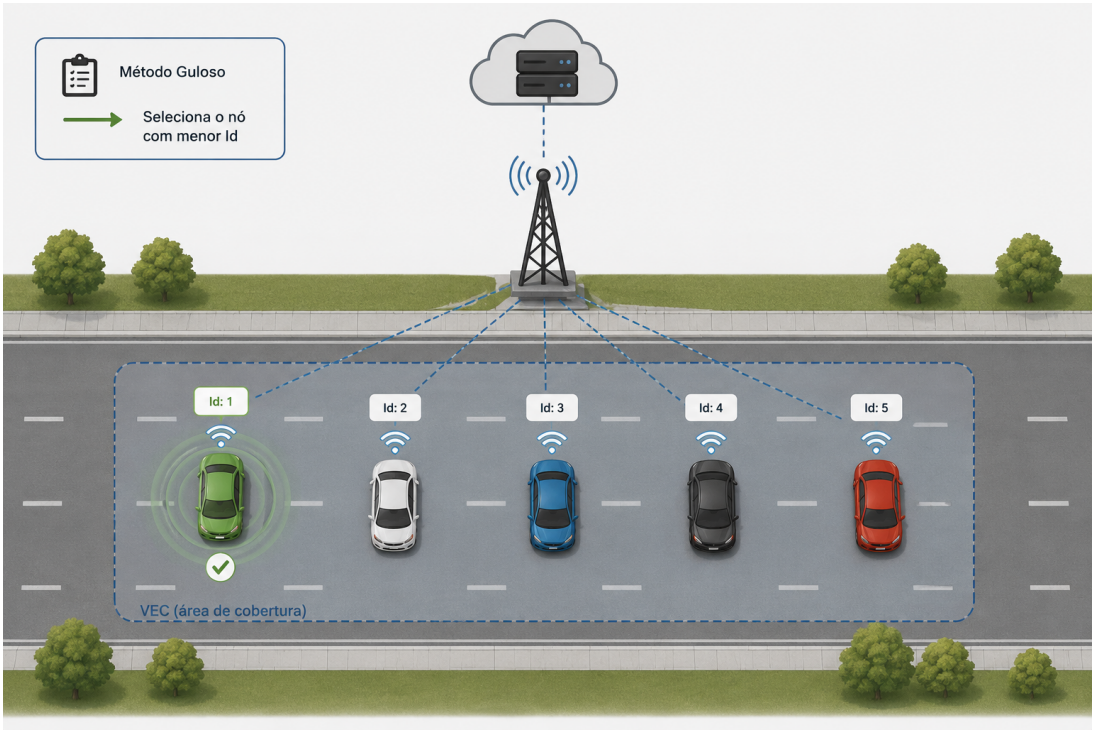
Diversos processos compõem o gerenciamento de recursos em ambientes de Computação de Borda Veicular, com o objetivo de aumentar a eficiência na utilização dos recursos computacionais disponíveis. Entre eles, destacam-se a seleção, a distribuição e o processamento de tarefas, que devem ser realizados de forma a reduzir o atraso na execução das aplicações e maximizar a utilização dos recursos disponibilizados pelos nós de borda (LIEIRA et al., 2023b). Outro componente fundamental é o agendamento de tarefas, responsável por organizar as requisições para que sejam processadas no momento mais adequado, contribuindo para a garantia da qualidade de serviço oferecida aos usuários da VEC. Além disso, esse processo busca melhorar o aproveitamento dos recursos computacionais e o desempenho geral da rede (HOUSSEIN et al., 2021).

Em Computação de Borda Veicular, também é essencial que o processo de descarregamento de tarefas seja realizado de forma eficiente. Quando um veículo não possui recursos suficientes para executar uma determinada tarefa localmente, ele pode enca-

minhá-la à VEC, que, por sua vez, é responsável por identificar um veículo vizinho capaz de disponibilizar seus recursos computacionais ociosos para o processamento da requisição (HAMDI; HUSSAIN; HUSSAIN, 2022). Nesse contexto, a etapa de tomada de decisão para a alocação de tarefas computacionais desempenha um papel fundamental, pois influencia diretamente a utilização dos recursos disponíveis na VEC e a redução da latência do sistema (LIEIRA et al., 2023b).

Diversas técnicas têm sido propostas para otimizar o processo de alocação de tarefas computacionais. Entre os métodos tradicionais destaca-se o algoritmo Guloso (*Greedy*), denominado dessa forma por realizar a seleção sem empregar mecanismos adicionais de otimização. Nesse método, o primeiro elemento do conjunto de candidatos é escolhido para atender à solicitação de processamento da tarefa (NGUYEN et al., 2021). Na Figura 8 ilustra-se um exemplo desse processo de seleção, no qual uma VEC é composta por cinco veículos, e o algoritmo seleciona o primeiro elemento do conjunto, correspondente ao veículo com o menor identificador (Id), para executar a tarefa requisitada.

Figura 8 – Exemplo de seleção do método Guloso.

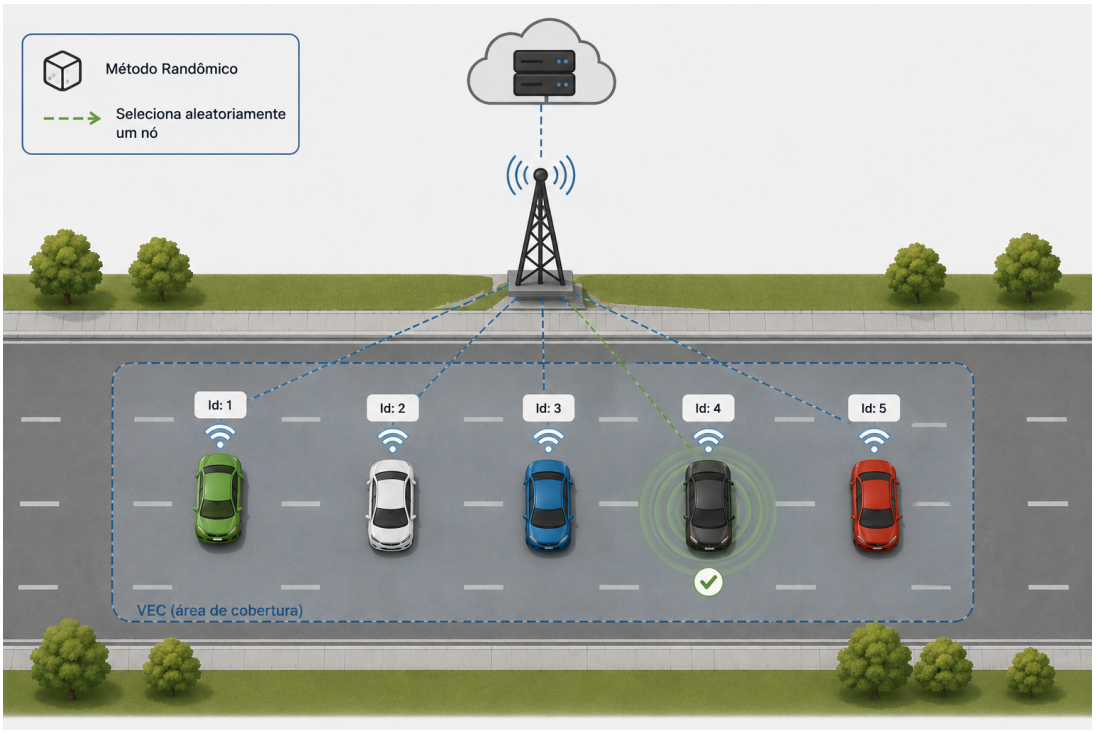


Fonte: Gerado por IA.

O método Randômico também é uma abordagem tradicional utilizada na literatura para o problema de alocação de tarefas. Assim como o método Guloso, ele não emprega mecanismos sofisticados de seleção. Entretanto, sua principal diferença consiste na escolha aleatória do nó de borda responsável pelo atendimento da tarefa, dentre todos os veículos disponíveis na VEC. Na Figura 9 apresenta-se um exemplo desse processo de seleção, no qual o veículo com $Id = 4$ foi escolhido aleatoriamente para atender à

requisição, sem considerar critérios como ordem de entrada na rede, disponibilidade de recursos computacionais ou qualquer outro parâmetro de decisão.

Figura 9 – Exemplo de seleção do método Randômico.



Fonte: Gerado por IA.

A modelagem matemática também é amplamente utilizada em processos de tomada de decisão. Um dos métodos mais empregados para esse fim é o Processo de Hierarquia Analítica (AHP, do inglês, *Analytic Hierarchy Process*). O AHP organiza o problema em uma estrutura hierárquica composta pelo objetivo principal, critérios de avaliação e alternativas de solução. Em seguida, utiliza matrizes de comparação par a par para atribuir pesos aos critérios e alternativas, aplicando conceitos matemáticos que auxiliam na tomada de decisão (PEREIRA et al., 2020).

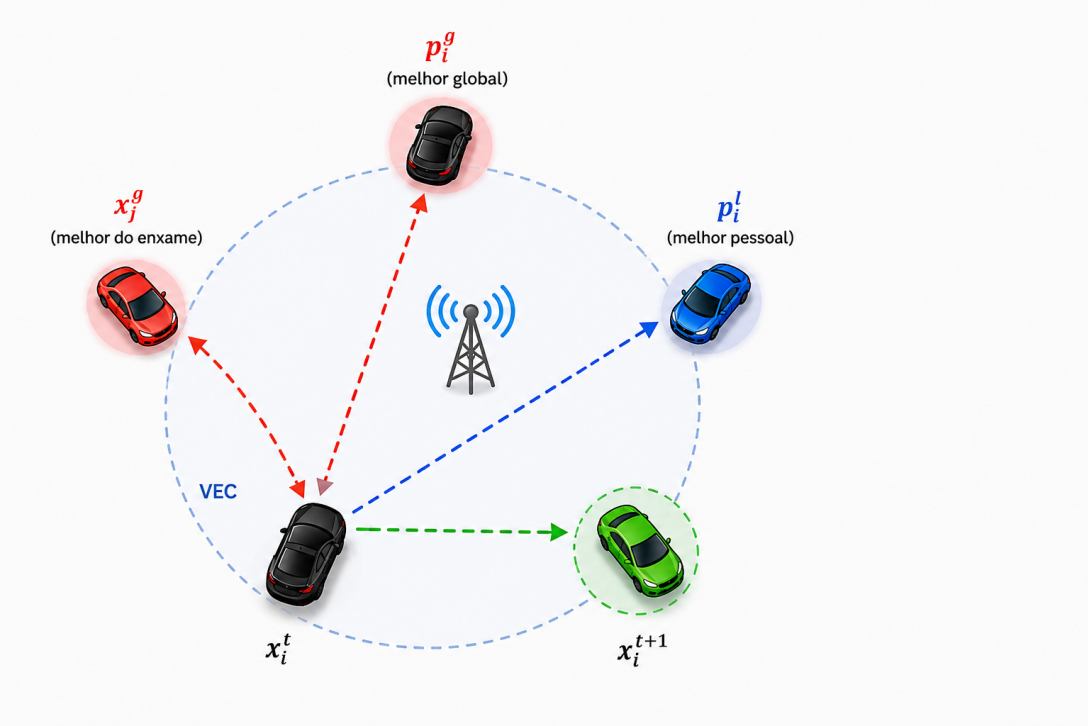
Mais recentemente, os algoritmos meta-heurísticos têm se destacado como alternativas eficientes para processos de tomada de decisão em ambientes de Computação de Borda Veicular (LIEIRA et al., 2023b). Esses algoritmos são inspirados em fenômenos observados na natureza, como processos evolutivos, comportamento coletivo de enxames, mecanismos genéticos e interações entre indivíduos. Seu principal objetivo é explorar o espaço de busca de um problema em busca de soluções de alta qualidade, aproximando-se da solução ótima global. Devido à sua simplicidade, flexibilidade e capacidade de adaptação, os algoritmos meta-heurísticos são considerados abordagens eficazes para aplicação em cenários complexos e dinâmicos, como os ambientes de VEC (TOMAR; BANSAL; SINGH, 2023).

O algoritmo de Otimização por Enxame de Partículas (PSO, do inglês, *Particle*

Swarm Optimization) é um método meta-heurístico de otimização estocástica baseado em inteligência de enxame. Sua inspiração é derivada do comportamento coletivo observado em grupos de animais, como bandos de pássaros, cardumes de peixes e rebanhos. Para encontrar soluções de qualidade, o algoritmo realiza sucessivas iterações nas quais as partículas ajustam suas posições com base nas informações obtidas durante o processo de busca (CHEN et al., 2020). Cada partícula representa uma solução candidata e participa ativamente do processo de otimização, explorando o espaço de busca em conjunto com as demais partículas do enxame (WANG; TAN; LIU, 2018).

O PSO é amplamente adotado por pesquisadores devido à sua simplicidade de implementação, flexibilidade de aplicação em diferentes problemas e capacidade de convergência para soluções satisfatórias sem a necessidade de informações adicionais, como derivadas ou funções contínuas diferenciáveis. Além disso, esse meta-heurístico concentra sua busca no compartilhamento de informações obtidas a partir das melhores soluções individuais e globais encontradas pelo enxame ao longo das iterações, em vez de depender exclusivamente de procedimentos matemáticos complexos (WANG; TAN; LIU, 2018). Na Figura 10 apresenta-se um exemplo do processo de movimentação das partículas no algoritmo PSO, no qual um nó de borda utiliza a influência das demais partículas do enxame para determinar sua nova posição no espaço de busca.

Figura 10 – Exemplo de movimentação do algoritmo PSO.

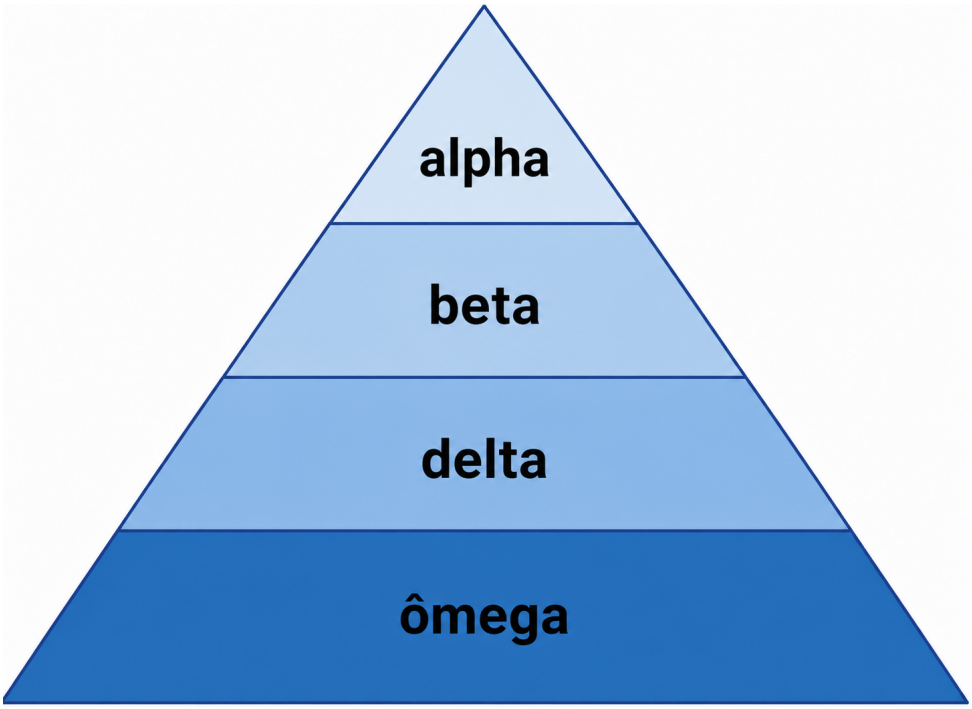


Fonte: Gerado por IA.

O algoritmo meta-heurístico de Otimização do Lobo Cinzento (GWO, do inglês, *Grey Wolf Optimizer*) é inspirado no comportamento social e na estratégia de caça dos

lobos cinzentos. Esses animais vivem em alcateias e ocupam o topo da cadeia alimentar, apresentando uma estrutura hierárquica bem definida. A organização da alcateia é representada por uma hierarquia composta por quatro níveis, conforme ilustrado na Figura 11: *alpha*, que representa o líder da alcateia e é responsável pela tomada das principais decisões; *beta*, que auxilia o líder e atua como seu principal subordinado; *delta*, responsável por funções como proteção, vigilância e organização da alcateia, obedecendo aos níveis superiores; e *omega*, que corresponde aos demais membros da alcateia e ocupa o nível hierárquico mais baixo (MIRJALILI; MIRJALILI; LEWIS, 2014).

Figura 11 – Pirâmide de hierarquia no GWO.



Fonte: Adaptado de (MIRJALILI; MIRJALILI; LEWIS, 2014).

Além da organização hierárquica, o GWO desperta o interesse da comunidade científica devido à estratégia cooperativa empregada durante o processo de caça. De acordo com (MIRJALILI; MIRJALILI; LEWIS, 2014), o algoritmo modela esse comportamento por meio de três etapas principais:

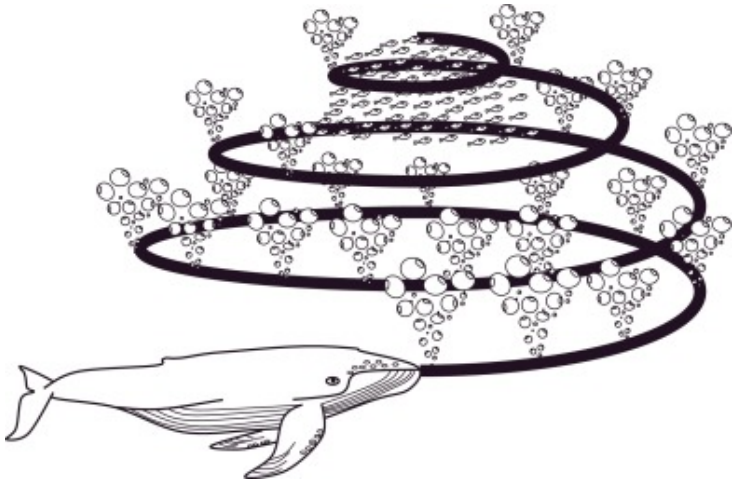
- rastrear, perseguir e aproximar-se da presa;
- cercar e assediar a presa até restringir seus movimentos;
- atacar a presa.

O algoritmo meta-heurístico de Otimização da Baleia (WOA, do inglês, *Whale Optimization Algorithm*) é inspirado no comportamento de caça das baleias-jubarte. Esse algoritmo considera características observadas nessa espécie, como sua elevada capacidade

cognitiva e seu comportamento social, que pode ocorrer tanto em grupos quanto de forma individual (MIRJALILI; LEWIS, 2016; GHAREHCHOPOGH; GHOLIZADEH, 2019).

A principal inspiração do WOA está na estratégia de alimentação das baleias-jubarte, que frequentemente caçam pequenos peixes próximos à superfície do mar utilizando uma técnica conhecida como alimentação por rede de bolhas. Nesse processo, as baleias nadam ao redor da presa, formando um padrão espiralado de bolhas semelhante ao formato do número "9", conforme representado na Figura 12 em um cenário de VEC. A partir da identificação da melhor solução do grupo, o algoritmo simula esse comportamento por meio de dois mecanismos principais: o movimento em espiral ascendente e o movimento de aproximação da presa, reproduzindo a estratégia cooperativa utilizada pelas baleias durante a caça (MIRJALILI; LEWIS, 2016).

Figura 12 – Exemplo de movimentação da baleia.



Fonte: (MIRJALILI; LEWIS, 2016).

2.3.1 Compartilhamento de Recursos e Cooperação Veicular

Os algoritmos de tomada de decisão apresentados anteriormente têm sido empregados para solucionar o problema de alocação de tarefas computacionais. Entretanto, para que essas tarefas sejam efetivamente processadas, é indispensável a cooperação entre os veículos que compõem a VEC, por meio do compartilhamento de seus recursos computacionais ociosos. Embora os veículos sejam considerados elementos com elevado potencial para disponibilizar recursos computacionais a aplicações emergentes dos Sistemas de Transportes Inteligentes, ainda enfrentam desafios em cenários realistas de VEC, como limitações de processamento, restrições computacionais, requisitos rigorosos de latência e atendimento de requisições em tempo real (WANG et al., 2022).

Recursos computacionais, como capacidade de processamento, memória, armazenamento e largura de banda, apresentam elevada demanda em ambientes de VEC, especialmente devido às limitações dos dispositivos de bordo (OBUs) instalados nos veículos

(LIEIRA et al., 2023b). Nesse contexto, a clusterização dos recursos computacionais representa uma alternativa para integrar e disponibilizar esses recursos de forma compartilhada. Para que essa abordagem seja viável, é fundamental a cooperação entre os nós de borda da VEC. A cooperação voluntária ocorre quando um veículo, enquanto não utiliza integralmente seus recursos computacionais, opta por disponibilizar sua capacidade ociosa para atender às demandas da rede (YEDDULAPALLI et al., 2024).

2.3.2 Billing

Grandes empresas de tecnologia, como Amazon e Google, adotaram o conceito de *billing* em seus serviços de Computação em Nuvem, oferecendo diferentes modalidades de faturamento, como cobrança por hora, minuto, lote ou período de utilização. Entre essas modalidades, uma das mais difundidas é a cobrança baseada no consumo efetivo dos recursos computacionais utilizados (TOLOSANA-CALASANZ et al., 2016; XIANG et al., 2022). De forma análoga, a implementação de um sistema de *billing* em ambientes de Computação de Borda Veicular pode contribuir para custear a implantação e a manutenção da infraestrutura, além de incentivar os usuários a compartilharem seus recursos computacionais ociosos, reduzindo, consequentemente, os efeitos da escassez de recursos na VEC (LIEIRA et al., 2025).

Outro importante mecanismo de incentivo à colaboração consiste na priorização das tarefas dos usuários quando estes necessitam utilizar os serviços da VEC. Diferentes estratégias têm sido propostas na literatura, como a priorização baseada em perfis de usuários construídos a partir de seus padrões de comportamento (SHOELEH et al., 2021), a definição de perfis considerando preferências, características e hábitos de utilização para atender às necessidades específicas de cada usuário (EKE et al., 2019), e a classificação das tarefas de acordo com seu nível de importância para orientar o processo de tomada de decisão durante a alocação de recursos (LIEIRA et al., 2025). Essas abordagens representam alternativas para custear infraestruturas e podem servir para incentivar a colaboração dos usuários e contribuir para a mitigação da escassez de recursos em ambientes de Computação de Borda Veicular, aplicando um *cashback* pela cooperação com a rede.

A implementação de um sistema *billing* abrange vários fatores, como o faturamento por consumo dos usuários, a segurança das transações, emissão de faturas e notas fiscais, integrações de pagamentos, implementação de segurança, autenticação, auditoria e políticas econômicas. Para o escopo desta tese, o *billing* representa um sistema automático de contabilização e estimativa de compensação pelo uso dos recursos. Além da compensação, o *billing* pode funcionar como um mecanismo de incentivo, pois possibilita estabelecer uma forma de recompensação aos usuários que colaboram disponibilizando seus recursos para a rede.

2.4 CONSIDERAÇÕES PARCIAIS

Com a evolução das tecnologias embarcadas, as empresas automobilísticas passaram a incorporar novos recursos computacionais e de comunicação em seus veículos. Nesse contexto, os Sistemas de Transportes Inteligentes assumiram um papel fundamental na evolução e implementação das Cidades Inteligentes. A comunicação contínua entre veículos em ambientes urbanos gera um grande volume de dados que pode ser explorado para aprimorar os serviços oferecidos pelos ITS. Entretanto, para que esses serviços sejam efetivos, é necessário que sejam eficientes, seguros, apresentem baixa latência e garantam elevados níveis de Qualidade de Serviço (QoS) e Qualidade de Experiência (QoE) aos usuários. A Computação de Borda Veicular surgiu como uma extensão dos paradigmas de Computação em Nuvem, aproximando o processamento das aplicações dos usuários finais. Contudo, a descentralização desse paradigma impõe limitações, principalmente relacionadas à disponibilidade de recursos computacionais. Dessa forma, observa-se um crescente esforço da comunidade científica no desenvolvimento de mecanismos capazes de utilizar esses recursos de maneira mais eficiente.

Diversas abordagens, incluindo técnicas tradicionais, modelos matemáticos, algoritmos heurísticos, meta-heurísticos e métodos baseados em aprendizado de máquina, têm sido empregadas em processos de tomada de decisão para alocação de tarefas em ambientes de Computação de Borda Veicular. Apesar dos avanços obtidos, desafios como a complexidade dos algoritmos, a elevada dinamicidade dos cenários urbanos, o grande volume de dados, as constantes alterações na topologia da rede, o tempo de inferência e a limitação dos recursos computacionais ainda permanecem em aberto. Nos últimos anos, os algoritmos meta-heurísticos têm se destacado como uma alternativa promissora para otimizar os processos de alocação de tarefas em VEC, principalmente devido à sua elevada capacidade de adaptação às mudanças contínuas características desses ambientes. Entretanto, além da adaptação às alterações de topologia, a escassez de recursos computacionais continua sendo um dos principais desafios da área. Nesse sentido, mecanismos de incentivo que recompensem os usuários pelo compartilhamento de seus recursos ou pela execução de tarefas podem contribuir para reduzir esse problema. Esses incentivos podem ser implementados por meio da priorização das tarefas dos usuários colaboradores ou da oferta de recompensas financeiras, utilizando um sistema *billing* capaz de remunerar os participantes de acordo com a quantidade de recursos computacionais compartilhados para a execução das tarefas.

3 ESTADO DA ARTE

Neste capítulo, é apresentado um levantamento bibliográfico dos trabalhos mais recentes que empregam diferentes técnicas para lidar com o problema de alocação e/ou priorização de tarefas computacionais. Cada abordagem analisada foi desenvolvida e avaliada em diferentes cenários de estudo, considerando características específicas e objetivos distintos relacionados à otimização dos recursos computacionais disponíveis. Entretanto, todas as propostas possuem como objetivo comum aprimorar o gerenciamento e a utilização eficiente dos recursos computacionais.

3.1 TRABALHOS RELACIONADOS

Metodologias de seleção eficientes e relevantes para processos de alocação de tarefas são fundamentais para permitir que serviços e aplicações de Computação de Borda Veicular (VEC) sejam capazes de lidar com cenários altamente dinâmicos e em constante mudança. Dessa forma, diversos estudos recentes têm sido desenvolvidos com o objetivo de propor soluções para os desafios relacionados à alocação de tarefas e ao gerenciamento eficiente de recursos computacionais.

Algoritmos meta-heurísticos também têm apresentado resultados promissores em processos de alocação de recursos. O Algoritmo de Otimização do Leão (LOA) foi utilizado como base para o desenvolvimento do LOPRIVE, um mecanismo de alocação de tarefas computacionais em VEC que realiza a priorização de tarefas de acordo com suas categorias (LIEIRA et al., 2025). Entretanto, os autores limitaram o processo de priorização às categorias das tarefas, não considerando aspectos relacionados à latência e vazão no processo de avaliação.

O Algoritmo de Otimização de Baleias (WOA) foi empregado como base para o desenvolvimento do MARTIAL, um mecanismo voltado à maximização da alocação de tarefas em ambientes de Computação de Borda. Os resultados demonstraram desempenho satisfatório quando comparado a algoritmos tradicionais da literatura (LIEIRA et al., 2022a). Entretanto, o estudo utilizou dados sintéticos em seu processo de simulação, não incorporou um simulador de rede e não considerou as características e complexidades de cenários baseados em comunicação V2I. Além disso, a abordagem não contempla mecanismos de priorização baseados nos usuários.

A meta-heurística de Otimização por Enxame de Partículas (PSO, do inglês, *Particle Swarm Optimization*) foi aplicada ao problema de agendamento de tarefas em ambientes de Computação em Nuvem, utilizando o simulador CloudSim-3.03. O objetivo

da abordagem foi minimizar o tempo total de execução das tarefas e reduzir o consumo energético das máquinas virtuais. Os resultados obtidos indicaram que o mecanismo apresentou desempenho satisfatório em relação ao tempo de execução e ao *makespan*, quando comparado a outros mecanismos de agendamento encontrados na literatura (KUMAR et al., 2022). Entretanto, o algoritmo não considera mecanismos de priorização de tarefas baseados no compartilhamento de recursos entre usuários.

O ATARY é um algoritmo inspirado no comportamento de caça do algoritmo de Otimização do Lobo Cinzento (GWO), desenvolvido para o problema de alocação de tarefas em ambientes de Computação de Borda Veicular (LIEIRA et al., 2023a). Os autores avaliaram a capacidade do mecanismo em atender às demandas de tarefas e utilizar os recursos computacionais disponíveis. Entretanto, a abordagem não considera cenários altamente dinâmicos e não apresenta mecanismos direcionados à priorização de tarefas durante o processo de alocação.

O método Guloso tradicional serviu como base para a implementação dos algoritmos PSG e PSG-M, aplicados ao processo de agendamento de tarefas em ambientes de Internet das Coisas (IoT) sobre uma rede de Computação em Névoa (Fog) (AZIZI et al., 2022). Os resultados obtidos, após a comparação com algoritmos tradicionais, indicaram eficiência em métricas relacionadas ao cumprimento de prazos, consumo total de energia, tempo de violação de prazo e *makespan*. Entretanto, aspectos como a quantidade de tarefas concluídas, a utilização dos recursos computacionais e a priorização de usuários com base no compartilhamento de recursos não foram considerados como foco principal da abordagem.

Um algoritmo baseado no Processo de Hierarquia Analítica (AHP), matrizes de prioridade e Tomada de Decisão Multicritério foi desenvolvido para tratar o problema de alocação de recursos em ambientes de Computação em Nuvem Veicular (PEREIRA et al., 2020). O método foi avaliado considerando métricas relacionadas à quantidade de recursos alocados, negados e bloqueados, apresentando resultados satisfatórios quando comparado a métodos tradicionais. Entretanto, a avaliação do mecanismo não foi realizada em um simulador de rede considerando um cenário dinâmico, nem contemplou mecanismos de priorização de usuários baseados no compartilhamento de recursos.

O AHP também foi combinado com Lógica *Fuzzy* em um algoritmo destinado à otimização do agendamento de tarefas em ambientes de Computação de Borda. O mecanismo utiliza o AHP para atribuir pesos aos critérios de priorização no processo de seleção das tarefas, com o objetivo de reduzir o consumo energético (HOSSEINI; NICKRAY; GHANBARI, 2022). As métricas utilizadas para avaliação consideraram o tempo de espera, tempo de resposta, atraso e quantidade de tarefas agendadas, apresentando resultados satisfatórios. Entretanto, o algoritmo não foi aplicado em um ambiente de comunicação VEC, que apresenta desafios específicos relacionados à mobilidade dos veículos,

dinamicidade da topologia e limitação dos recursos computacionais disponíveis.

Um mecanismo denominado Harmônico, baseado na Teoria dos Jogos, foi desenvolvido com o objetivo de maximizar o uso equilibrado dos recursos em múltiplas Nuvens de Veículos (VCs). O algoritmo utiliza o valor de Shapley para realizar a ordenação das tarefas durante o processo de alocação (RIBEIRO et al., 2023). Durante a avaliação de desempenho, o mecanismo foi comparado com algoritmos de alocação existentes e apresentou melhores resultados em diferentes cenários e restrições de recursos utilizados nas simulações. Entretanto, a abordagem não teve como objetivo principal aplicar prioridades de alocação baseadas na participação e colaboração dos usuários.

Uma técnica de alocação de múltiplas tarefas em Computação de Borda para veículos inteligentes foi desenvolvida com base na Otimização por Enxame de Partículas Quânticas (SHAO et al., 2023). A proposta apresentou eficiência no processo de execução das tarefas, considerando métricas relacionadas à latência, sobrecarga, transmissão, prioridade e utilização de recursos. Contudo, o trabalho não contempla mecanismos de recompensa aos usuários pelo compartilhamento de recursos, nem propõe a implementação de um sistema *billing*.

Um mecanismo de descarregamento e alocação de recursos foi desenvolvido com o objetivo de facilitar a implementação de Computação de Borda Móvel (MEC) em redes de Veículos Aéreos Não Tripulados (UAVs). Para isso, os autores utilizaram algoritmos baseados em programação convexa sequencial e programação quadrática sequencial com aprendizado por reforço profundo baseado em Q-learning (LIU; LIU; PENG, 2023). Os resultados demonstraram que o mecanismo foi capaz de minimizar o custo ponderado relacionado à latência e ao consumo energético. Entretanto, a abordagem não teve como foco principal a maximização da quantidade de tarefas atendidas, a utilização eficiente dos recursos disponíveis ou a priorização de tarefas baseada no compartilhamento de recursos.

Outro esquema de descarregamento e alocação de recursos foi proposto com foco no descarregamento seguro e colaborativo em ambientes de Computação de Borda Veicular (JU et al., 2023). O mecanismo foi desenvolvido utilizando aprendizado por reforço profundo e buscou otimizar aspectos relacionados ao atraso de processamento, potência de transmissão, seleção do espectro de frequência e tomada de decisão para alocação de recursos computacionais. Entretanto, o objetivo central do trabalho não contempla mecanismos de priorização de tarefas nem estratégias de recompensa aos usuários pelo compartilhamento de recursos computacionais.

Uma nova arquitetura holística de virtualização de redes, baseada em gêmeo digital e fatiamento de rede, foi proposta por (GONG et al., 2023) para o gerenciamento da alocação de recursos computacionais. A arquitetura utiliza um processo de tomada de decisão baseado em Cadeias de Markov e um algoritmo aprimorado que combina o valor de Shapley-Q com gradiente de política determinístico profundo para realizar a alocação

dos recursos disponíveis. Os resultados obtidos demonstraram desempenho satisfatório na redução do tempo de resposta e na otimização da largura de banda da rede sem fio e dos requisitos das tarefas. Entretanto, a abordagem proposta não teve como objetivo principal maximizar a quantidade de tarefas atendidas e o aproveitamento dos recursos computacionais disponíveis, além de não contemplar mecanismos de priorização de usuários ou a implementação de um sistema *billing*.

Um algoritmo baseado em aprendizagem por reforço profundo com restrições de Lyapunov foi desenvolvido para realizar a alocação e o descarregamento de tarefas em redes veiculares (KUMAR; ZHAO; FERNANDO, 2023). Os resultados indicaram que a técnica apresentou desempenho satisfatório na distribuição de tarefas computacionais e na alocação de recursos de comunicação, contribuindo para a redução do consumo energético e dos requisitos de atraso. Entretanto, a abordagem proposta não foi direcionada ao processo de priorização de tarefas durante a alocação dos recursos.

Um algoritmo denominado MALTREAT, baseado no algoritmo meta-heurístico de Otimização do Falcão de Harris (HHO, do inglês, *Harris Hawk Optimization*), foi desenvolvido com o objetivo de otimizar o processo de busca por recursos computacionais ociosos em um cenário de computação urbana utilizando redes veiculares ad-hoc com comunicação V2V e V2I. Os resultados das simulações demonstraram eficiência no processo de alocação de recursos, apresentando aproveitamento satisfatório dos recursos disponíveis e redução da quantidade de serviços negados (GOTTSFRITZ et al., 2023).

Outro algoritmo meta-heurístico foi utilizado no desenvolvimento do SIGHT. A abordagem emprega o algoritmo de morcego (*Bat Algorithm*) como base para a alocação de tarefas em ambientes de Computação em Névoa com comunicação V2V e V2I (QUESSADA et al., 2026). Os resultados obtidos demonstraram eficiência na alocação de tarefas, utilização dos recursos computacionais e redução da quantidade de tarefas negadas. Entretanto, em ambos os trabalhos, os autores não abordaram mecanismos de priorização da alocação baseados no compartilhamento de recursos entre usuários, tampouco estratégias de recompensa por meio da implementação de um sistema *billing*.

Um mecanismo baseado em funções de pertinência fuzzy foi desenvolvido para classificação de tarefas considerando critérios como prazo de execução, distância e problemas relacionados à mobilidade. Além disso, o método utiliza um algoritmo de seleção baseado em Q-learning para identificar o melhor nó responsável por atender às demandas em um ambiente de Computação em Névoa Veicular (VFC, do inglês, *Vehicular Fog Computing*) (EHSAN et al., 2026). O objetivo da abordagem é reduzir o tempo de espera das tarefas e otimizar o *throughput* do sistema. O mecanismo foi avaliado em diferentes cenários e condições de operação, apresentando resultados satisfatórios. Entretanto, o processo de priorização não considera incentivos baseados no compartilhamento de recursos pelos usuários e não contempla mecanismos de recompensa financeira utilizando um

sistema *billing*.

Na Tabela 1 apresenta-se um resumo dos trabalhos relacionados discutidos anteriormente, considerando o algoritmo utilizado, o tipo de estrutura aplicada, o principal objetivo da abordagem e os problemas que cada proposta busca solucionar.

Observa-se que os trabalhos de (AZIZI et al., 2022; KUMAR et al., 2022; HOSSEINI; NICKRAY; GHANBARI, 2022; LIU; LIU; PENG, 2023; GONG et al., 2023) concentram-se principalmente na otimização do consumo energético e na avaliação de métricas relacionadas ao cumprimento de prazos, tempo de execução e *makespan*. Entretanto, essas abordagens não direcionam suas análises para a quantidade de tarefas concluídas e os recursos efetivamente utilizados, métricas importantes para compreender a distribuição dos recursos computacionais e a capacidade de atendimento das demandas da rede.

Os trabalhos de (PEREIRA et al., 2020) e (LIEIRA et al., 2022a) utilizam cenários de simulação sintéticos, não considerando as complexidades presentes em ambientes reais de simulação, como alterações de topologia, mobilidade dos dispositivos e variações na quantidade e disponibilidade dos recursos computacionais ao longo do tempo.

Por outro lado, os trabalhos de (QUESSADA et al., 2026; EHSAN et al., 2026; LIEIRA et al., 2025; RIBEIRO et al., 2023; LIEIRA et al., 2023a; SHAO et al., 2023; JU et al., 2023; KUMAR; ZHAO; FERNANDO, 2023) têm como objetivo otimizar a quantidade de tarefas alocadas e a utilização dos recursos disponíveis. Entretanto, essas abordagens não contemplam mecanismos de priorização de alocação baseados na colaboração dinâmica dos usuários, aspecto que pode contribuir para reduzir a escassez de recursos computacionais e incentivar os usuários a compartilharem seus recursos com a rede.

3.2 CONSIDERAÇÕES PARCIAIS

Com base nos trabalhos do estado da arte sobre alocação de recursos/tarefas, implementou-se, como fator diferenciador, o framework PRIOBILL, um mecanismo dinâmico para alocação de tarefas em VEC com priorização de usuários e um sistema de faturamento. No cenário PRIOBILL, cada vez que um usuário entra em um VEC, o número de vezes que ele atende aos requisitos de serviço de uma tarefa e a quantidade de recursos que ele compartilha com a rede são contabilizados. O algoritmo de tomada de decisão utiliza os dados para priorizar tarefas no momento das solicitações e para equilibrar a distribuição das solicitações de serviço. Além disso, o faturamento é baseado na quantidade de recursos que os usuários compartilham.

O mecanismo de tomada de decisão para alocação de tarefas utiliza o algoritmo

metaheurístico LOA em um cenário VEC V2I. A inspiração no LOA deriva de seu modelo hierárquico e de sua fácil adaptação a diferentes cenários, especialmente neste trabalho, onde um conjunto de tarefas solicita serviço da RSU, que executa o método de decisão e seleciona o veículo mais adequado para atender à demanda. O PRIOBILL foi desenvolvido para selecionar o melhor nó de borda para processar uma tarefa em um cenário VEC. Os atributos dos nós de borda são usados como parâmetros para garantir que o processo de tomada de decisão considere o balanceamento de carga e a distribuição equitativa do serviço.

Tabela 1 – Abstração do estado da arte.

Trabalho	Método	Estrutura	Simulador	Objetivos Principais	Prioridade Billing
Lieira et al. (2025)	LOA	VEC	Omnet++/Veins/SUMO	Maximização de tarefas com priorização por categoria	
Lieira et al. (2022a)	WOA	EC	Simulação sintética	Maximização de recursos	
Kumar et al. (2022)	PSO	VM	Cloudsim-3.03	Latência e consumo de energia	
Lieira et al. (2023a)	WOA	VEC	Omnet++/Veins/SUMO	Maximização de recursos	
Azizi et al. (2022)	Greedy	FOG-IoT	Não especificado	Consumo de energia	
Pereira et al. (2020)	AHP	VEC	Simulação sintética	Maximização de recursos	
Hosseini, Nickray e Ghanbari (2022)	AHP-Fuzzy	EC	Não especificado	Consumo de energia	
Ribeiro et al. (2023)	Teoria dos jogos	VC	Não especificado	Maximizar e equilibrar recursos	
Shao et al. (2023)	QPSO	EC	Não especificado	Maximização de tarefas, priorização e utilização de recursos	
Gottsfritz et al. (2023)	HHO	VANETS/V2V-V2I	Omnet++/Veins/SUMO	Maximização de recursos	
Quessada et al. (2026)	BAT	FOG/V2V-V2I	Omnet++/Veins/SUMO	Maximização de recursos	
Ehsan et al. (2026)	Fuzzy/Q-Learning	VFC	Omnet++/Veins/SUMO	Tempo de espera e throughput	
Ju et al. (2023)	Aprendizado por reforço	VEC	Não especificado	Latência, transmissão e seleção de tarefas	
PRIOBILL	LOA	VEC-V2I	SUMO/Veins/OMNeT++	Processamento de tarefas, utilização de recursos, latência, *throughput*, priorização e sistema billing	

Fonte: Produzida pelo autor.

4 FRAMEWORK PARA GERENCIAMENTO DE ALOCAÇÃO DE TAREFAS EM COMPUTAÇÃO DE BORDA VEICULAR COM PRIORIDADE POR COLABORAÇÃO E SISTEMA BILLING

Neste capítulo, são apresentados os métodos de pesquisa desenvolvidos com base nos conceitos e informações discutidos ao longo deste trabalho, aplicados a um cenário urbano de Computação de Borda Veicular. O *framework* PRIOBILL é implementado com o objetivo de realizar a clusterização dos elementos do cenário, ordenar as tarefas de acordo com as prioridades definidas, selecionar o nó de borda mais adequado para o processamento das demandas e contabilizar os recursos compartilhados para o sistema de *billing*.

O processo de seleção do melhor nó de borda para execução das tarefas é baseado no algoritmo meta-heurístico bioinspirado Algoritmo de Otimização do Leão. Dessa forma, o PRIOBILL busca enfrentar o problema da limitação de recursos computacionais em ambientes de Computação de Borda Veicular, por meio de um *framework* capaz de priorizar o atendimento das tarefas e recompensar os usuários de acordo com o compartilhamento de seus recursos computacionais com a VEC. Além disso, o mecanismo foi desenvolvido visando maximizar a quantidade de tarefas processadas e otimizar a utilização dos recursos disponibilizados pelos nós de borda.

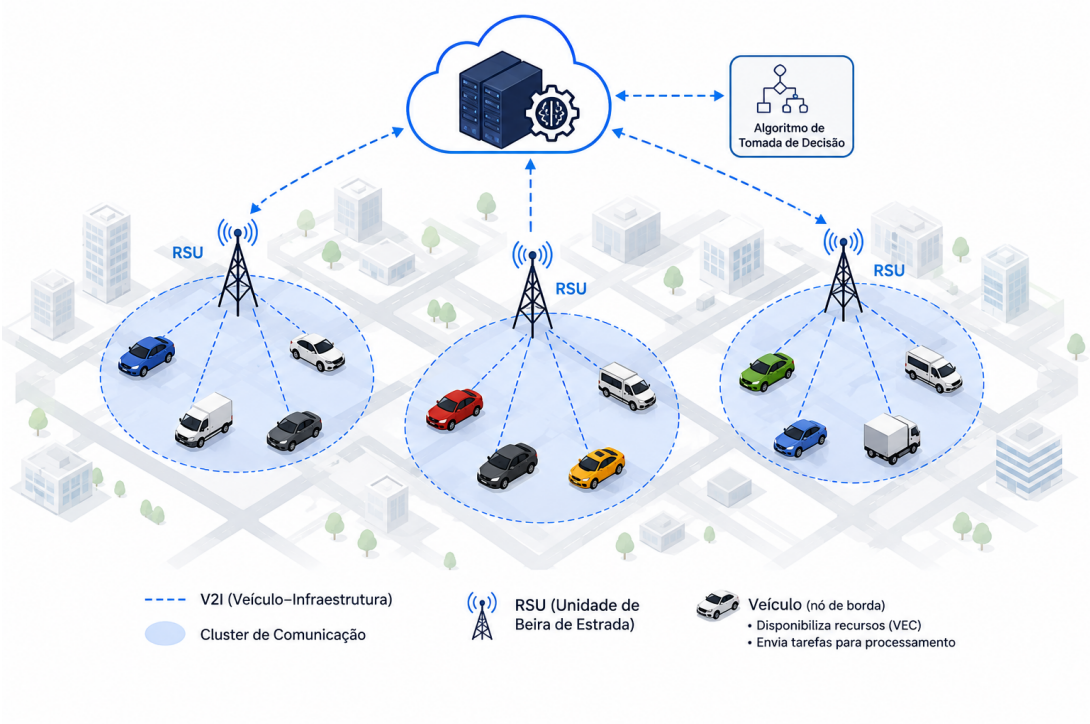
Assim, este capítulo apresenta a definição do cenário de Computação de Borda Veicular utilizado, a formulação do problema abordado, as etapas de desenvolvimento do PRIOBILL, as configurações dos experimentos de simulação e a análise dos resultados obtidos.

Para facilitar a compreensão das equações e dos pseudocódigos apresentados neste trabalho, a Tabela 2 apresenta um resumo das principais notações utilizadas.

Tabela 2 – Sumário de Notações

Símbolo	**Descrição**
$\mathcal{C}$	Conjunto de veículos (membros do *cluster*)
$\mathcal{V}$	Conjunto de veículos dentro da cobertura da RSU
$\mathcal{Q}$	Fila de tarefas
$\mathcal{S}$	Conjunto de tarefas selecionadas
$\mathcal{O}$	Conjunto de tarefas ordenadas
$\mathcal{T}$	Conjunto de informações da tarefa
j	Índice do veículo
j^*	Nó de borda selecionado (melhor)
i	Índice de tarefas
λ	Taxa de chegada de tarefas
T	Número máximo de iterações
M_j, P_j, S_j	Memória, processamento e armazenamento do veículo j
m_i, p_i, s_i	Requisitos de recursos da tarefa i
$R_{\max}$	Recursos máximos disponíveis
$A_{\max}$	Número máximo de tarefas alocadas
$E_{\max}$	Número máximo de entradas nas RSU
A_j	Número de tarefas previamente atribuídas ao veículo j
E_j	Contagem de entrada de veículos j nas RSU
S_j	*Fitness (score)* do veículo j
Cs_i	Pontuação de colaboração do nó de borda
S^*	Melhor fitness (score)
$\mathbf{R}_j$	Vetor de recursos do veículo j (memória, processamento, armazenamento)
$\mathbf{r}_i$	Vetor de demanda de recursos da tarefa i
$\mathbf{x}_j$	Posição do veículo j
$\mathbf{x}^*$	Posição do alvo (tarefa)
r	Identificador RSU
$\mathbf{p}_r$	Posição da RSU
ρ	Contador de iterações de clusterização
θ	Número mínimo de nós para clusterização
N_{alloc}	Número de tarefas processadas
N_{deny}	Número de tarefas não processadas
N_{sent}	Número de tarefas enviadas
msg	Estrutura da mensagem
w_R	Peso para recursos disponibilizados
w_A	Peso para tarefas processadas
w_E	Peso para contador de entradas
$E_{\max}$	Número máximo de entradas em RSU
$A_{\max}$	Número máximo de atendimento efetuados
t_i	Tarefas i
N_t	Número de tarefas chegando
$\prec$	Relação de prioridade entre tarefas
τ_i	Tempo de execução da tarefa t_i
Θ	Taxa de transferência do sistema (*throughput*)
$\sigma(\cdot)$	Função de alocação de tarefas a nós de borda
σ^*	Função de alocação ótima
$\mathcal{S}_{\surd}$	Conjunto de tarefas processadas com sucesso
R_{total}	Total de recursos consumidos pelas tarefas alocadas
M_{alloc}	Total de recursos de memórias alocados
P_{alloc}	Total de recursos de processamento alocadas
S_{alloc}	Total de recursos de armazenamento alocados
T_{sim}	Tempo total de simulação

Figura 13 – Visão geral do modelo de sistema V2I em VEC.



Fonte: Gerado por IA.

4.1 MODELO DE SISTEMA

Neste trabalho, considera-se um cenário urbano composto por estruturas distribuídas ao longo da cidade, com o objetivo de facilitar a comunicação em tempo real, o compartilhamento de dados e o acesso dos veículos aos serviços disponibilizados pela rede. Os veículos localizados dentro da área de cobertura de uma Unidade de Beira de Estrada (RSU) formam um *cluster*, no qual compartilham seus recursos computacionais para compor um *pool* de recursos e podem solicitar capacidade computacional para o processamento de suas tarefas.

Na Figura 13 apresenta-se o modelo de sistema considerado neste trabalho. O cenário urbano é composto por Unidades de Beira de Estrada (RSUs) distribuídas pela cidade, responsáveis por gerenciar a comunicação entre os veículos presentes em suas áreas de cobertura, caracterizando uma comunicação veículo-infraestrutura (V2I). A RSU atua como elemento intermediário da rede, recebendo as solicitações de serviços encaminhadas pelos veículos, executando o algoritmo de tomada de decisão e enviando as respostas aos veículos pertencentes ao seu respectivo *cluster*.

Os veículos, por sua vez, são considerados nós de borda da Computação de Borda Veicular (VEC), podendo disponibilizar seus recursos computacionais ociosos para a rede e encaminhar tarefas para serem processadas pelos recursos compartilhados disponíveis no ambiente VEC.

4.2 FORMULAÇÃO DO PROBLEMA

Com base no cenário urbano apresentado na Figura 13, considera-se um conjunto de quarteirões contendo veículos estacionados e em movimento, formando uma topologia heterogênea e dinâmica. Nesse ambiente, os veículos são equipados com computadores de bordo e podem se deslocar de forma independente pelo cenário, seguindo rotas específicas definidas para cada veículo. As aplicações e serviços solicitados podem apresentar diferentes níveis de complexidade computacional, sendo classificados como tarefas leves, médias ou pesadas.

Considera-se um conjunto de tarefas associado ao cenário descrito, representado por T=$1, ..., n$, em que cada tarefa corresponde a um serviço ou uma demanda computacional que pode ser requisitada pelos veículos durante sua movimentação no ambiente urbano. Cada tarefa necessita de uma quantidade específica de recursos computacionais para ser processada e executada. Neste trabalho, são considerados como recursos a memória, o armazenamento, a capacidade de processamento e o tempo de execução.

Dessa forma, cada tarefa é caracterizada por um identificador único (id_task), pela quantidade de memória requerida (m_t), armazenamento necessário (s_t), capacidade de processamento demandada (p_t) e tempo de execução estimado (t_t), além da identificação do veículo solicitante ($id\_vehicle_t$) ao qual a tarefa pertence. Esse último atributo é fundamental para o mecanismo de priorização implementado, pois permite considerar o histórico de colaboração do veículo proprietário durante o processo de alocação da tarefa.

Os veículos são considerados os nós de borda do ambiente VEC. Eles se deslocam pelo cenário urbano e disponibilizam seus recursos computacionais para atender às solicitações de processamento de tarefas encaminhadas à rede. Dessa forma, considera-se um conjunto de veículos V, definido como $V = 1, ..., n$, em que cada veículo possui um identificador único ($id\_vehicle_v$), capacidade de memória (m_v), capacidade de armazenamento (s_v) e capacidade de processamento (p_v), utilizados para atender às demandas computacionais recebidas.

Os atributos relacionados aos recursos computacionais são gerados aleatoriamente para cada veículo, permitindo representar a heterogeneidade dos dispositivos embarcados presentes no cenário. Além desses recursos, os veículos possuem informações relacionadas ao seu histórico de participação na VEC, como o número de vezes em que ingressaram na rede ($entry_v$), considerando que os veículos podem se deslocar pelo ambiente, sair da área de cobertura de uma VEC e posteriormente ingressar em outra, e a quantidade de tarefas processadas ($task\_proc_v$), que representa o número de solicitações atendidas pelo respectivo nó de borda.

Os atributos $entry_v$ e $task\_proc_v$ são atualizados dinamicamente conforme a contribuição de cada nó de borda para a VEC. Essas informações são utilizadas pelo meca-

nismo desenvolvido tanto no processo de seleção do nó de borda responsável pelo processamento das tarefas quanto na definição da prioridade de atendimento das solicitações.

O agendamento das tarefas é realizado considerando o histórico de colaboração do veículo proprietário da solicitação. Dessa forma, por meio da identificação do veículo associado à tarefa ($id\_vehicle_t$), a contribuição desse usuário para a rede é analisada utilizando os atributos $entry_v$ e $task\_proc_v$. A partir dessas informações, é definida uma pontuação de colaboração (Cs_i), calculada com base no histórico de participação do nó de borda, conforme apresentado na Equação 4.1,

$$Cs_i = w_E \frac{E_i}{E_{\max}} + w_A \frac{A_i}{A_{\max}}, \quad w_E + w_A = 1 \tag{4.1}$$

onde E_i representa o número de entradas (participações) do veículo em um *cluster*, A_i corresponde ao número de tarefas efetivamente processadas pelo veículo, $E_{\max}$ e $A_{\max}$ representam, respectivamente, os valores máximos considerados para as entradas e os atendimentos, enquanto w_E e w_A correspondem aos pesos atribuídos a cada componente da pontuação de colaboração.

Formalmente, considera-se que um conjunto de N_t tarefas é encaminhado à RSU para processamento. Inicialmente, uma relação de ordenação $\prec$ é aplicada ao conjunto de tarefas, de modo que $t_i \prec t_j$ (ou seja, a tarefa t_i possui maior prioridade que a tarefa t_j) quando a condição apresentada na Equação 4.2 é satisfeita:

$$(Cs_i) > (Cs_j) \tag{4.2}$$

onde i e j representam índices genéricos das tarefas consideradas.

Após receber as tarefas ordenadas com base no compartilhamento dos usuários, a RSU executa o mecanismo de tomada de decisão responsável por selecionar o nó de borda mais adequado para realizar o processamento da tarefa. O mecanismo de seleção retorna o identificador do veículo escolhido ($id\_vehicle_v$). Em seguida, a RSU encaminha uma mensagem ao nó de borda selecionado contendo as informações referentes aos recursos necessários para execução da tarefa, incluindo a quantidade de memória (m_t), armazenamento (s_t) e capacidade de processamento (p_t) requerida.

Uma tarefa t_i é considerada viável para execução em um nó de borda v_j se, e somente se, os recursos computacionais disponíveis no nó forem suficientes para atender às demandas da tarefa, conforme definido pela seguinte condição:

$$M_j \geq m_i, \quad P_j \geq p_i, \quad S_j \geq s_i \tag{4.3}$$

O nó de borda avalia sua disponibilidade de recursos para processar a solicitação recebida e retorna uma resposta à RSU informando a possibilidade de execução da ta-

refa. Caso o processamento seja realizado com sucesso, os recursos utilizados durante a execução são temporariamente descontados da capacidade disponível do nó de borda pelo período correspondente ao tempo de execução da tarefa (t_t). Após a conclusão do processamento, os recursos computacionais utilizados são liberados e tornam-se novamente disponíveis para atender novas solicitações da VEC. Seja $\sigma : \mathcal{T} \rightarrow \mathcal{C}$ uma função de alocação responsável por mapear cada tarefa para um determinado nó de borda. O framework PRIOBILL busca determinar a solução ótima σ^* para o problema de otimização definido pela maximização conjunta do número de tarefas alocadas, da utilização dos recursos computacionais e da taxa de transferência do sistema, conforme apresentado na Equação 4.4:

$$\sigma^* = \arg\max_{\sigma} \left(N_{\text{alloc}}, ; R_{\text{total}}, ; \Theta\right) \quad \text{sujeito a:} \tag{4.4}$$

$$\mathbf{R}_{\sigma(t_i)} \geq \mathbf{r} * i, \qquad \forall, t_i \in \mathcal{T} \tag{4.5}$$

$$\sum *t_i :, \sigma(t_i) = v_j p_i \leq P_j, \qquad \forall, v_j \in \mathcal{C} \tag{4.6}$$

$$v_j \in \mathcal{C}(t), \qquad \forall, t_i \in \mathcal{T}, ; t \in [0, \tau_i] \tag{4.7}$$

onde $N_{\text{alloc}} = |t_i \in \mathcal{T} : \sigma(t_i)$ é viável$|$ representa a quantidade de tarefas alocadas com sucesso, $R_{\text{total}} = \sum_i (s_i + p_i + m_i)$ corresponde ao total de recursos computacionais utilizados pelas tarefas alocadas (Eq. 4.4) e Θ representa a taxa de transferência do sistema, definida como a quantidade de tarefas processadas por unidade de tempo.

A restrição (4.5) garante que os recursos disponíveis no nó de borda selecionado sejam suficientes para atender aos requisitos da tarefa. A restrição (4.6) limita a utilização dos recursos de processamento de cada nó de borda, evitando sua sobrecarga durante a execução das tarefas. Por fim, a restrição (4.7) assegura que o nó selecionado v_j permaneça dentro da área de cobertura da RSU durante todo o período de execução da tarefa, considerando a mobilidade dinâmica dos veículos que compõem o ambiente VEC.

Como mecanismo de incentivo ao compartilhamento de recursos computacionais, para cada tarefa t_i processada com sucesso, a quantidade de recursos utilizada durante sua execução é contabilizada na fatura do nó de borda responsável pelo processamento, representado por $v\_\sigma(t_i)$. O total de recursos computacionais faturados considerando todas as tarefas alocadas com sucesso é calculado conforme:

$$M_{\text{alloc}} = \sum_{i \in \mathcal{S}_{\surd}} m_i, \; P_{\text{alloc}} \; = \sum_{i \in \mathcal{S}_{\surd}} p_i, \; S_{\text{alloc}} = \sum_{i \in \mathcal{S}_{\surd}} s_i. \tag{4.8}$$

onde $\mathcal{S}_{\surd}$ representa o conjunto de tarefas processadas com sucesso. Esse mecanismo caracteriza um modelo de faturamento baseado em uso, seguindo o conceito de Infraes-

trutura como Serviço (IaaS, do inglês, *Infrastructure as a Service*), no qual os pagamentos são destinados aos veículos que disponibilizam seus recursos computacionais para a rede VEC.

4.3 CLUSTERIZAÇÃO

Para a formação do *cluster*, define-se um controlador responsável por implementar e gerenciar o ambiente VEC. Na arquitetura de representação da topologia viária, os nós de borda são modelados como vértices, enquanto os *links* de comunicação estabelecidos entre eles são representados como arestas. Por meio dos sinais WAVE (do inglês, *Wireless Access in Vehicular Environments*) e LTE (do inglês, *Long Term Evolution*) recebidos pelos nós de borda e pela RSU, as informações do cenário viário são coletadas e utilizadas para o gerenciamento da VEC. Além disso, a VEC recebe um conjunto completo de dados relacionados ao ambiente viário, permitindo o gerenciamento dinâmico do *cluster* em tempo real.

O algoritmo de agrupamento K-means é aplicado durante a etapa de inicialização da rede para formação dos *clusters*. Posteriormente, um mecanismo dinâmico de manutenção dos *clusters* é empregado para atualizar sua composição conforme a movimentação dos nós de borda no cenário.

Após a formação do *cluster*, a RSU recebe solicitações de processamento de tarefas seguindo uma distribuição de Poisson com taxa $\lambda = 1$. Em seguida, o mecanismo de alocação de tarefas é iniciado. A RSU, atuando como nó responsável pelo gerenciamento do *cluster*, executa o mecanismo de tomada de decisão para identificar o nó de borda mais adequado, dentre os participantes do *cluster*, para atender à solicitação recebida.

Após a seleção, a RSU encaminha uma mensagem ao nó de borda escolhido contendo as informações referentes à tarefa a ser processada. O nó de borda avalia sua disponibilidade de recursos computacionais e retorna uma resposta à RSU informando sua capacidade ou impossibilidade de atender à demanda solicitada.

4.4 INFRAESTRUTURA VEÍCULO-PARA-INFRAESTRUTURA (V2I)

A comunicação entre os membros do *cluster* é realizada por meio de mensagens *beacons*, transmitidas utilizando o mecanismo de *broadcast*. Após a formação dos *clusters*, a RSU envia periodicamente mensagens *beacons* para monitorar a presença dos veículos em sua área de cobertura. Caso a RSU envie um *beacon* e não receba uma resposta do veículo, considera-se que o nó de borda não está mais dentro do raio de comunicação da

Figura 14 – Exemplo de cenário de transmissão de RSU para nós de borda.



Fonte: Trecho de Manhattan - Gerado por IA.

RSU e, consequentemente, deixou a VEC.

Entretanto, devido à mobilidade dos veículos no cenário urbano, o veículo pode retornar posteriormente à mesma VEC ou ingressar em uma nova área de cobertura associada a outra RSU, dependendo da rota percorrida. Na Figura 14 apresenta-se um exemplo obtido a partir do Simulador de Mobilidade Urbana (SUMO), no qual uma RSU realiza a disseminação de uma mensagem *broadcast* para os nós de borda localizados dentro de seu raio de cobertura.

As RSUs atuam como elementos intermediários de comunicação entre os nós de borda e a infraestrutura de rede. Por meio da comunicação sem fio, as RSUs disseminam mensagens em tempo real para os veículos localizados dentro de sua área de cobertura. Essas mensagens podem representar diferentes tipos de eventos, como informações de localização, solicitações para formação de *clusters* ou requisições para alocação de tarefas computacionais.

O Algoritmo 1 apresenta as operações realizadas pela RSU durante o envio de mensagens *beacons*. Durante a execução da simulação (linha 1), a RSU transmite mensagens que podem desencadear três tipos de eventos: *i*) inundação, *ii*) clusterização ou *iii*) alocação de tarefas.

Quando o *beacon* enviado corresponde ao evento $FLOOD$ (linha 2), uma nova mensagem *beacon msg* é criada contendo informações da RSU, como o identificador r, a posição $\mathbf{p}_r$ e a origem da mensagem, definida como *RSU* (linhas 3, 4, 5 e 6). Posteriormente, a mensagem é criada e transmitida (linhas 7 e 8) utilizando *broadcast* para todos os veículos localizados dentro do alcance da RSU. Ao final desse processo, um novo evento de inundação é agendado (linha 9).

Caso o evento recebido seja do tipo $CLUSTER$ (linha 10), o algoritmo verifica inicialmente a existência de veículos $\mathcal{C}$ no cenário (linha 11). Caso existam veículos disponíveis, um novo evento de agrupamento é agendado (linha 12). Entretanto, quando a quantidade de veículos presentes no cenário é suficiente para a formação do *cluster* (linha 13), o processo de agrupamento é iniciado (linhas 14 e 15).

Se o processo de criação do *cluster* ocorrer pela primeira vez (linha 16), o algoritmo K-means é executado para realizar o agrupamento dos nós de borda (linha 17), e a variável "*init*"recebe o valor falso (linha 18), indicando que o processo inicial de formação do *cluster* foi concluído. Por outro lado, caso o *cluster* já tenha sido previamente criado (linha 19), o algoritmo realiza apenas sua atualização (linha 20), considerando a mobilidade dos veículos no cenário.

Em seguida, a variável responsável por contabilizar a quantidade de RSUs inicializadas ("ρ") é atualizada (linha 21), e um novo evento de agrupamento é programado (linha 22), permitindo a manutenção dinâmica da estrutura da VEC.

Quando o evento enviado corresponde à inicialização do processo de alocação de tarefas ($ALLOC$) (linha 23), o algoritmo verifica se existe um *cluster* previamente formado e disponível (linha 24). Caso a condição seja atendida, o processo de alocação de tarefas é iniciado (linha 25). Após a execução desse processo, um novo evento de alocação de tarefas é agendado (linha 26), mantendo o funcionamento contínuo do mecanismo de gerenciamento da VEC.

A mensagem é transmitida com o objetivo de alcançar o nó de borda selecionado para o processamento da tarefa. Ao receber a solicitação, o nó de borda verifica se possui recursos computacionais suficientes para atender à demanda recebida e, posteriormente, envia uma mensagem de resposta à RSU informando o resultado da verificação.

Ao receber a resposta do nó de borda, a RSU verifica se a solicitação foi classificada como "alocada"ou "negada". Caso a tarefa seja alocada com sucesso, a RSU contabiliza a alocação realizada e atualiza os recursos disponíveis do nó de borda, descontando temporariamente os recursos utilizados pela tarefa. Além disso, a RSU agenda uma nova mensagem associada ao término do tempo de execução da tarefa, permitindo que os recursos utilizados sejam liberados e retornem à disponibilidade do nó de borda.

Por outro lado, caso a resposta seja classificada como "negada", a RSU apenas

Algoritmo 1 – Inundação, Clusterização e Alocação de Tarefas na RSU

```
1  Enquanto a simulação estiver em execução faça
2      Se e = FLOOD então
3          msg ← nova BeaconMsg;
4          msg.rsuID ← r;
5          msg.pos ← p_r;
6          msg.type ← RSU;
7          populateWSM(msg);
8          sendDown(msg);
9          agendar evento para t + 1;
10     Se e = CLUSTER então
11         Se |C| < θ então
12             agendar evento para t + 5;
13         Senão
14             ready ← true;
15             atualizaVizinhos(C);
16             Se init ∧ (ρ mod 7 = 0) então
17                 RunKmeans();
18                 init ← false;
19             Senão Se ρ mod 7 = 0 então
20                 atualizaCluster(β);
21             ρ ← ρ + 1;
22             agendar evento para t + 1;
23     Se e = ALLOC então
24         Se ready = true então
25             iniciaProcessoAlocacao();
26         agendar evento para t + 1 + uniform(0.05, 0.1);
```

contabiliza a solicitação como uma tarefa não processada, indicando que o nó de borda selecionado não possuía recursos suficientes ou não estava disponível para atender à demanda.

4.5 GERENCIAMENTO DE TAREFAS

O processo de gerenciamento de tarefas representa um módulo fundamental da estrutura desenvolvida. Esse módulo é responsável por monitorar o estado dos *clusters*, a RSU em execução e as solicitações de tarefas encaminhadas para processamento na VEC. Inicialmente, um conjunto de tarefas é gerado de forma aleatória e encaminhado para processamento. Em seguida, o algoritmo realiza o escalonamento dessas tarefas, priorizando aquelas pertencentes aos usuários que apresentam maior contribuição para a rede. O algoritmo de escalonamento apresenta o funcionamento do mecanismo de priorização

aplicado durante o processo de ordenação das tarefas.

O algoritmo de escalonamento recebe como parâmetros a quantidade de tarefas (λ) que solicitam recursos computacionais, o conjunto responsável por armazenar as tarefas na fila de saída ($\mathcal{Q}$), o conjunto de veículos associados às tarefas ($\mathcal{V}$) e o conjunto de tarefas pertencentes a cada veículo ($\mathcal{T}$). Como resultado, o algoritmo retorna o conjunto de tarefas ordenadas de acordo com a prioridade definida ($\mathcal{O}$).

O mecanismo inicia a iteração considerando a quantidade de tarefas que devem ser ordenadas (linha 1) e verifica se ainda existem tarefas disponíveis para processamento (linha 2). Caso existam tarefas pendentes, a primeira tarefa i é removida da fila de saída ($\mathcal{Q}$) (linha 3), adicionada à lista temporária de tarefas selecionadas para o processo de ordenação ($\mathcal{S}$) (linha 4) e removida da fila original ($\mathcal{Q}$) (linha 5).

Após a seleção das tarefas, o algoritmo executa a função de comparação (linha 6), responsável por consultar o histórico de colaboração do veículo associado à tarefa, verificando a quantidade de serviços realizados por esse veículo. Essas informações são utilizadas para definir a prioridade de atendimento das tarefas na ordem de processamento.

Ao final do processo de comparação, as tarefas selecionadas são inseridas na lista de tarefas ordenadas ($\mathcal{O}$) (linhas 7 e 8), e a lista temporária é reinicializada para permitir a execução do próximo ciclo de agendamento (linha 9). Por fim, o conjunto de tarefas ordenadas por prioridade ($\mathcal{O}$) é retornado ao processo de alocação de tarefas (linha 10).

Algoritmo 2 – Escalonamento e ordenação de tarefas na RSU

Entrada: λ, $\mathcal{Q}$, $\mathcal{V}$, $\mathcal{T}$
Saída: $\mathcal{O}$

```
1  Para k ← 1 to λ faça
2      Se Q não está vazia então
3          i ← front(Q);
4          inserir i em S;
5          remover o primeiro elemento de Q;
6  ordenar S utilizando CompareTasks;
7  Para i ∈ S faça
8      adicionar i em O;
9  limpar S;
10 Retorna O
```

O Algoritmo 3 apresenta o processo de gerenciamento de tarefas, no qual o Algoritmo 2 é utilizado para realizar o escalonamento das tarefas, enquanto o framework PRIOBILL é responsável por selecionar o nó de borda mais adequado para executar o processamento da tarefa.

O processo inicia com o mapeamento dos veículos presentes no cenário $\mathcal{C}$ e daqueles pertencentes à área de cobertura da RSU r, responsável pelo atendimento da solicitação recebida. Esses veículos são armazenados em uma matriz de nós de borda $\mathcal{V}$ (linhas 1, 2 e 3). Em seguida, o algoritmo verifica a existência de tarefas pendentes na fila de tarefas (linha 4) e obtém a quantidade de tarefas recebidas, representada por Λ (linha 5).

Posteriormente, o algoritmo verifica se existem veículos disponíveis no *cluster* $\mathcal{V}$ (linha 6). Caso existam, é avaliada a taxa de chegada de tarefas Λ (linha 7). Quando o número de tarefas recebidas é maior que 1, o algoritmo de escalonamento dinâmico (Algoritmo 2) é executado para realizar a ordenação das tarefas conforme a prioridade definida (linha 8).

Caso apenas uma tarefa seja recebida (linha 9), ela é retirada da fila de saída de tarefas $\mathcal{Q}$ e inserida diretamente na lista de tarefas ordenadas $\mathcal{O}$ (linha 10), sendo posteriormente removida da fila de tarefas pendentes (linha 11). Após essa etapa, o algoritmo percorre as tarefas recebidas (linha 12), respeitando a ordem estabelecida pela lista de prioridade (linhas 13 e 14), e obtém as informações referentes à tarefa associada ao nó de borda $\mathcal{T}[i]$ (linha 15).

Com todas as informações necessárias obtidas, o framework PRIOBILL é executado para realizar o processo de seleção do nó de borda responsável pelo processamento da tarefa, retornando o nó selecionado (linha 16). Em seguida, uma mensagem de solicitação é criada (linha 17), contendo as informações da tarefa, como identificador, quantidade de armazenamento, capacidade de processamento requerida, memória necessária, tempo de execução, nó de borda destinatário selecionado, RSU responsável pelo envio da solicitação e o tipo de mensagem.

Por fim, a mensagem é transmitida ao nó de borda selecionado (linha 18), e a quantidade de tarefas encaminhadas para processamento é contabilizada por meio da variável N_{sent} (linha 19).

Algoritmo 3 – Processo de Alocação de Tarefas na RSU

```
1  Para v ∈ C faça
2      Se r = v.rsuId então
3          adicionar v.id em V;
4  Se Q não está vazia então
5      λ ← front(Λ);
6      Se |V| > 0 então
7          Se λ > 1 então
8              escalation(λ);
9          Senão Se λ = 1 então
10             adicionar front(Q) em O;
11             remover primeiro elemento de Q;
12         Para k ← 1 to λ faça
13             i ← front(O);
14             remover primeiro elemento de O;
15             t_i ← T[i];
16             j* ← selectEdgeNode();
17             populateWSM(msg);
18             sendDelayedDown(msg);
19             N_sent ← N_sent + 1;
```

Após o envio da solicitação, o nó de borda verifica sua capacidade de processamento para atender à tarefa recebida (Algoritmo 5) e retorna uma mensagem de resposta à RSU. Ao receber essa resposta, a RSU realiza o processamento das informações de acordo com a abstração apresentada no Algoritmo 4.

O Algoritmo 4 é responsável por processar as respostas encaminhadas pelos nós de borda. Como entrada, o algoritmo recebe a mensagem de resposta e o vetor contendo as informações dos nós de borda; como saída, realiza a atualização dos recursos utilizados e das variáveis de controle do sistema. Inicialmente, verifica-se se a mensagem recebida corresponde a uma resposta do tipo "alocada"(linha 1). Caso positivo, uma tarefa processada com sucesso é contabilizada em N_{alloc} (linha 2), e o nó de borda responsável pelo processamento da tarefa é identificado (linha 3). Em seguida, o veículo correspondente é localizado (linha 4), e os recursos de memória M_j, processamento P_j e armazenamento S_j utilizados durante a execução da tarefa são decrementados dos recursos disponíveis do nó de borda (linhas 5, 6 e 7, respectivamente).

Posteriormente, é realizado o processo de faturamento, no qual os recursos compartilhados pelo nó de borda são contabilizados nas variáveis M_{alloc}, P_{alloc} e S_{alloc} (linhas

8, 9 e 10, respectivamente). Além disso, a soma total dos recursos utilizados no processamento das tarefas é armazenada em R_{alloc} (linha 11). Após a atualização dos recursos, uma nova mensagem é criada e agendada (linhas 12 e 13) com um atraso correspondente ao tempo de execução da tarefa. Ao final desse período, os recursos previamente alocados são liberados e retornam ao conjunto de recursos disponíveis do nó de borda.

Caso a mensagem de resposta seja do tipo "negada"(linha 14), significa que o nó de borda selecionado não possui recursos suficientes para atender à solicitação. Nesse caso, apenas uma tarefa não processada é contabilizada em N_{den} (linha 15).

Algoritmo 4 – Processamento da resposta de alocação na RSU

Entrada: msg, V
Saída: Recursos utilizados e contadores atualizados

```
1  Se msg.message = alocada então
2      N_alloc ← N_alloc + 1;
3      Se msg.dest ∈ V então
4          v ← V[msg.dest];
5          M_j ← M_j − m_i;
6          P_j ← P_j − p_i;
7          S_j ← S_j − s_i;
8      M_alloc ← Σ_{i∈S} m_i;
9      P_alloc ← Σ_{i∈S} p_i;
10     S_alloc ← Σ_{i∈S} s_i;
11     R_alloc ← R_alloc + r_i;
12     criar mensagem delay_time;
13     agendar evento em t + delay;
14 Senão Se msg.message = negada então
15     N_den ← N_den + 1;
```

4.6 GERENCIAMENTO NO NÓ DE BORDA

O módulo do nó de borda é responsável pela comunicação entre o veículo e a RSU, realizada por meio de mensagens *beacons*. Durante o processo de alocação de tarefas, após a RSU selecionar um veículo por meio do algoritmo de tomada de decisão, uma mensagem *beacon* é enviada ao nó de borda para que este verifique a disponibilidade de seus recursos computacionais e informe se possui capacidade para processar a tarefa solicitada.

O Algoritmo 5 apresenta uma abstração do gerenciamento realizado pelo nó de borda. Como parâmetros de entrada, o algoritmo recebe a mensagem enviada pela RSU (msg), os dados do veículo (V) e a identificação do próprio nó de borda (j). Como saída,

retorna uma resposta indicando se a tarefa pode ou não ser alocada ao veículo.

Inicialmente, o algoritmo verifica se a mensagem recebida é destinada ao veículo correspondente (linha 1). Em seguida, verifica o tipo da mensagem enviada pela RSU para identificar se ela corresponde a uma solicitação de processamento de tarefa (linha 2). Caso essa condição seja satisfeita, os dados referentes à capacidade computacional do nó de borda são armazenados no vetor v (linha 3).

Na sequência, o algoritmo obtém os recursos disponíveis do nó de borda, correspondentes à memória (M_j), ao processamento (P_j) e ao armazenamento (S_j) (linha 4), bem como os recursos requeridos pela tarefa, representados pela memória (m_i), processamento (p_i) e armazenamento (s_i) (linha 5). Essas informações são utilizadas para verificar se o nó de borda dispõe de recursos suficientes para executar a tarefa (linha 6).

Caso os recursos sejam suficientes, o nó de borda confirma a possibilidade de processamento da solicitação, criando uma mensagem contendo a identificação da tarefa, a RSU responsável pela solicitação, o tipo da mensagem, o identificador do nó de borda remetente, a quantidade de memória, processamento e armazenamento requerida, o tempo de utilização dos recursos e a indicação de que a tarefa foi "alocada" (linha 7). Em seguida, o envio dessa mensagem é agendado (linha 8).

Caso o nó de borda não possua recursos suficientes para atender à solicitação, o algoritmo cria uma mensagem indicando que a tarefa foi "negada" (linha 10) e agenda sua transmissão para a RSU (linha 11).

Algoritmo 5 – Gerenciamento de tarefas no nó de borda

Entrada: msg, $\mathcal{V}$, j
Saída: Resposta de alocação

```
1  Se j = msg.dest então
2      Se msg.type = TASK então
3          v_j ← V[j];
4          R_j = (M_j, P_j, S_j);
5          r_i = (m_i, p_i, s_i);
6          Se R_j ≥ r_i então
7              criar mensagem "alocada";
8              agendar resposta no tempo t;
9          Senão
10             criar mensagem "negada";
11             agendar resposta no tempo t;
```

4.7 *BILLING*

Neste contexto, é implementado um sistema de *billing* integrado ao processo otimizado de alocação de tarefas computacionais em VEC. O objetivo é maximizar o aproveitamento da infraestrutura disponível, otimizando os custos de implantação e manutenção e proporcionando uma arquitetura sustentável e eficiente, capaz de minimizar desperdícios e contribuir para o retorno do investimento (OWOADE et al., 2023).

Neste trabalho, considera-se um modelo de cobrança baseado no uso, em que os usuários pagam exclusivamente pelos recursos computacionais efetivamente consumidos. Esse modelo é especialmente adequado para ambientes flexíveis e dinâmicos, como aqueles encontrados em VEC (OWOADE et al., 2023). Além disso, direcionar os pagamentos aos usuários que disponibilizam seus recursos computacionais para a rede constitui um importante mecanismo de incentivo ao compartilhamento de recursos ociosos.

O sistema de *billing* implementado utiliza escalonamento automático, disponibilizando recursos aos usuários por meio de um mecanismo eficiente de alocação de tarefas que explora os recursos computacionais dos nós de borda presentes na VEC. Dessa forma, as tarefas são processadas diretamente na borda da rede, reduzindo os custos associados ao uso de infraestruturas centralizadas. Com o escalonamento automático, os pagamentos são realizados sob demanda, em tempo real e de acordo com a utilização efetiva dos recursos, eliminando a necessidade de manter capacidade computacional superdimensionada (OWOADE et al., 2023).

A implementação, manutenção e operação de um sistema de *billing* representam atividades complexas e de elevado custo (PATTUROSE et al., 2024). Entretanto, considerando que as VECs já se encontram inseridas em ambientes urbanos, a integração de um sistema de *billing* a um mecanismo eficiente de gerenciamento de recursos pode tornar esse ambiente mais atrativo para seus usuários. Nesse contexto, os serviços disponibilizados pela VEC podem ser estruturados, oferecidos e comercializados segundo o modelo de Infraestrutura como Serviço (IaaS) (XIE et al., 2022). Em VEC, os recursos computacionais são compartilhados pelos próprios veículos, que atuam como nós de borda. Essa característica permite que o processamento seja realizado localmente, proporcionando menor latência, comunicação mais eficiente e redução dos custos operacionais quando comparado aos modelos tradicionais de computação centralizada (MENEGUETTE et al., 2021).

A implementação de um sistema de *billing* em uma VEC tem como finalidade incentivar os usuários presentes na área de cobertura da rede a compartilharem seus recursos computacionais ociosos, contribuindo para minimizar a escassez de recursos disponível no ambiente. Para isso, o modelo de *billing* implementado baseia-se nas Equações 4.9, 4.10 e 4.11, que representam, respectivamente, o total de recursos de memória, processamento e armazenamento compartilhados pelos nós de borda durante o processamento das tarefas.

$$M_{\text{alloc}} = \sum_{i \in S} m_i \tag{4.9}$$

$$P_{\text{alloc}} = \sum_{i \in S} p_i \tag{4.10}$$

$$S_{\text{alloc}} = \sum_{i \in S} s_i \tag{4.11}$$

onde m_i é a memória necessária para a tarefa, p_i é o processamento necessário para a tarefa e s_i é o armazenamento necessário para a tarefa.

4.8 MECANISMO DE ALOCAÇÃO DE TAREFAS PRIOBILL: BASEADO EM LOA

Neste *framework*, foi desenvolvido um mecanismo de tomada de decisão para alocação de tarefas em um ambiente de Computação de Borda Veicular com comunicação V2I, incorporando priorização de tarefas baseada no compartilhamento de recursos e um sistema de *billing*. O PRIOBILL foi desenvolvido com base no algoritmo bioinspirado Algoritmo de Otimização do Leão (LOA), cuja inspiração advém dos comportamentos sociais e das estratégias de caça observadas em leões. Esses animais apresentam elevado grau de cooperação tanto nas atividades sociais quanto durante o processo de caça. Nesse contexto, a colaboração entre os membros do grupo é fundamental, pois a estratégia de ataque considera o melhor posicionamento de cada indivíduo em relação à presa, aumentando as chances de sucesso da captura.

A organização social dos leões também desempenha um papel importante no mecanismo de tomada de decisão. Os bandos são tradicionalmente divididos em dois grupos: residentes e nômades. O grupo de residentes é, em geral, composto por cinco fêmeas, um ou dois machos e os filhotes das fêmeas. O segundo grupo é formado pelos nômades, que, diferentemente dos residentes, apresentam comportamento predominantemente solitário e deslocam-se continuamente em busca de novos territórios. Com base nessa organização, o PRIOBILL estabelece uma analogia com o comportamento descrito pelo LOA. Assim, uma parcela L dos nós de borda é utilizada para representar machos e fêmeas nômades, enquanto a parcela restante representa machos e fêmeas residentes. Durante o processo de caça, as fêmeas possuem prioridade na captura da presa. De forma análoga, no algoritmo desenvolvido, a prioridade de atendimento das tarefas é determinada considerando-se o número de vezes que um nó de borda compartilhou seus recursos com a VEC e o número de solicitações que processou ao longo de sua participação na rede.

Após a etapa de priorização das tarefas, o mecanismo emprega o comportamento social do LOA para construir a população inicial no espaço de busca, utilizando os recursos

computacionais disponibilizados pelos nós de borda presentes na VEC. Nessa analogia, o conjunto de nós de borda corresponde a um bando de leões, enquanto cada nó de borda possui uma prioridade associada ao seu histórico de colaboração com a rede. Durante cada iteração do algoritmo, é atribuída uma posição a cada nó de borda em relação à tarefa a ser processada, representada pela presa. Em seguida, seguindo a estratégia de aproximação empregada pelo LOA durante o processo de caça, o nó de borda desloca-se em direção à tarefa com o objetivo de cercá-la e capturá-la, representando o processo de seleção do recurso computacional mais adequado para realizar o processamento.

Dessa forma, o funcionamento do LOA no contexto do PRIOBILL é composto pelas seguintes etapas:

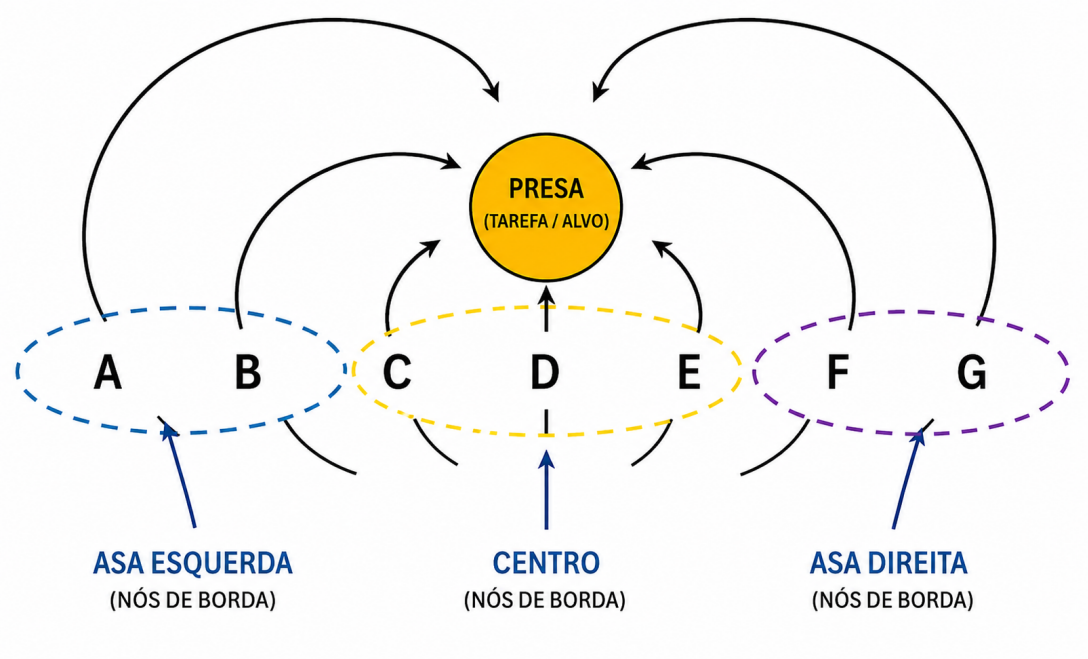
- Inicialização;
- Caça;
- Busca por um local seguro;
- Vagância;
- Defesa;
- Migração;
- Equilíbrio populacional.

Inicialização. Nesta etapa, cada "leão" representa um nó de borda da VEC. Cada nó de borda corresponde a uma solução candidata do problema de otimização. Considerando um problema com dimensão N_{var}, um nó de borda é representado por $Leao = [x_1, x_2, x_3, \ldots, x_{N_{var}}]$. Uma porcentagem $\%N$ das soluções é selecionada aleatoriamente para compor o grupo de nós de borda nômades, enquanto as soluções restantes são distribuídas aleatoriamente entre P bandos. Cada solução recebe um gênero específico, que permanece inalterado durante todo o processo de otimização. Para essa distribuição, define-se uma porcentagem $\%S$, variando entre 75% e 90%, para representar as fêmeas de cada bando, sendo os demais indivíduos classificados como machos. Durante o processo de busca, cada nó de borda registra as posições visitadas. Com base nesse histórico, é estabelecido o território correspondente a cada bando (YAZDANI; JOLAI, 2016).

Caça. Nesta etapa, as fêmeas de cada bando são responsáveis pela busca da presa que garantirá a alimentação do grupo. Para isso, utilizam estratégias cooperativas de perseguição e cercamento, aumentando a probabilidade de captura. Os machos também participam do processo de caça, desempenhando funções complementares durante a perseguição. Assim como no LOA, os nós de borda são distribuídos em diferentes funções de ataque e perseguição, conforme ilustrado na Figura 15 (STANDER, 1992).

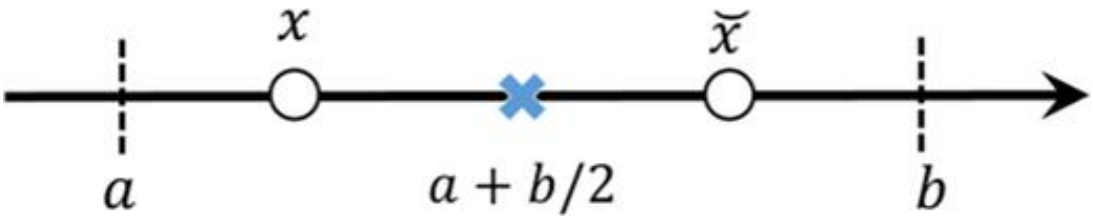
As funções de caça são organizadas em três grupos: Asa Esquerda, Centro e Asa Direita. Durante esse processo, os nós de borda classificados como fêmeas atualizam suas

Figura 15 – Leão caça.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

Figura 16 – Princípio da Aprendizagem Baseada em Oposição.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

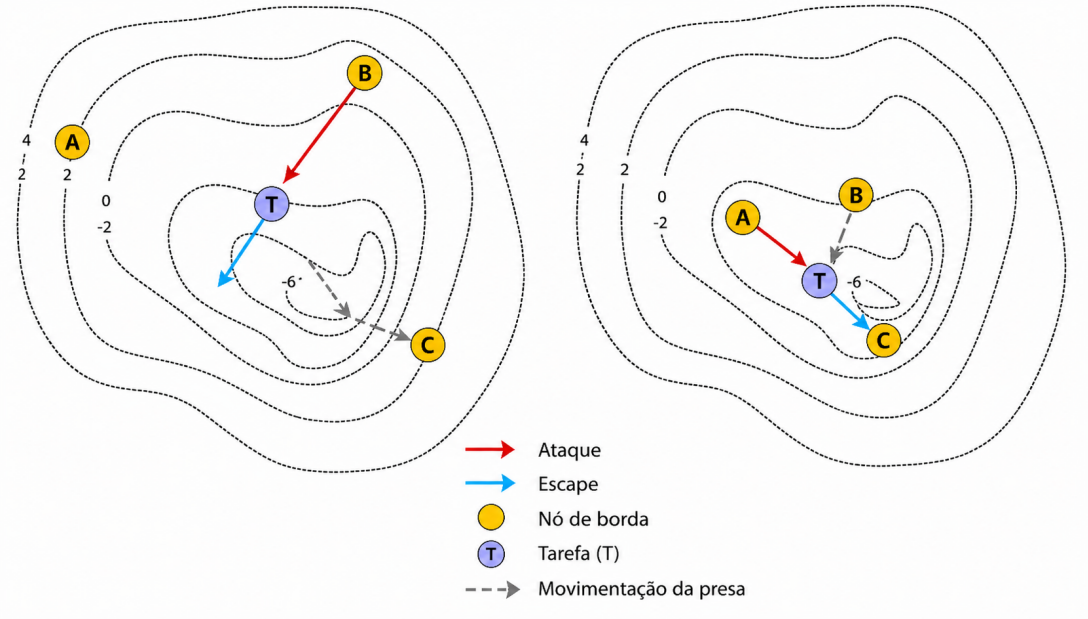
posições utilizando como referência sua própria posição e as posições dos demais nós de borda pertencentes ao *cluster*. Ao longo das iterações de caça, parte desses nós realiza o cercamento e o ataque à presa pelo lado oposto. Para isso, emprega-se o conceito de Aprendizagem Baseada em Oposição (*Opposition-Based Learning* – OBL) (TIZHOOSH, 2005).

Seja $X(x_1, x_2, x_3, \ldots, x_{N_{var}})$ um ponto em um espaço de busca de dimensão N_{var}, em que $x_1, x_2, x_3, \ldots, x_{N_{var}}$ são números reais e $x_i \in [a_i, b_i]$, para $i = 1, 2, 3, \ldots, N_{var}$. O ponto oposto de X é definido por $\tilde{X}(\tilde{x}_1, \tilde{x}2, \tilde{x}3, \ldots, \tilde{x}Nvar)$, em que $x_i = a_i + b_i x_i, i = 1, 2, \ldots, Nvar$.

Na Figura 16 apresenta-se o princípio da Aprendizagem Baseada em Oposição.

Com base nessas definições, os nós de borda são divididos em três grupos. O grupo do Centro é composto pelos nós de borda que apresentam a melhor aptidão, enquanto os demais são distribuídos entre as Alas Direita e Esquerda. A presa (TAREFA) representa o alvo do processo de otimização realizado pelos nós de borda. Durante a etapa de caça,

Figura 17 – Escape.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

os nós de borda da VEC são selecionados e cada um realiza um movimento em direção à tarefa. Caso um nó de borda obtenha uma melhoria em sua aptidão em relação à tarefa, considera-se que a presa realiza um movimento de fuga, fazendo com que uma nova posição da tarefa seja determinada de acordo com a Equação 4.12 (YAZDANI; JOLAI, 2016).

$$TAREFA' = TAREFA + \text{rand}(0,1) \times PI \times (TAREFA - NO) \tag{4.12}$$

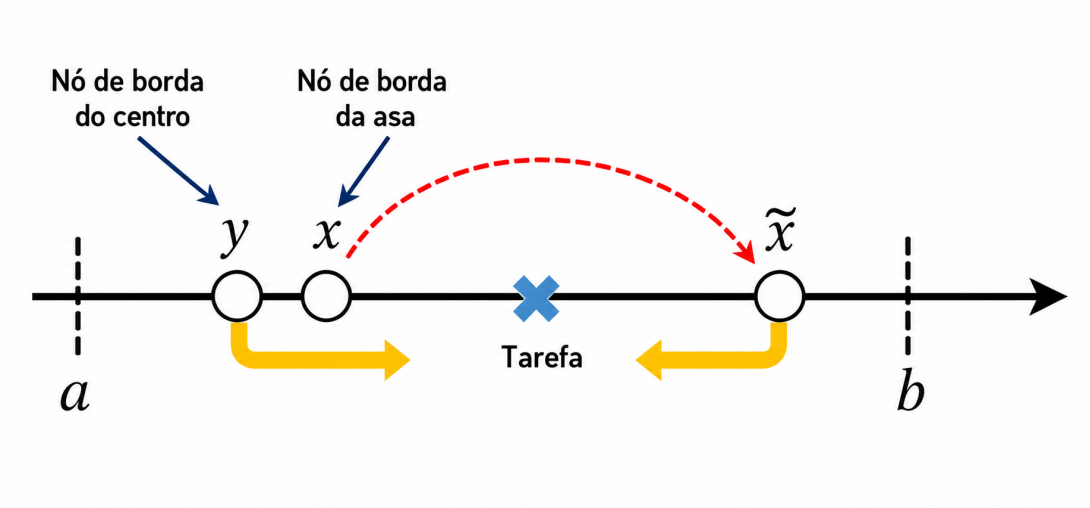
onde $TAREFA$ representa a posição atual da presa, NO corresponde à nova posição do nó de borda que realiza o ataque à presa, e PI representa a porcentagem de melhoria no *fitness* do nó de borda, conforme ilustrado na Figura 17.

Para cercar a presa, os nós de borda caçadores atualizam seus movimentos com base na Equação 4.13. Nessa etapa, as novas posições dos nós de borda pertencentes às alas esquerda e direita são calculadas da seguinte forma:

$$NO' = \begin{cases} \text{rand}((2 \times TAREFA - NO), TAREFA), & (2 \times TAREFA - NO) < TAREFA \\ \text{rand}(TAREFA, (2 \times TAREFA - NO)), & (2 \times TAREFA - NO) > TAREFA \end{cases} \tag{4.13}$$

onde $TAREFA$ representa a posição atual da tarefa, NO corresponde à posição atual do nó de borda e NO' indica sua nova posição. Já as posições dos nós de borda pertencentes ao Centro são atualizadas de acordo com a Equação 4.14.

Figura 18 – Movimento caça.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

$$NO' = \begin{cases} \text{rand}(NO, TAREFA), & NO < TAREFA \\ \text{rand}(TAREFA, NO), \end{cases} \tag{4.14}$$

Na Figura 18 ilustra-se o movimento de caça realizado por um nó de borda pertencente ao Centro e por um nó de borda localizado em uma das Alas durante o processo de otimização.

O processo de caça do LOA apresenta como uma de suas principais vantagens a utilização de uma vizinhança circular ao redor da $tarefa$ durante a busca pela melhor solução. Essa estratégia permite que os nós de borda caçadores cerquem a $tarefa$ a partir de diferentes direções, aumentando a capacidade de exploração do espaço de busca. Além disso, favorece a evasão de ótimos locais, contribuindo para a obtenção de soluções de melhor qualidade (YAZDANI; JOLAI, 2016).

Busca por um lugar seguro. Nem todos os nós de borda fêmeas participam diretamente da caça. Algumas permanecem posicionadas em regiões do território e, embora não atuem na captura da $tarefa$, suas posições são utilizadas pelo LOA para aprimorar as soluções encontradas. Assim, a nova posição de um nó de borda fêmea é atualizada de acordo com a Equação 4.15 (YAZDANI; JOLAI, 2016):

$$\begin{aligned} NOF' = NOF + 2D \cdot \text{rand}(0,1)\{R_1\} + U(-1,1) \cdot \tan(\theta) \cdot D\{R_2\} \\ \text{s.t.} \quad \{R_1\} \cdot \{R_2\} = 0, \qquad \|\{R_2\}\| = 1 \end{aligned} \tag{4.15}$$

onde NOF representa a posição atual do nó de borda fêmea, D corresponde à distância entre a posição de NOF e o ponto selecionado por meio do torneio dentro do território do

bando. O vetor $R1$ possui como origem a posição anterior de NOF e aponta em direção à posição selecionada, enquanto $R2$ é perpendicular a $R1$ (YAZDANI; JOLAI, 2016).

A estratégia de torneio adotada determina o sucesso de um nó de borda com base na melhoria de sua posição em relação à posição anterior. Assim, para o grupo P, o sucesso do nó de borda i na iteração t é definido de acordo com a Equação 4.16:

$$S(i,t,P) = \begin{cases} 1, & Best_{i,P}^{t} < Best_{i,P}^{t-1}, \\ 0, & Best_{i,P}^{t} = Best_{i,P}^{t-1}. \end{cases} \tag{4.16}$$

onde $Best$ representa a melhor posição alcançada pelo nó de borda i na iteração t.

A quantidade de sucessos obtidos desempenha um papel importante na definição do tamanho do torneio. Um número elevado de sucessos indica que os nós de borda estão se deslocando em direção a uma região próxima da solução ótima, enquanto um número reduzido de sucessos sugere que eles estão oscilando ao redor da melhor solução encontrada, sem obter melhorias significativas (YAZDANI; JOLAI, 2016). Esse indicador é utilizado para ajustar dinamicamente o tamanho do torneio, conforme a Equação 4.17, sendo atualizado a cada iteração. Quando o valor de sucesso diminui, o tamanho do torneio é aumentado, promovendo maior diversidade na busca por soluções (YAZDANI; JOLAI, 2016).

$$K_j(s) = \sum_{i=1}^{n} S(i,t,P), \qquad j \in \{1,\ldots,P\} \tag{4.17}$$

onde n representa a quantidade de nós de borda no bando e $K_j(s)$ corresponde ao número de nós de borda que obtiveram melhoria em seu *fitness* na última iteração.

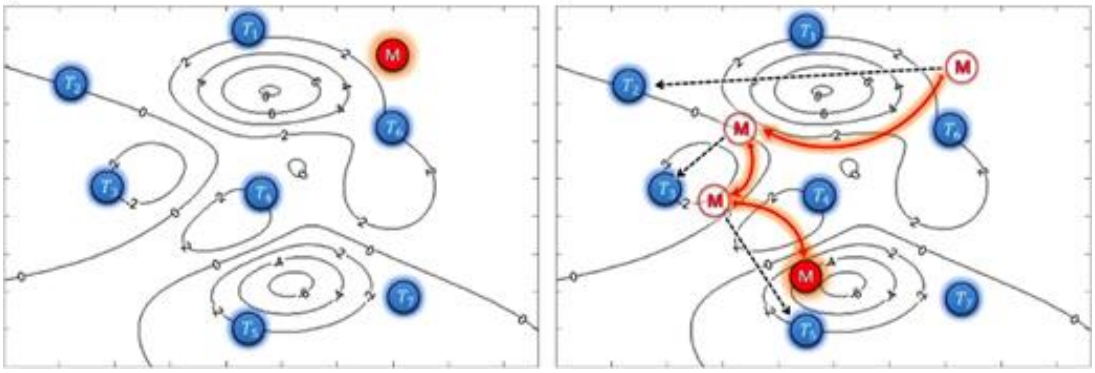
Vagando

Os nós de borda machos percorrem aleatoriamente uma região correspondente a %R do território. Caso um nó de borda macho encontre, durante esse deslocamento, uma posição com melhor *fitness* do que a sua posição atual, essa nova posição passa a ser considerada sua melhor solução visitada. Esse comportamento contribui para que o LOA explore regiões promissoras do espaço de busca e refine as soluções encontradas (YAZDANI; JOLAI, 2016). Na Figura 19 ilustra-se esse processo, no qual o nó de borda macho se desloca em direção à área selecionada por uma distância de x unidades, sendo x um valor aleatório com distribuição uniforme, conforme definido pela Equação 4.18.

$$x \sim \mathcal{U}(0, 2d) \tag{4.18}$$

em que d é a distância entre a posição do nó de borda macho e a área selecionada do território.

Figura 19 – Vagar.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

Durante o processo de vagar, os nós de borda nômades também desempenham um papel importante na exploração do espaço de busca. Seu comportamento de deslocamento contribui para evitar que o LOA fique preso em ótimos locais, favorecendo a descoberta de novas regiões promissoras. A atualização da posição dos nós de borda nômades é realizada de acordo com a Equação 4.19 (YAZDANI; JOLAI, 2016):

$$NOM'_{ij} = \begin{cases} NOM_{ij}, & \text{if } rand_j > p_{r_i}, \\ RAND_j, & \text{otherwise.} \end{cases} \tag{4.19}$$

onde NOM_i representa a posição atual do i-ésimo nó de borda nômade, j corresponde à dimensão analisada, $rand_j$ é um número aleatório no intervalo [0,1], $RAND$ é um vetor aleatório definido no espaço de busca e pr_i representa a probabilidade associada a cada nó de borda, calculada conforme a Equação 4.20.

$$p_{r_i} = 0.1 + \min\left(0.5,\ \frac{(\text{Nomad}_i - \text{Best}_{\text{nomad}})}{\text{Best}_{\text{nomad}}}\right) \quad i = 1, 2, \ldots, \text{number of nomad lions} \tag{4.20}$$

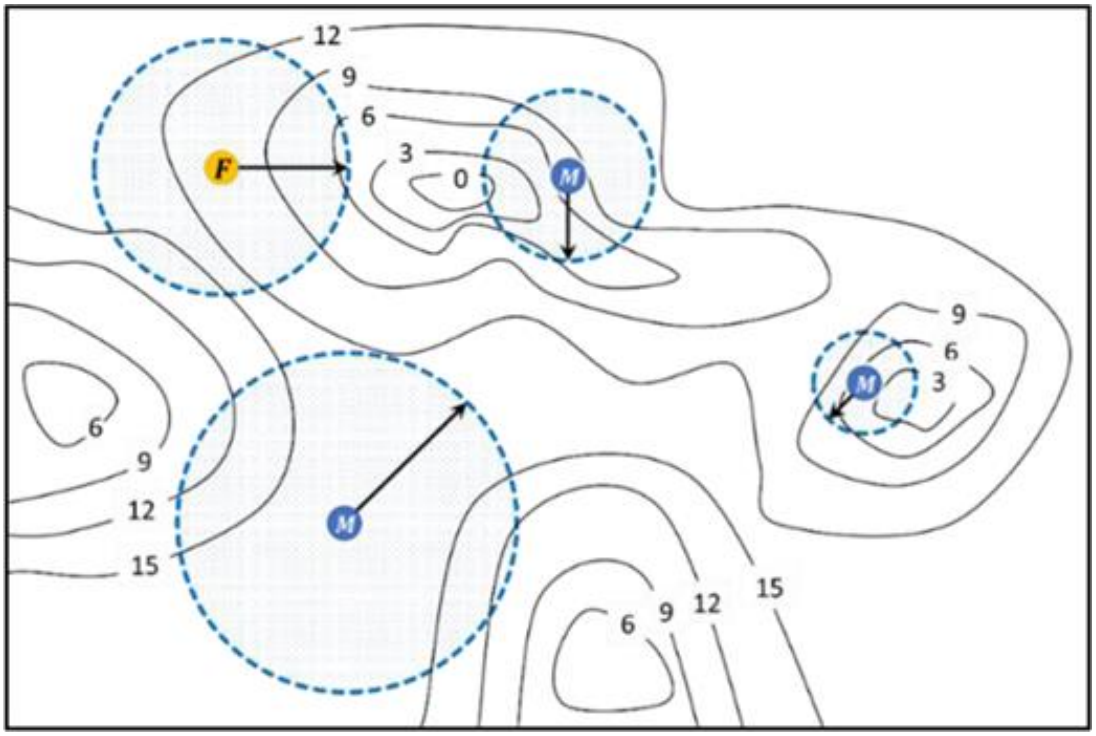
onde $nomad_i$ representa o custo associado à posição atual do nó de borda nômade i e $Best_{nomad}$ corresponde ao melhor custo encontrado entre os nós de borda nômades.

Nas Figuras 20 e 21 apresentam-se o processo de geração de uma nova posição pelos nós de borda nômades. Esse comportamento de movimentação permite que os nós de borda nômades com pior desempenho tenham maior probabilidade de abandonar regiões inadequadas do espaço de busca, contribuindo para a exploração de novas soluções (YAZDANI; JOLAI, 2016).

Defesa.

Em um bando, os nós de borda machos, ao atingirem a maturidade, apresentam um comportamento mais agressivo e competem com outros nós de borda machos per-

Figura 20 – Nomade 1.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

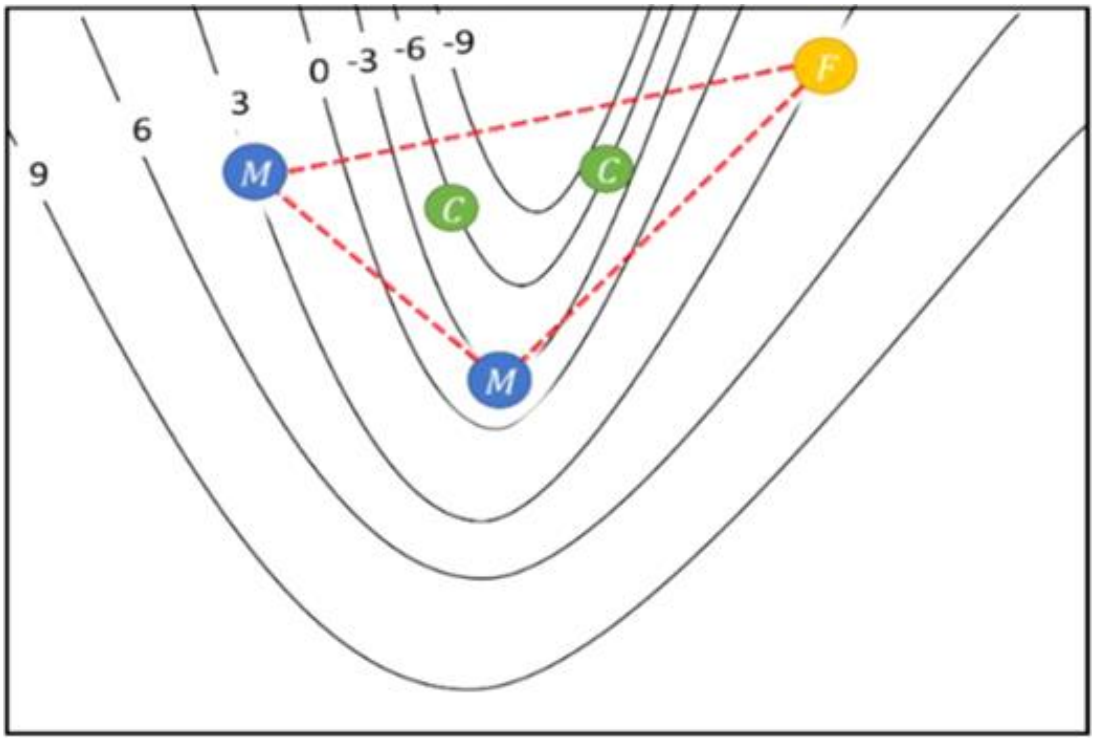
tencentes ao mesmo bando. Os nós de borda derrotados deixam de pertencer ao bando e passam a ser classificados como nômades. Entretanto, caso um nó de borda nômade possua capacidade suficiente para superar os machos residentes de um bando, o nó de borda residente derrotado é removido e torna-se um nômade.

No LOA, o processo de defesa é dividido em duas etapas principais, conforme apresentado na Figura 22: a defesa contra novos nós de borda residentes maduros e a defesa contra nós de borda nômades. Esse processo contribui para que o LOA mantenha os nós de borda machos com melhores soluções como integrantes do mecanismo de otimização.

Migração. Importante processo para a busca por diversidade e troca de informações, a migração simula a mudança de estilo de vida e o comportamento migratório dos leões na natureza. Em cada bando, alguns nós de borda fêmeas são selecionados aleatoriamente e migram para o grupo de nômades. Os novos nós de borda fêmeas nômades e os nós de borda fêmeas existentes são classificados com base no seu *fitness*. Posteriormente, os melhores nós de borda fêmeas identificados são selecionados aleatoriamente e distribuídos entre os bandos para ocupar as vagas deixadas pelas fêmeas que migraram. Esse processo permite aumentar a diversidade da população e manter a qualidade das soluções encontradas pelos nós de borda.

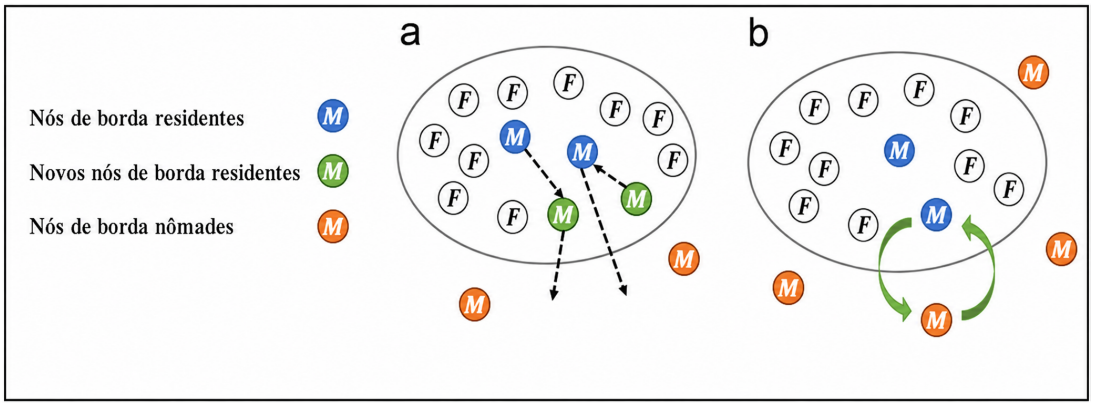
Equilíbrio populacional. O equilíbrio da população de nós de borda a cada iteração faz parte do mecanismo de controle do LOA. O número de nós de borda "vivos"é

Figura 21 – Nomade 2.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

Figura 22 – Defesa.

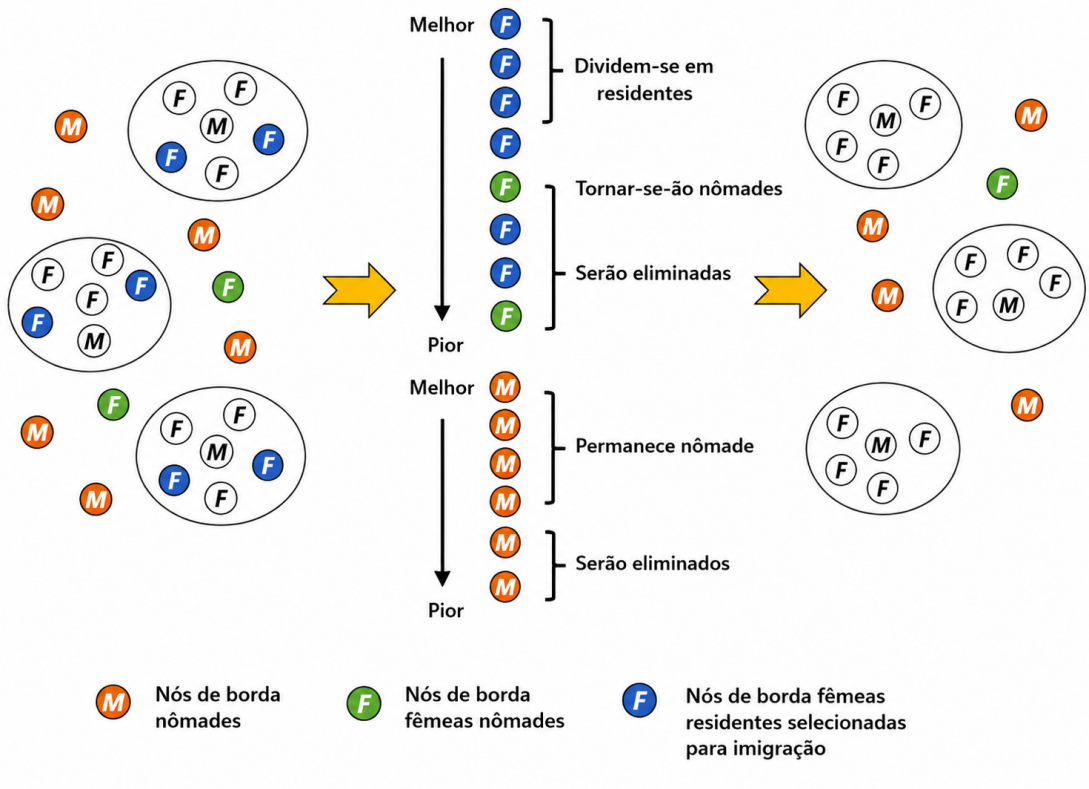


Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

definido respeitando a quantidade de membros estabelecida para cada bando. Da mesma forma, a quantidade de nós de borda nômades também é mantida de acordo com os parâmetros definidos. Durante esse processo, os nós de borda com pior aptidão são removidos da população pelo bando, conforme ilustrado na Figura 23, que apresenta os processos de migração e equilíbrio populacional dos nós de borda.

Baseado neste mecanismo, foi implementado o módulo de seleção de tarefas do PRIOBILL. Na Figura 24 apresenta-se o processo de ataque do módulo PRIOBILL. Nesse processo, os nós de borda são considerados os leões e realizam movimentos no espaço de busca para determinar qual nó de borda irá capturar a tarefa (presa). O posicionamento

Figura 23 – Representação dos processos de migração e equilíbrio populacional.



Fonte: Adaptado de (YAZDANI; JOLAI, 2016).

inicial dos nós de borda em relação à tarefa é definido com base na comparação de seus recursos computacionais disponíveis. Posteriormente, a cada iteração, os nós de borda atualizam suas posições e se aproximam do alvo conforme a aptidão calculada e ajustada durante o processo de otimização. Ao final das iterações, o nó de borda que apresentar a melhor aptidão é selecionado para realizar o processamento da tarefa alvo.

Outra ação importante realizada pelo LOA e aplicada durante as iterações do PRIOBILL é o processo de migração, ilustrado na Figura 25. A migração ocorre ao longo das iterações do algoritmo. Os nós de borda da VEC são identificados como macho nômade (veículo laranja), fêmea nômade (veículo verde) ou fêmea residente (veículo azul), de acordo com sua prioridade, definida por meio de seus atributos, principalmente o número de entradas na VEC e o número de compartilhamentos realizados. Durante cada iteração, os nós de borda passam por um processo de avaliação e balanceamento populacional, no qual podem alterar seu grupo de acordo com a nova aptidão calculada. Dessa forma, um nó de borda pode migrar entre grupos, permanecer no grupo atual ou, caso seja identificada uma baixa capacidade para atender ao processamento da tarefa solicitada, ser inserido na lista de "mortos". Essa lista representa um conjunto de nós de borda considerados inadequados para o processo de seleção devido à análise de seus recursos disponíveis e desempenho durante as iterações.

A função de aptidão aplicada no PRIOBILL tem como objetivo encontrar a menor

Figura 24 – Movimento dos nós de borda do PRIOBILL para atacar a tarefa alvo.



Fonte: Gerado por IA.

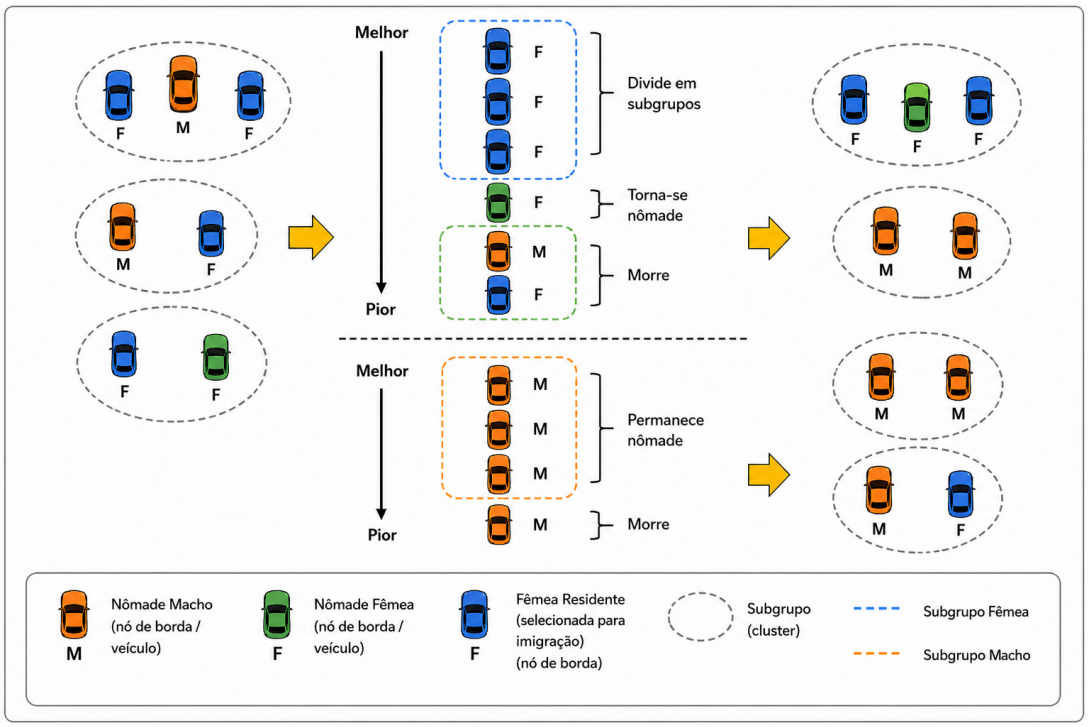
pontuação de aptidão com base nos recursos disponíveis do nó de borda, na quantidade de serviços processados por esse nó e no número de vezes que ele ingressou em uma VEC. A Equação 4.21 apresenta o cálculo utilizado, no qual é priorizada a quantidade de recursos disponíveis e aplicada uma penalização de acordo com o número de serviços executados e entradas realizadas pelo nó de borda. Essa penalização tem como finalidade distribuir os serviços de maneira mais justa e equilibrada entre os nós da VEC, reduzindo a possibilidade de sobrecarga de determinados nós e promovendo maior equidade durante o processo de alocação de tarefas.

$$F_i = w_R \cdot \left(1 - \frac{R_i}{R_{\max}}\right) + w_A \cdot \frac{A_i}{A_{\max}} + w_E \cdot \frac{E_i}{E_{\max}} \tag{4.21}$$

onde R_i corresponde à quantidade de recursos disponíveis no nó de borda, A_i representa a quantidade de serviços já executados pelo nó de borda e E_i corresponde à quantidade de entradas do nó de borda em uma VEC. Os valores de w_R, w_A e w_E representam os pesos atribuídos aos recursos disponíveis, às tarefas alocadas e ao contador de entradas na RSU, respectivamente.

Após o cálculo da aptidão e a identificação do melhor nó de borda em cada iteração, um movimento dos nós de borda é aplicado para atualizar suas posições, con-

Figura 25 – Etapa de migração do nó de borda



Fonte: Gerado por IA.

forme apresentado na Equação 4.22. Esse movimento é realizado utilizando as informações dos nós de borda, que são atualizadas a cada iteração em relação à presa, representada pela tarefa alvo a ser processada. Dessa forma, um movimento estocástico é aplicado em direção ao melhor ponto identificado pelo processo de otimização.

$$x_j^{t+1} = \left| x_j^t + r_j \left(x_j^* - x_j^t \right) \right| \tag{4.22}$$

onde x_j^t corresponde à posição atual do nó de borda, x_j^* representa a posição da tarefa e r_j é um vetor aleatório utilizado para realizar o movimento com uma fração aleatória em direção à tarefa.

O PRIOBILL possui um processo iterativo de otimização baseado no LOA, utilizado como mecanismo de tomada de decisão para selecionar o melhor nó de borda. O processo busca minimizar uma função de aptidão, enquanto realiza a atualização dinâmica das posições dos candidatos em relação à localização da tarefa.

O Algoritmo 6 apresenta uma abstração do mecanismo de tomada de decisão implementado neste trabalho. Como entrada, o algoritmo recebe os nós de borda da VEC $\mathcal{C}$ e as informações da tarefa a ser processada $\mathbf{r}_i$. Como saída, o nó de borda j^* identificado com maior potencial para processar a tarefa é selecionado. Inicialmente, o número máximo de iterações T é definido (linha 1). A função $checkMax$ é utilizada para encontrar o maior valor de recursos disponíveis dos nós de borda ($R_{\max}$), o maior número de entradas na VEC ($E_{\max}$) e o maior número de serviços executados ($A_{\max}$), valores que serão utilizados

Algoritmo 6 – PRIOBILL Seleção do nó de borda

Entrada: $\mathcal{C}$, $\mathbf{r}_i$
Saída: j^*
1 $T \leftarrow 20$;
2 $(R_{\max}, E_{\max}, A_{\max}) \leftarrow \text{checkMax}(\mathcal{B})$;
3 **Para** $t \leftarrow 1$ **to** T **faça**
4 **Para** $j \in \mathcal{C}$ **faça**
5 applymigration();
6 $S_j \leftarrow \text{Fitness}(j, R_{\max}, A_{\max}, E_{\max})$;
7 **Se** $S_j < S^*$ **então**
8 $S^* \leftarrow S_j$;
9 $j^* \leftarrow j$;
10 $\mathbf{x}_j \leftarrow \text{position}(j)$;
11 $\mathbf{x}^* \leftarrow \text{position}(\mathbf{r}_i)$;
12 $\mathbf{x}_j \leftarrow \text{updatePosition}(\mathbf{x}_j, \mathbf{x}^*)$;
13 **Retorna** j^*

no cálculo da aptidão (linha 2). Em seguida, a iteração do mecanismo é iniciada (linha 3), percorrendo todos os membros da VEC (linha 4). O processo de migração é aplicado para avaliar a fragilidade dos nós de borda (linha 5). Posteriormente, o cálculo da aptidão é executado, conforme detalhado no Algoritmo 7 (linha 6). O algoritmo verifica se a aptidão calculada é melhor que a encontrada anteriormente (linha 7). Caso seja, a melhor aptidão S^* (linha 8) e o melhor nó de borda j^* (linha 9) são atualizados. As informações referentes à posição atual do nó de borda $\mathbf{x}_j$ e à posição atual da tarefa $\mathbf{x}^*$ são coletadas (linhas 10 e 11) para executar a função de atualização de posição (linha 12). Após a conclusão de todas as iterações, o algoritmo retorna o nó de borda selecionado j^* (linha 13).

O Algoritmo PRIOBILL minimiza iterativamente a função de aptidão definida no Algoritmo 7. A função recebe como parâmetros o nó de borda da iteração (i), o maior valor de recursos disponíveis entre os nós de borda ($R_{\max}$), o maior número de entradas nas VECs ($E_{\max}$) e o maior número de serviços executados ($A_{\max}$). Como retorno, a função fornece a pontuação de aptidão calculada *score*. Utilizando as informações dos recursos disponíveis do nó de borda R_i, o número de alocações já realizadas pelo nó de borda A_i e a quantidade de vezes que ele ingressou em uma VEC E_i (linhas 1, 2 e 3, respectivamente), a Equação 4.21 é aplicada para calcular a aptidão do nó de borda (linha 4). Por fim, a função retorna o valor de *score* para o algoritmo de tomada de decisão (linha 5).

4.9 AVALIAÇÃO E ANÁLISES

Foram realizadas extensas simulações para avaliar e analisar o desempenho do PRIOBILL. O cenário de simulação foi desenvolvido com base em parâmetros represen-

Algoritmo 7 – Função de avaliação de aptidão

Entrada: i, $R_{\max}$, $A_{\max}$, $E_{\max}$
Saída: *score*

$R_i \leftarrow \text{availableResources}(i)$;
$A_i \leftarrow \text{allocatedTasks}(i)$;
$E_i \leftarrow \text{entryCount}(i)$;
$score \leftarrow w_R \cdot \left(1 - \frac{R_i}{R_{\max}}\right) + w_A \cdot \frac{A_i}{A_{\max}} + w_E \cdot \frac{E_i}{E_{\max}}$;
Retorna *score*

tativos de ambientes reais, sendo aplicadas variações em diferentes configurações para avaliar múltiplos cenários e desafios. A fim de tornar as simulações mais realistas, foram utilizadas as seguintes ferramentas:

- **SUMO.** O Simulador de Mobilidade Urbana (do inglês, *Simulation of Urban Mobility*) é responsável por gerenciar a mobilidade dos veículos, reproduzindo o comportamento do tráfego urbano e rodoviário;
- **OMNeT++.** O *Objective Modular Network Testbed in C++* é um simulador orientado a objetos e baseado em componentes, amplamente utilizado para simulação de redes de comunicação, incluindo protocolos e troca de mensagens entre veículos e RSUs;
- **VEINS.** O módulo *Vehicles in Network Simulation* (VEINS) é responsável por integrar o SUMO ao OMNeT++, permitindo a troca de mensagens e informações em tempo real entre o simulador de mobilidade e o simulador de redes.

4.9.1 Cenário e parâmetros da simulação

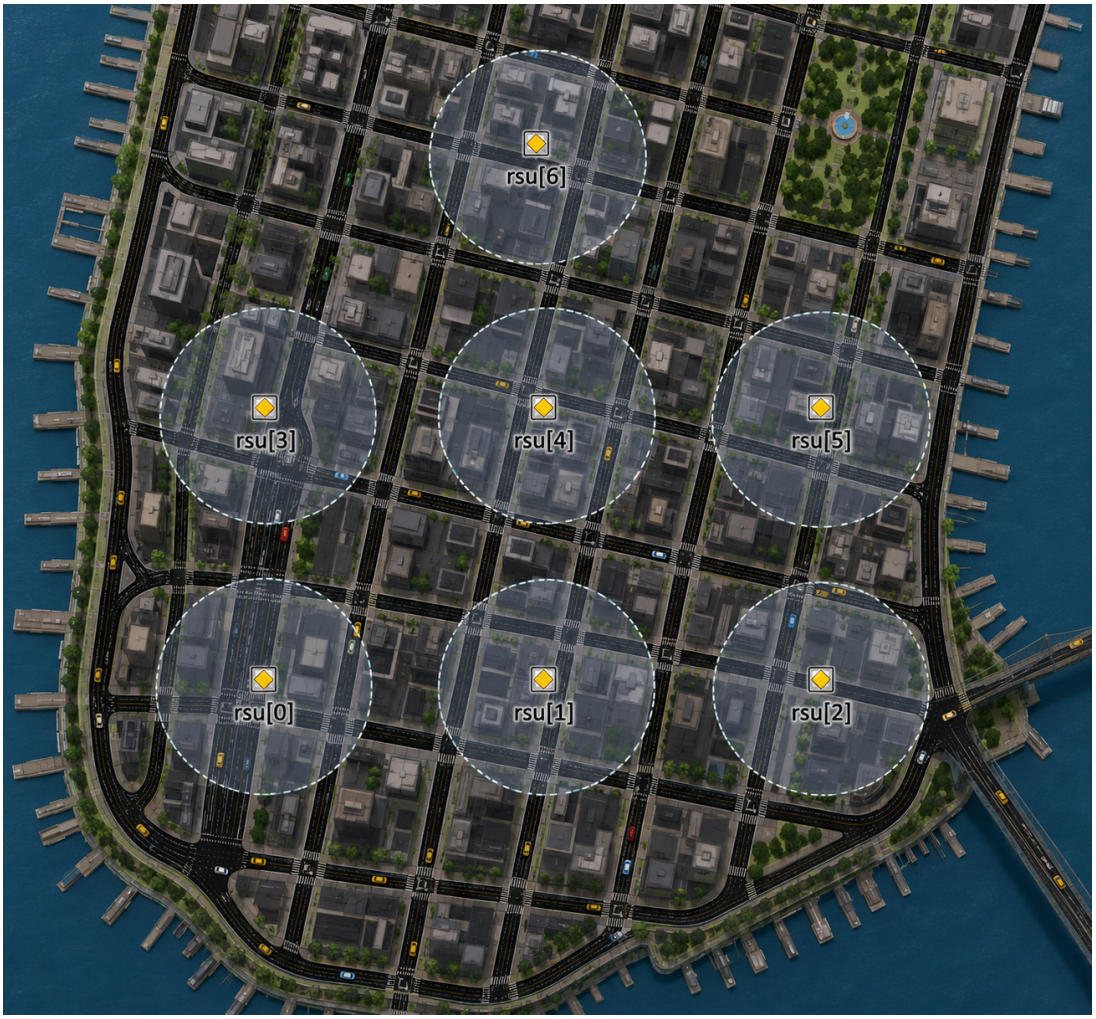
Um cenário de simulação realista foi implementado com base na região de Manhattan, conforme ilustrado na Figura 26. Nesse cenário, foram distribuídas sete RSUs ao longo da área de estudo com o objetivo de maximizar a cobertura de atendimento.

Para a construção do cenário, foram utilizadas ferramentas capazes de capturar e converter dados reais de mapas em um ambiente de mobilidade para simulação. O trecho apresentado na Figura 27 corresponde à região de Midtown Manhattan, na cidade de Nova Iorque. Os dados foram obtidos por meio do *OpenStreetMap*[1] e o recorte de aproximadamente 2100 m × 2200 m foi realizado utilizando a ferramenta *osmium-tool*[2].

Após a extração do trecho, a ferramenta *osmium-tool* também foi utilizada para converter o mapa para um arquivo com extensão `.xml` (`manhattan_small.osm.xml`). Em seguida, a rede do SUMO foi gerada por meio do comando *netconvert*. Durante essa

[1] https://www.openstreetmap.org.br/
[2] https://osmcode.org/osmium-tool/

Figura 26 – Midtown Manhattan com representação gráfica de RSUs.



Fonte: Gerado por IA com base no modelo do SUMO.

etapa, foi aplicado um filtro para incluir apenas veículos de passeio no mapa, reduzindo a complexidade da rede e evitando a geração de um cenário excessivamente grande e computacionalmente pesado.

Após a geração da rede do SUMO, foi utilizado o script *randomTrips.py* para criar o tráfego veicular. Foram configurados três níveis distintos de mobilidade, com o objetivo de representar diferentes condições de circulação de veículos ao longo da simulação: um período de baixo fluxo (madrugada), um período de fluxo moderado (manhã) e um período de congestionamento (horário de pico). Nos primeiros 600 segundos da simulação, foi inserido, em média, um veículo a cada 8 segundos. Entre 600 e 1200 segundos, essa média passou para um veículo a cada 3 segundos. Já no intervalo entre 1200 e 1800 segundos, foi inserido, em média, um veículo por segundo. Dessa forma, o cenário inicia com maior escassez de recursos computacionais, tornando o processo de alocação mais desafiador, e evolui para um ambiente com maior disponibilidade de recursos, porém sujeito a constantes alterações na topologia dos *clusters* em função da elevada mobilidade

Figura 27 – Trecho da Midtown Manhattan utilizado para simulação.



Fonte: Gerado por IA.

dos veículos.

Além disso, para tornar o fluxo de tráfego mais realista, em vez de permitir que os veículos entrassem exatamente no instante especificado, foi aplicada uma distribuição binomial, tornando aleatórios os instantes de partida dos veículos e limitando a quantidade máxima de veículos que podem entrar ou sair simultaneamente. Também foi aplicado um fator marginal, de modo que o tráfego gerado na área de simulação se iniciasse preferencialmente nas extremidades do mapa. A Tabela 3 apresenta as configurações específicas aplicadas a cada cenário de simulação com diferentes níveis de mobilidade.

Tabela 3 – Parâmetros de mobilidades

Parâmetros	**Madrugada**	**Manhã**	**Pico**
Horário considerado	5h	11h	17h
Tempo da simulação (s)	0 a 600	600 a 1200	1200 a 1800
Geração de veículos (s)	8	3	1
Distribuição binomial	2	3	4
Fator Marginal	5	8	10

Os veículos apresentam mobilidade aleatória, cada um com sua própria origem,

destino e rota, gerados por meio do SUMO. As Unidades de Beira de Estrada (RSUs) foram distribuídas ao longo do cenário com o objetivo de maximizar a área de cobertura, conforme ilustrado na Figura 28. Cada algoritmo foi executado durante 1800 segundos de simulação. Para representar diferentes níveis de demanda computacional, foram definidas quatro classes de tarefas:

- **Tarefas menores.** Os recursos de processamento, armazenamento e memória são gerados aleatoriamente com valores entre 1 e 3;
- **Tarefas intermediárias.** Os recursos de processamento, armazenamento e memória são gerados aleatoriamente com valores entre 4 e 6;
- **Tarefas maiores.** Os recursos de processamento, armazenamento e memória são gerados aleatoriamente com valores entre 7 e 10;
- **Tarefas aleatórias.** Os recursos de processamento, armazenamento e memória são gerados aleatoriamente com valores entre 1 e 10.

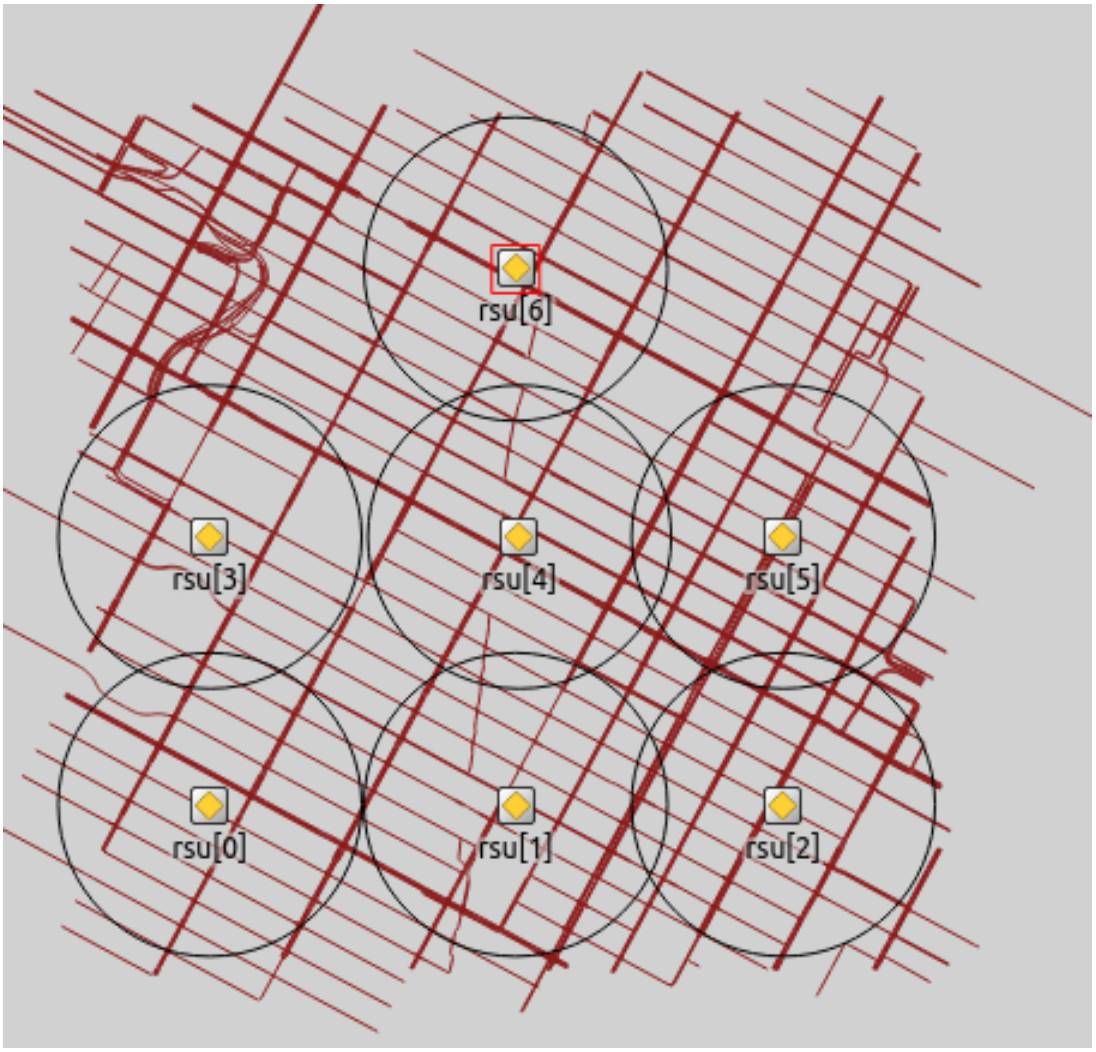
Para cada algoritmo avaliado, foram realizadas 33 execuções independentes, para proporcionar estabilidade e justiça nos resultados, sendo a média aritmética de cada métrica utilizada para a apresentação dos resultados. Também foi calculado um intervalo de confiança de 95%; entretanto, a variação observada foi muito pequena, não sendo perceptível nos resultados apresentados. Isso ocorreu devido a utilização de parâmetros de recursos, dos veículos e das tarefas, idênticos para todos os algoritmos, para garantir a igualdade na comparação dos mecanismos. Cada veículo possui um conjunto de tarefas associado ao seu identificador (*ID*), podendo receber aleatoriamente entre uma e cinco tarefas. No algoritmo 7 os valores de pesos aplicados para os *fitness* de recursos, atendimentos e entradas, foram definidos como 0.5, 0.3 e 0.2, respectivamente, para priorizar a quantidade de recursos no cálculo da aptidão, antes dos atendimentos e entradas, a fim de evitar que a escolha do melhor nó de borda seja somente direcionada para os atendimentos e entradas e, assim, evitar uma sobrecarga nos atendimentos. Na Tabela 4 apresentam-se os principais parâmetros gerais utilizados nas simulações.

4.9.2 Métricas de Avaliação de Desempenho

Para validar o PRIOBILL, o *framework* desenvolvido foi comparado com os trabalhos relacionados por meio da análise de seis métricas de desempenho: número de tarefas processadas, número de tarefas não processadas, quantidade de recursos utilizados, latência, taxa de transferência (*throughput*) e *billing*.

- **Tarefas processadas:** corresponde ao número de tarefas que foram recebidas pela RSU, priorizadas e encaminhadas para um nó de borda capaz de processá-las

Figura 28 – Posicionamento das RSUs no trecho de Midtown Manhattan no SUMO.



Fonte: Mapa retirado do SUMO.

(Equação (4.23)):

$$task\_processed = \sum task\_completed \tag{4.23}$$

onde *task_completed* representa cada tarefa concluída com sucesso e *task_processed* corresponde ao total de tarefas processadas durante a simulação;

- **Tarefas não processadas:** corresponde às tarefas que solicitaram recursos à RSU, mas que, após a seleção do nó de borda, não puderam ser processadas devido à indisponibilidade de recursos computacionais (Equação (4.24)):

$$unprocessed\_task = \sum uncompleted\_task \tag{4.24}$$

onde *uncompleted_task* representa cada tarefa que não pôde ser concluída e *unprocessed_task* corresponde ao total de tarefas não processadas durante a simulação;

- **Quantidade de recursos processados:** representa a quantidade total de recursos computacionais utilizados, considerando os recursos de armazenamento, processa-

Tabela 4 – Parâmetros gerais da simulação

Parâmetros	**Valores**
Chegada de tarefas	Distribuição de Poisson
Tempo de simulação	1800 segundos
Recursos dos veículos	
Memória (MB)	rand (1, 10)
Armazenamento (MB)	rand (1, 10)
Processamento (MB)	rand (1, 10)
Classe de tarefas	
Menores (MB)	rand (1, 3)
Intermediárias (MB)	rand (4, 6)
Maiores (MB)	rand (7, 10)
Aleatória (MB)	rand (1, 10)
Tarefas por veículos	rand (1, 5)
Max Velocidade Veículo	13,90 m/s
Min Distância percorrida	500m
Tipo do cenário	Midtown Manhattan
Área do cenário	aprox 2100m x 2200m
Communicação	V2I
Política de Clusterização	K-means
Numéro de RSUs	7
Faixa de comunicação RSU	400m
Peso p/ fitness - Recursos w_R	0.5
Peso p/ fitness - Atendimentos w_A	0.3
Peso p/ fitness - Entradas w_E	0.2

mento e memória disponibilizados pelos nós de borda (Equação (4.25)):

$$R_{\text{total}} = \sum_{i=1}^{N} (S_i + P_i + M_i) \tag{4.25}$$

onde S_i representa os recursos de armazenamento utilizados, P_i os recursos de processamento utilizados, M_i os recursos de memória utilizados e R_{total} corresponde à soma total dos recursos consumidos durante a simulação;

- **Latência:** corresponde ao tempo decorrido entre o início do processo de alocação de uma tarefa e a conclusão de seu processamento. Neste trabalho, é considerada a latência média $\bar{L}$, apresentada na Equação (4.26):

$$\bar{L} = \frac{1}{N} \sum_{i=1}^{N} L_i \tag{4.26}$$

onde L_i representa a latência da i-ésima tarefa e N corresponde ao número total de tarefas processadas;

- **Throughput:** corresponde à taxa de transferência do sistema, calculada pela razão entre o número de tarefas processadas com sucesso e o tempo total de simulação, conforme apresentado na Equação (4.27):

$$\Theta = \frac{|\mathcal{S}|}{T_{\text{sim}}} \tag{4.27}$$

  onde $\mathcal{S}$ representa o conjunto de tarefas processadas com sucesso e T_{sim} corresponde ao tempo total de simulação.

- **Billing:** O sistema de *billing* contabiliza os recursos processados com base no compartilhamento de recursos computacionais realizado pelos nós de borda.

Para a cobrança, este trabalho adota o modelo de custos das políticas de precificação do AWS Fargate (AWS, Accessado: 16/05/2026), no qual os recursos de memória, processamento e armazenamento são tarifados separadamente. Como a documentação da AWS apresenta os valores de cobrança por hora, neste trabalho esses custos foram convertidos para segundos, de acordo com os parâmetros adotados nas simulações. Na Tabela 5 são apresentados os valores de cobrança por recurso após a conversão da tarifa horária para uma tarifa por segundo, bem como a conversão da moeda de dólar para real (R$), considerando a cotação de R$5,15 em 19/06/2026.

Na AWS, o custo de processamento é definido por CPU virtual ($vCPU$), sendo que cada $vCPU$ suporta até 8 GB de memória (AWS, Accessado: 16/05/2026). Assim, neste trabalho, considera-se a utilização de um $vCPU$ por processo, uma vez que as tarefas simuladas demandam, no máximo, 10 MB de processamento. Os custos referentes aos recursos de memória e armazenamento já são considerados em megabytes (MB).

Tabela 5 – Conversão de valores: Horas para segundos para Reais

Recurso	**Por hora**	**Por segundo**	**Em Reais (R$)**
Processamento (1x vCPU)	$0,08096	$0,00002248	R$0,00012
Memoria (GB)	$0,004445	0,0000063	R$0,00000123
Armazenamento (GB)	$0,000111	$0,0000000308	R$0,00000016

Fonte: Preço baseado no AWS (Accessado: 16/05/2026)

4.9.3 Algoritmos avaliados

Para validar o mecanismo desenvolvido, o PRIOBILL foi comparado com diferentes técnicas heurísticas e metaheurísticas empregadas na literatura para problemas de otimização e alocação de recursos, conforme os trabalhos de Lieira et al. (2024), Lieira et al. (2022a), Kumar et al. (2022), Azizi et al. (2022), Choo, Kim e Pack (2018) e Hosseini, Nickray e Ghanbari (2022). Esses estudos utilizam, respectivamente, as técnicas GWO,

Algoritmo de Otimização de Baleias (WOA), PSO, Greedy, Random e AHP. Além dessas abordagens, também é avaliada a proposta desenvolvida neste trabalho (PRIOBILL). As principais características de cada método são descritas a seguir:

- **GULOSO.** Trata-se de um mecanismo de seleção amplamente utilizado na literatura, no qual o primeiro nó de borda capaz de atender à solicitação é escolhido para processar a tarefa (AZIZI et al., 2022);
- **ALEATÓRIO.** Consiste em um algoritmo tradicional que seleciona aleatoriamente um dos nós de borda ativos do *cluster* para processar a tarefa (CHOO; KIM; PACK, 2018);
- **AHP.** O *Analytic Hierarchy Process* (AHP) é um método multicritério de apoio à decisão que utiliza uma matriz de comparação ponderada para identificar o nó de borda mais adequado ao processamento da tarefa (HOSSEINI; NICKRAY; GHANBARI, 2022);
- **WOA.** Trata-se de um algoritmo bioinspirado baseado no Algoritmo de Otimização de Baleias, que reproduz o comportamento de caça das baleias, permitindo que os nós de borda "cerquem"a tarefa até que o melhor candidato seja selecionado (LIEIRA et al., 2022a);
- **PSO.** Trata-se de um algoritmo metaheurístico baseado na Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO), inspirado no comportamento coletivo de bandos de pássaros e cardumes de peixes durante a busca por um objetivo. Nesse método, os nós de borda representam as partículas do enxame e compartilham informações entre si para identificar o melhor nó para processar a tarefa (KUMAR et al., 2022);
- **GWO.** É um algoritmo bioinspirado baseado na Otimização do Lobo Cinzento (*Grey Wolf Optimizer*), que reproduz a estratégia de caça dessa espécie para selecionar o nó de borda mais adequado ao processamento da tarefa solicitada (LIEIRA et al., 2023b);
- **PRIOBILL.** Corresponde à estrutura proposta neste trabalho para alocação dinâmica de tarefas em Computação de Borda Veicular, incorporando um mecanismo de priorização baseado no compartilhamento de recursos. O algoritmo de tomada de decisão responsável por selecionar o nó de borda é inspirado no Algoritmo de Otimização do Leão.

4.10 RESULTADOS

Nesta seção, são apresentados os resultados obtidos nas simulações, bem como uma análise comparativa entre os algoritmos e as métricas de desempenho avaliadas. Além

disso, é apresentada uma estimativa do sistema de *billing* implementado no framework PRIOBILL, considerando os recursos computacionais compartilhados pelos nós de borda.

Os resultados estão organizados em subseções, cada uma dedicada a uma das métricas avaliadas, acompanhadas de suas respectivas análises e considerações parciais. Ao final da seção, é apresentada uma discussão consolidada dos principais resultados obtidos.

4.10.1 Tarefas processadas

Nesta seção, são apresentados os resultados referentes ao número de tarefas processadas em todos os cenários avaliados.

Na Figura 29 apresenta-se o número de tarefas menores processadas no cenário da madrugada. Nos primeiros intervalos, até 200 s, não houve tarefas processadas devido à baixa quantidade de veículos presentes no cenário. No intervalo de 300 s, o PRIOBILL apresentou o melhor desempenho, processando, em média, 2,95% mais tarefas que o AHP, 7,81% mais que o PSO, 7,96% mais que o ALEATÓRIO, 8,23% mais que o WOA, 8,61% mais que o GWO e 25,76% mais tarefas que o GULOSO.

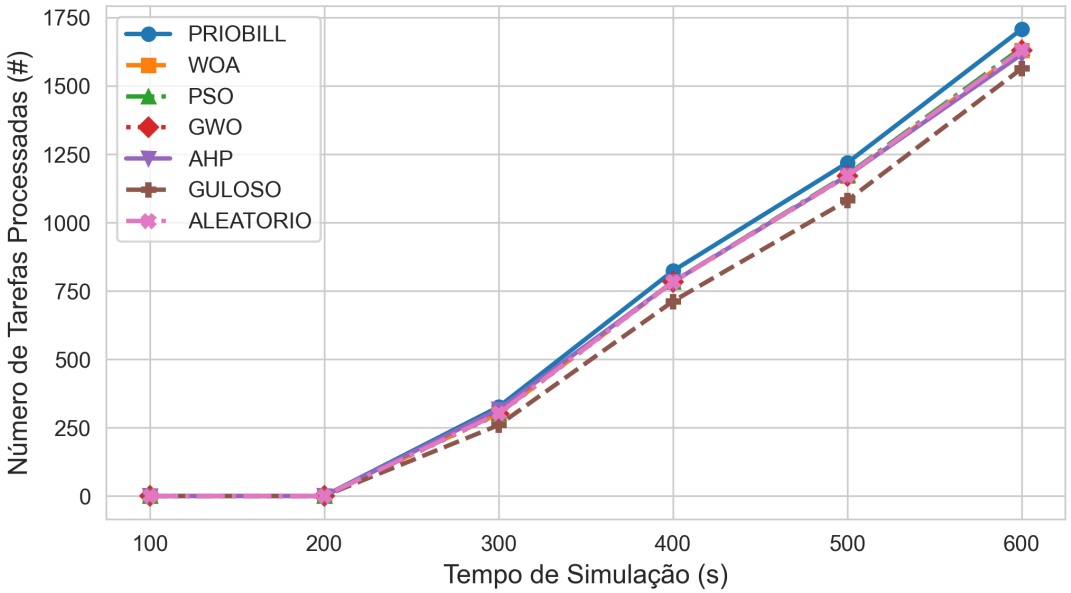
No intervalo de 400 s, o PRIOBILL processou entre 5,04% e 5,23% mais tarefas que os algoritmos PSO, AHP, ALEATÓRIO, GWO e WOA, respectivamente, além de 15,62% mais tarefas que o GULOSO. Aos 500 s de simulação, o PRIOBILL manteve o melhor desempenho, processando 3,67% mais tarefas que o PSO, 3,94% mais que o ALEATÓRIO, 4,00% mais que o AHP, 4,04% mais que o WOA, 4,07% mais que o GWO e 12,75% mais que o GULOSO.

No último intervalo desse cenário, aos 600 s, o PRIOBILL novamente apresentou o melhor desempenho, processando 4,15% mais tarefas que o PSO, 4,64% mais que o ALEATÓRIO, 4,50% mais que o WOA e o GWO, 5,69% mais que o AHP e 9,22% mais que o GULOSO.

Na Figura 30 apresenta-se o número de tarefas processadas no cenário intermediário, correspondente ao período diurno. Aos 700 s de simulação, quando o número de veículos no cenário começou a aumentar, o PRIOBILL apresentou o melhor desempenho, com média de 2.187 tarefas processadas. O segundo melhor resultado foi obtido pelo PSO, com 2.084 tarefas processadas, seguido pelo GWO, com 2.068, WOA, com 2.067, ALEATÓRIO, com 2.064, e GULOSO, com 2.012 tarefas processadas.

Aos 800 s de simulação, o PRIOBILL novamente apresentou o melhor desempenho, com 2.690 tarefas processadas. O GULOSO obteve o segundo melhor resultado, com 2.566 tarefas processadas, seguido pelo ALEATÓRIO, com 2.563, PSO, com 2.557, GWO, com 2.544, AHP, com 2.541, e WOA, com 2.540 tarefas processadas.

Figura 29 – Número de tarefas menores processadas no horário da madrugada.



Fonte: Gerado pelo autor.

Aos 900 s de simulação, GULOSO e PRIOBILL apresentaram os melhores desempenhos, com 3.100 e 3.073 tarefas processadas, respectivamente. Em seguida, ficaram PSO, GWO, WOA, ALEATÓRIO e AHP, com 2.973, 2.964, 2.960, 2.957 e 2.928 tarefas processadas, respectivamente.

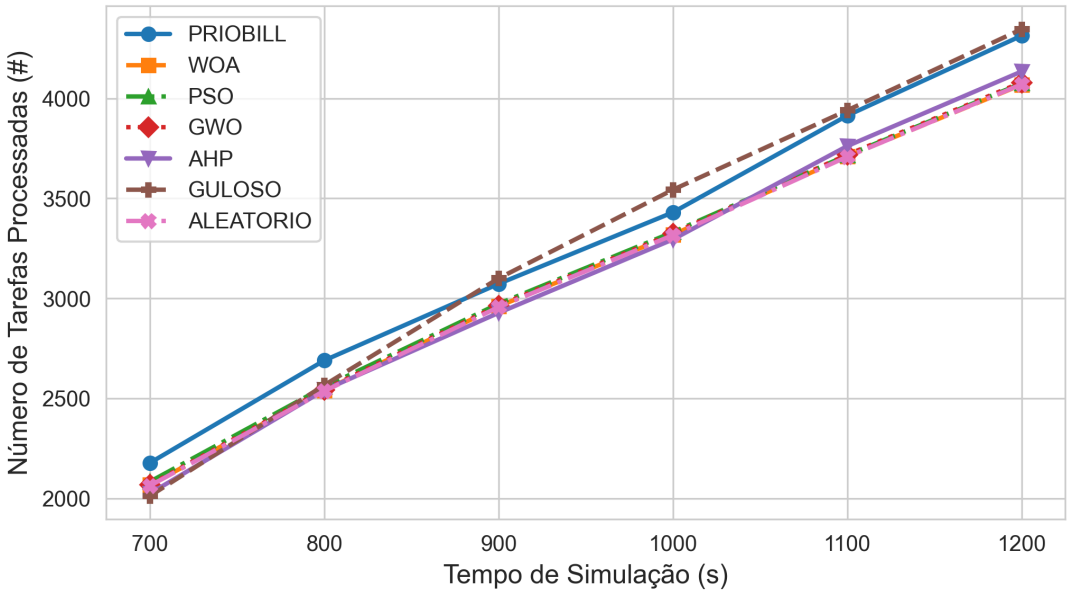
Aos 1.000 s, novamente GULOSO e PRIOBILL obtiveram os melhores resultados, com 3.543 e 3.430 tarefas processadas, respectivamente. O PSO processou 3.331 tarefas, o GWO 3.324, o WOA e o ALEATÓRIO 3.319 e 3.316, respectivamente, enquanto o AHP processou 3.295 tarefas.

Aos 1.100 s de simulação, apenas GULOSO e PRIOBILL ultrapassaram a marca de 3.900 tarefas processadas, com 3.940 e 3.914 tarefas, respectivamente.

Por fim, aos 1.200 s, GULOSO e PRIOBILL novamente apresentaram os melhores desempenhos, com 4.346 e 4.313 tarefas processadas, respectivamente. O AHP processou 4.134 tarefas, enquanto GWO, PSO, ALEATÓRIO e WOA processaram 4.080, 4.075, 4.070 e 4.068 tarefas, respectivamente.

Na Figura 31 é apresentado o número de tarefas processadas no cenário de maior pico de tráfego. O GULOSO e o PRIOBILL continuaram apresentando os melhores desempenhos aos 1.300 s de simulação, mesmo com o aumento da quantidade de veículos no cenário. A diferença entre GULOSO e PRIOBILL foi de apenas 0,88% no número de tarefas processadas, enquanto o AHP processou 4,04% menos tarefas que o GULOSO, seguido por GWO, PSO, ALEATÓRIO e WOA, com diferenças de 5,09%, 5,11%, 5,18% e 5,41%, respectivamente.

Figura 30 – Número de tarefas menores processadas no horário da manhã.



Fonte: Gerado pelo autor.

Aos 1.400 s, o GULOSO apresentou novamente o melhor desempenho, seguido de perto pelo PRIOBILL, que processou apenas 1,84% menos tarefas. O AHP processou 4,26% menos tarefas que o GULOSO. GWO e ALEATÓRIO apresentaram desempenho 4,26% inferior, enquanto PSO e WOA processaram 4,38% e 4,65% menos tarefas, respectivamente.

Aos 1.500 s de simulação, o GULOSO manteve o melhor desempenho, seguido pelo PRIOBILL, com apenas 2,22% menos tarefas processadas. O AHP processou 4,98% menos tarefas. Já WOA, PSO, ALEATÓRIO e GWO apresentaram desempenhos próximos, com reduções de 4,32%, 4,25%, 4,10% e 4,04%, respectivamente.

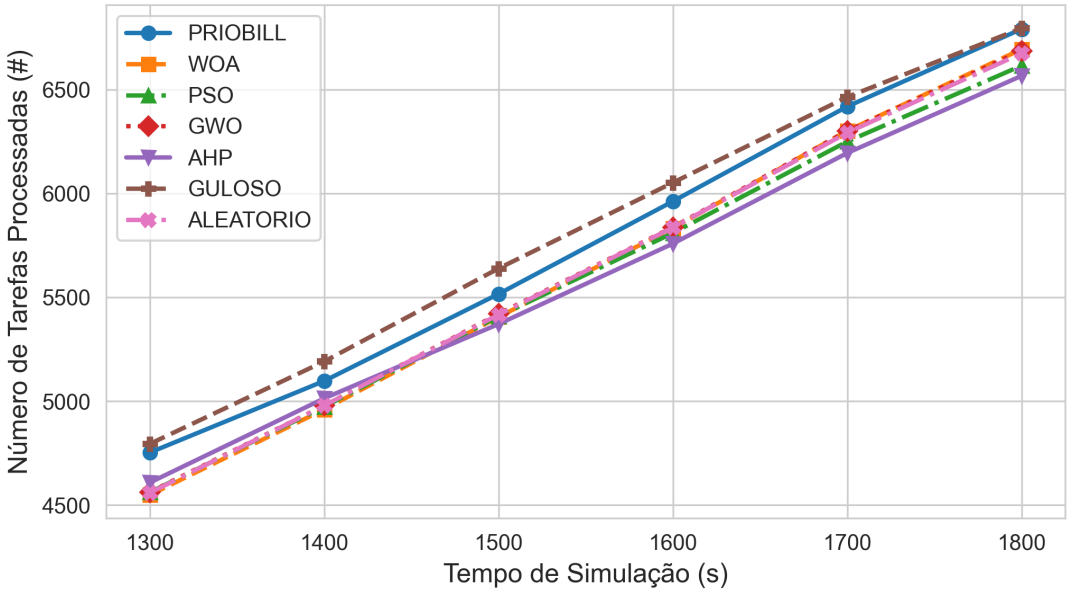
Aos 1.600 s de simulação, o PRIOBILL reduziu a diferença em relação ao GULOSO, processando apenas 1,51% menos tarefas. GWO, ALEATÓRIO e WOA processaram 3,76%, 3,70% e 3,67% menos tarefas, respectivamente, enquanto o PSO apresentou uma redução de 4,17% e o AHP de 5,10%.

Aos 1.700 s, o PRIOBILL continuou melhorando seu desempenho no cenário com maior número de veículos, processando apenas 0,71% menos tarefas que o GULOSO. O WOA processou 2,58% menos tarefas, o GWO ficou 2,60% abaixo, seguido pelo ALEATÓRIO, com 2,73% menos tarefas. O PSO processou 3,43% menos tarefas, enquanto o AHP apresentou o pior desempenho, ficando 4,35% abaixo do GULOSO.

Ao final da simulação, aos 1.800 s, PRIOBILL e GULOSO apresentaram desempenhos praticamente equivalentes, com vantagem de apenas 0,05% para o GULOSO no número de tarefas processadas. Em seguida, ficaram WOA, GWO, ALEATÓRIO, PSO e AHP, com 1,51%, 1,63%, 1,81%, 2,72% e 3,49% menos tarefas processadas, respectiva-

mente.

Figura 31 – Número de tarefas menores processadas no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 32 é apresentado o número de tarefas intermediárias processadas no cenário da madrugada. Nos primeiros 200 s de simulação, devido à baixa quantidade de veículos circulando no cenário, nenhuma tarefa foi processada.

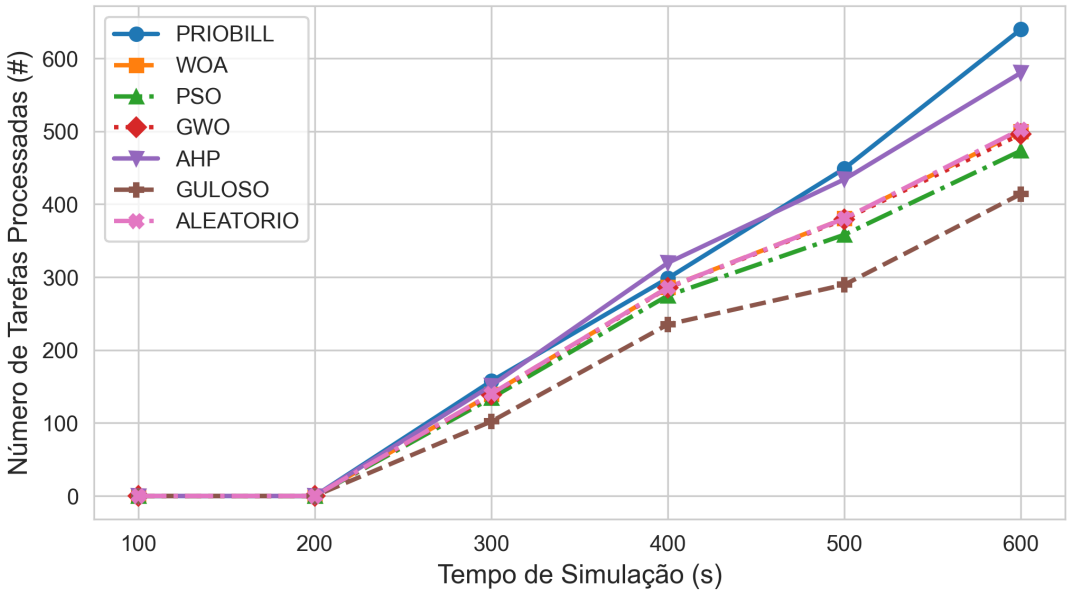
Aos 300 s de simulação, o PRIOBILL apresentou o melhor desempenho, processando, em média, 4% mais tarefas que o AHP, 12,44% mais que o ALEATÓRIO e 12,61% mais que o WOA. Além disso, obteve desempenho 13,70% superior ao GWO, 17,26% superior ao PSO e 54,09% superior ao GULOSO.

Aos 400 s de simulação, o AHP apresentou o melhor desempenho, seguido de perto pelo PRIOBILL, que processou 7,06% menos tarefas. O WOA processou 11,78% menos tarefas, seguido pelo ALEATÓRIO, com 11,83%, GWO, com 12,00%, PSO, com 16,32%, e GULOSO, com 35,91% menos tarefas processadas.

Aos 500 s, o PRIOBILL voltou a apresentar o melhor desempenho, seguido pelo AHP, que processou 3,43% menos tarefas. Na sequência, ficaram WOA, com 18,02%, ALEATÓRIO, com 18,04%, e GWO, com 18,40% menos tarefas processadas. Os desempenhos mais distantes foram observados para o PSO, com 25,36%, e para o GULOSO, com 55,03% menos tarefas processadas.

Ao final desse cenário, aos 600 s de simulação, o PRIOBILL processou, em média, 640 tarefas. O AHP apresentou o segundo melhor desempenho, com 10,27% menos tarefas processadas, seguido por ALEATÓRIO, WOA, GWO, PSO e GULOSO, com reduções de 27,28%, 27,95%, 29,00%, 35,18% e 54,52%, respectivamente.

Figura 32 – Número de tarefas intermediárias processadas no horário da madrugada.



Fonte: Gerado pelo autor.

O número de tarefas intermediárias processadas no período da manhã é apresentado na Figura 33. Aos 700 s de simulação, com o aumento da frequência de entrada de veículos nos últimos 100 s, o PRIOBILL atingiu uma média de 799 tarefas processadas. O segundo melhor desempenho foi obtido pelo AHP, com 19,36% menos tarefas processadas. Em seguida, ficaram GWO, ALEATÓRIO, WOA, PSO e GULOSO, com desempenhos 29,73%, 30,25%, 30,67%, 35,92% e 50,29% inferiores, respectivamente.

Aos 800 s, o PRIOBILL ampliou sua vantagem em relação aos demais algoritmos, processando, em média, 1.067 tarefas, o que representa 30,98% mais tarefas que o AHP, o segundo melhor método. O GWO processou 33,08% menos tarefas, o ALEATÓRIO 33,40% e o WOA 22,32% menos tarefas. O PSO e o GULOSO apresentaram os piores desempenhos, com reduções de 38,85% e 43,10%, respectivamente.

Aos 900 s de simulação, o PRIOBILL processou, em média, 1.162 tarefas. GWO, ALEATÓRIO, WOA e AHP apresentaram resultados na faixa de 24% menos tarefas processadas. O GULOSO processou 25,62% menos tarefas, enquanto o PSO apresentou o pior desempenho, com 28,96% menos tarefas processadas.

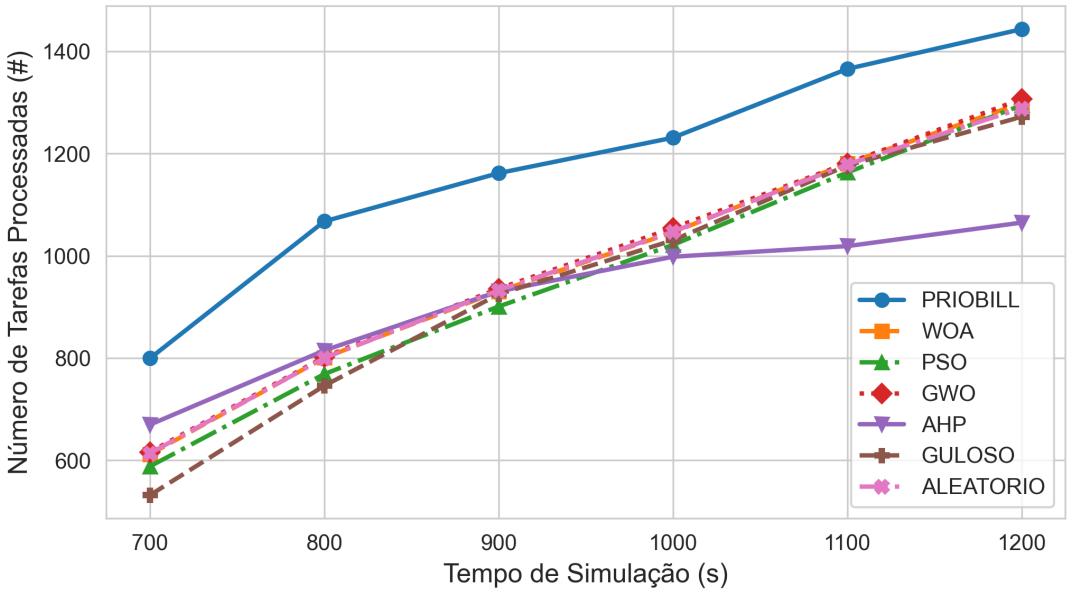
Aos 1.000 s de simulação, o PRIOBILL novamente apresentou o melhor desempenho, seguido pelo GWO, com 16,77% menos tarefas processadas. Na sequência, ficaram ALEATÓRIO, com 17,56%, e WOA, com 17,68% menos tarefas processadas. GULOSO, PSO e AHP apresentaram os menores desempenhos, com reduções de 19,48%, 20,53% e 23,33%, respectivamente.

Aos 1.100 s, o PRIOBILL obteve a maior média de tarefas processadas. O GWO apresentou o segundo melhor desempenho, com 15,61% menos tarefas processadas. O

WOA ficou muito próximo, com 15,63% menos tarefas, seguido pelo ALEATÓRIO, com 15,84% menos tarefas. O GULOSO processou 16,05% menos tarefas, o PSO 17,41% menos e o AHP apresentou o pior desempenho, com 34,01% menos tarefas processadas.

Ao final desse cenário, aos 1.200 s de simulação, o PRIOBILL processou 1.443 tarefas. O GWO processou 10,47% menos tarefas, seguido por WOA, PSO, ALEATÓRIO, GULOSO e AHP, com reduções de 11,16%, 11,50%, 12,01%, 13,48% e 35,53%, respectivamente.

Figura 33 – Número de tarefas intermediárias processadas no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 34 é apresentado o número de tarefas intermediárias processadas no horário de pico. O PRIOBILL obteve o melhor desempenho aos 1.300 segundos de simulação, seguido pelo GWO e pelo WOA, que processaram 3,06% e 3,66% menos tarefas, respectivamente. O PSO processou 4,15% menos tarefas, enquanto o ALEATÓRIO apresentou 4,25% menos tarefas processadas. O GULOSO teve um desempenho 7,01% inferior ao do PRIOBILL, e o AHP apresentou o pior resultado, com 33,45% menos tarefas processadas.

No cenário com 1.400 segundos, o GWO obteve o melhor desempenho. Em seguida ficaram o WOA, o ALEATÓRIO, o PSO, o PRIOBILL, o GULOSO e o AHP, com 0,41%, 0,64%, 1,09%, 2,36%, 3,41% e 30,63% menos tarefas processadas, respectivamente.

Com 1.500 segundos de simulação, o GULOSO voltou a apresentar o melhor desempenho. Em seguida ficaram o GWO, o WOA, o ALEATÓRIO, o PSO e o PRIOBILL, com 2,32%, 2,62%, 2,69%, 2,88% e 4,01% menos tarefas processadas, respectivamente. O AHP novamente apresentou o pior desempenho, com 33,11% menos tarefas processadas.

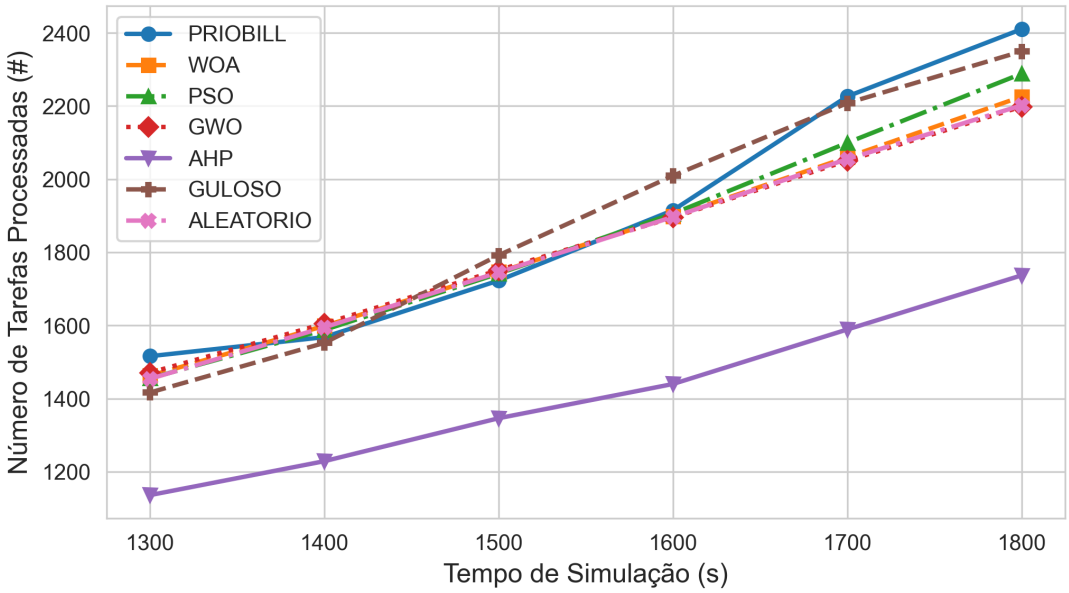
Aos 1.600 segundos, o GULOSO processou, em média, 2.009 tarefas. O PRIO-

BILL melhorou seu desempenho e ficou muito próximo do melhor resultado, processando apenas 4,94% menos tarefas. Em seguida vieram o PSO, o WOA, o ALEATÓRIO, o GWO e o AHP, com 5,45%, 5,85%, 5,93%, 6,02% e 39,57% menos tarefas processadas, respectivamente.

Aos 1.700 segundos de simulação, o PRIOBILL confirmou sua melhora e voltou a apresentar o melhor desempenho. O GULOSO ficou na segunda colocação, com apenas 0,83% menos tarefas processadas. Em seguida vieram o PSO, o WOA, o ALEATÓRIO, o GWO e o AHP, com 6,04%, 8,08%, 8,34%, 8,58% e 40,12% menos tarefas processadas, respectivamente.

No último intervalo, aos 1.800 segundos, o PRIOBILL novamente apresentou o melhor desempenho, com média de 2.410 tarefas processadas. O GULOSO ficou em segundo lugar, com 2,58% menos tarefas processadas. O PSO apresentou 5,32% menos tarefas processadas, seguido pelo WOA, ALEATÓRIO, GWO e AHP, com reduções de 8,30%, 9,46%, 9,65% e 38,77%, respectivamente.

Figura 34 – Número de tarefas intermediárias processadas no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 35 é apresentado o número de tarefas maiores processadas no cenário da madrugada. Nos primeiros 200 segundos de simulação, nenhum mecanismo conseguiu processar tarefas, devido à baixa quantidade de veículos no cenário. Aos 300 segundos de simulação, o PRIOBILL processou, em média, 44 tarefas. O PSO apresentou o segundo melhor desempenho, com média de 14,7 tarefas processadas, seguido pelo ALEATÓRIO, com 14,6, pelo WOA, com 13,9, e pelo GWO, com média de 13,5 tarefas processadas.

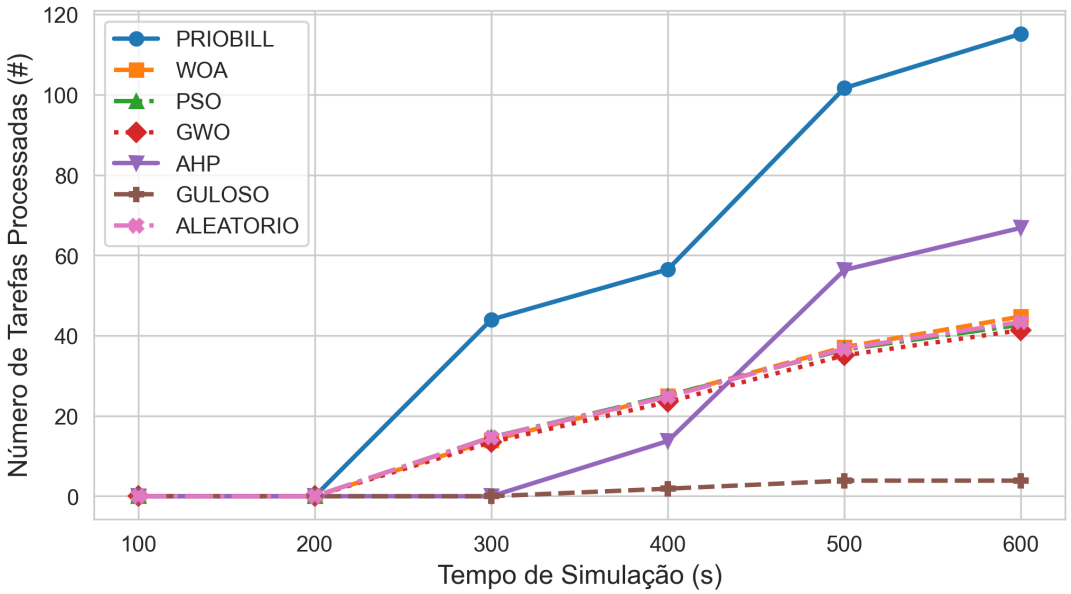
Com 400 segundos de simulação, o PRIOBILL processou, em média, 56,5 tarefas, seguido pelo WOA e pelo PSO, ambos com média de 25 tarefas processadas. O

ALEATÓRIO processou, em média, 24,8 tarefas, seguido pelo GWO, com 23,5, pelo AHP, com 13,7, e pelo GULOSO, com apenas 1,9 tarefa processada, em média.

Aos 500 segundos de simulação, o PRIOBILL atingiu a média de 101,7 tarefas processadas. O AHP processou, em média, 56,4 tarefas, seguido pelo WOA, ALEATÓRIO, PSO, GWO e GULOSO, com médias de 37,1, 36,7, 36,5, 35,1 e 3,9 tarefas processadas, respectivamente.

Com 600 segundos de simulação, o PRIOBILL manteve o melhor desempenho, alcançando média de 115,2 tarefas processadas. O AHP processou, em média, 66,8 tarefas, seguido pelo WOA, com 44,7, pelo ALEATÓRIO, com 43,4, pelo PSO, com 42,7, pelo GWO, com 41,4, e pelo GULOSO, com média de 3,9 tarefas processadas.

Figura 35 – Número de tarefas pesadas processadas no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 36 apresenta-se os resultados do número de tarefas maiores processadas no horário da manhã. Nesse cenário, observa-se a dificuldade imposta por uma simulação realista com tarefas de maior porte, na qual poucas ou até mesmo nenhuma tarefa foi alocada em determinados intervalos devido à escassez de recursos.

Com 700 segundos de simulação, o PRIOBILL atingiu a média de 117,6 tarefas processadas, seguido pelo AHP, com média de 66,8 tarefas. O WOA apresentou média de 46,4 tarefas processadas, enquanto o ALEATÓRIO processou 45,3 tarefas. O GWO alcançou média de 43,3 tarefas processadas e o GULOSO processou apenas 10 tarefas.

Com 800 segundos de simulação, o PRIOBILL e o AHP mantiveram suas médias de tarefas processadas, com 117,6 e 66,8 tarefas, respectivamente. O WOA, o ALEATÓRIO, o PSO e o GWO apresentaram pequenas evoluções, alcançando médias de 48,9, 47,8, 47,1

e 45,7 tarefas processadas, respectivamente. O GULOSO também não conseguiu processar novas tarefas, mantendo sua média de 10 tarefas.

Aos 900 segundos, o PRIOBILL permaneceu com o melhor desempenho, mesmo sem processar nenhuma tarefa no intervalo anterior. O AHP e o GULOSO também não realizaram novos atendimentos, mantendo suas médias de 66,8 e 10 tarefas processadas, respectivamente. O WOA, o ALEATÓRIO, o PSO e o GWO apresentaram pequenas variações, atingindo médias de 49,0, 47,9, 47,2 e 45,8 tarefas processadas, respectivamente.

Com 1.000 segundos de simulação, a escassez de recursos dos nós de borda permaneceu, e o PRIOBILL manteve sua média de 117,6 tarefas processadas. O AHP também manteve seu desempenho. O WOA, o ALEATÓRIO, o PSO, o GWO e o GULOSO apresentaram pequenas evoluções, alcançando médias de 49,5, 48,6, 47,7, 46,3 e 10,8 tarefas processadas, respectivamente.

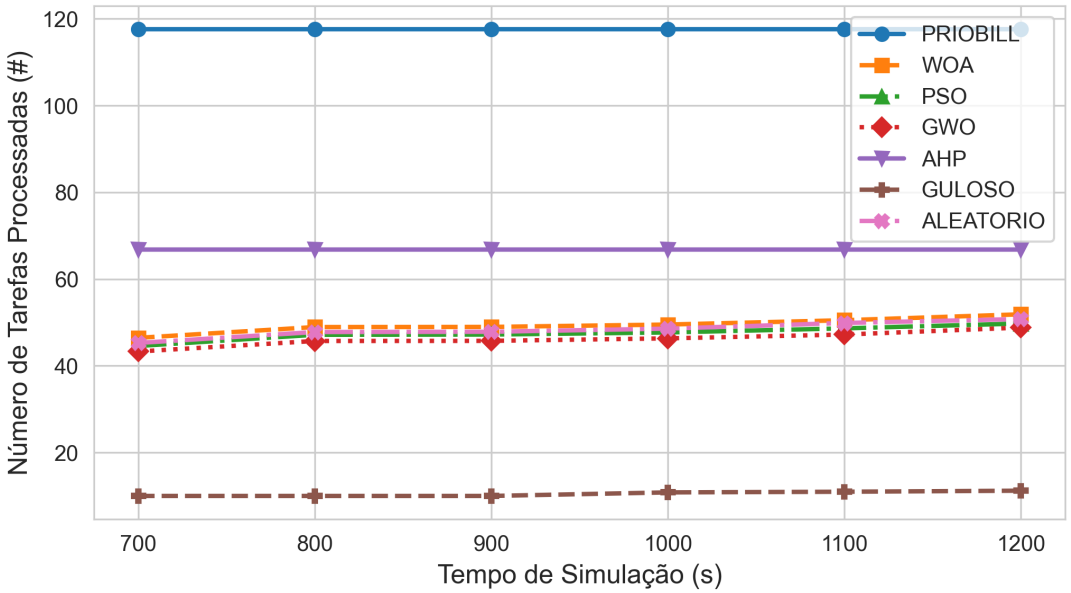
Nos dois últimos intervalos, aos 1.100 e 1.200 segundos, o PRIOBILL não processou novas tarefas e manteve a média de 117,6 tarefas processadas, permanecendo, ainda assim, como o mecanismo de melhor desempenho. O AHP também não realizou novos atendimentos nesse período, mas continuou ocupando a segunda colocação. O WOA alcançou médias de 50,5 e 51,9 tarefas processadas, respectivamente. O ALEATÓRIO obteve médias de 49,9 e 50,8 tarefas processadas nos dois últimos intervalos. O PSO atingiu médias de 48,6 e 49,7 tarefas processadas, enquanto o GWO alcançou 47,2 e 48,8 tarefas processadas. Por sua vez, o GULOSO processou 11 e 11,2 tarefas nos dois últimos intervalos, respectivamente.

Esse cenário evidencia a dificuldade enfrentada quando há escassez de recursos nas VECs. Mesmo com uma quantidade intermediária de veículos circulando no cenário, as tarefas maiores exigem recursos que muitos nós de borda não possuem disponíveis, reduzindo significativamente o número de alocações realizadas. Como consequência, todos os mecanismos apresentaram baixo número de tarefas processadas. Ainda assim, o PRIOBILL manteve o melhor desempenho ao longo de todo o cenário.

O número de tarefas maiores processadas no horário de pico é apresentado na Figura 37. Aos 1.300 segundos de simulação, o PRIOBILL manteve a média de 117,6 tarefas processadas. O AHP também manteve sua média de 66,8 tarefas processadas. O WOA processou, em média, 56 tarefas, enquanto o ALEATÓRIO atingiu média de 55,2 tarefas. O PSO e o GWO processaram, em média, 52,8 tarefas cada. Já o GULOSO processou 16,8 tarefas.

Aos 1.400 segundos, o PRIOBILL elevou sua média para 121,8 tarefas processadas, mantendo o melhor desempenho. O AHP permaneceu na segunda colocação, com média de 67,9 tarefas processadas. O GWO melhorou seu desempenho, alcançando média de 62,5 tarefas processadas.

Figura 36 – Número de tarefas pesadas processadas no horário da manhã.



Fonte: Gerado pelo autor.

Com 1.500 segundos de simulação, o PRIOBILL atingiu média de 122,8 tarefas processadas. O GWO apresentou o segundo melhor desempenho, com média de 72,5 tarefas processadas, seguido de perto pelo AHP, com 71,4 tarefas. O ALEATÓRIO, o WOA, o PSO e o GULOSO completaram a classificação, com médias de 66,9, 65,6, 63,2 e 32,6 tarefas processadas, respectivamente.

Aos 1.600 segundos, o PRIOBILL manteve o melhor desempenho, processando, em média, 123 tarefas. O GWO atingiu média de 81,6 tarefas processadas e o AHP processou 79,2 tarefas. O ALEATÓRIO, o WOA, o PSO e o GULOSO apresentaram médias de 72,4, 70,1, 68,1 e 44,2 tarefas processadas, respectivamente.

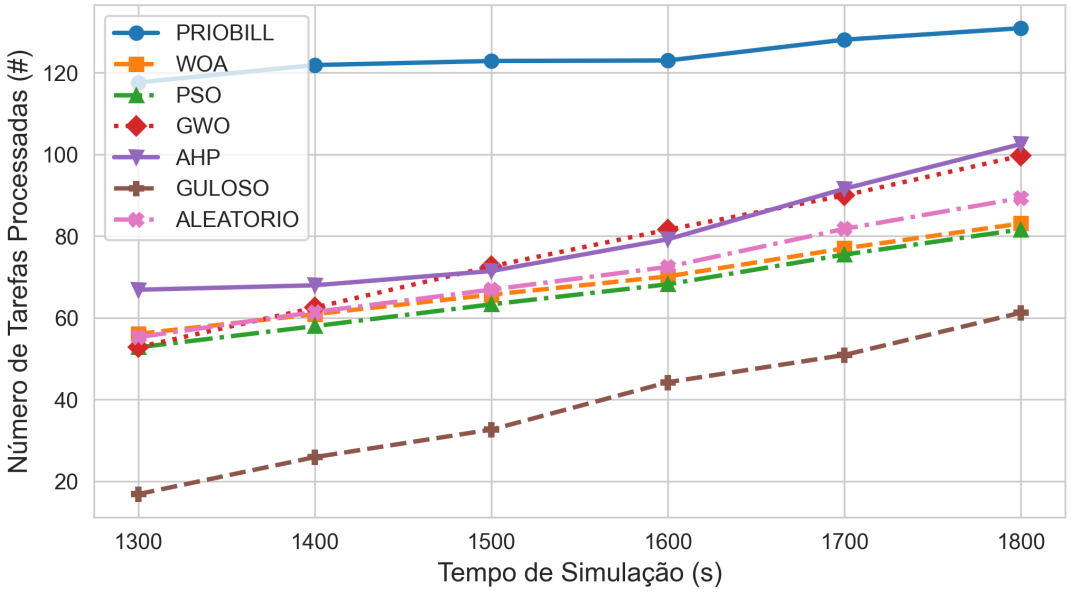
Aos 1.700 segundos de simulação, o PRIOBILL continuou apresentando o melhor desempenho, alcançando média de 128 tarefas processadas. O AHP retomou a segunda colocação, com média de 91,6 tarefas processadas, seguido de perto pelo GWO, com 89,9. O ALEATÓRIO processou, em média, 81,8 tarefas, enquanto o WOA, o PSO e o GULOSO atingiram médias de 77,0, 75,4 e 50,9 tarefas processadas, respectivamente.

Ao final da simulação, com 1.800 segundos, o PRIOBILL confirmou o melhor desempenho, alcançando média de 130,9 tarefas processadas. O AHP, o GWO, o ALEATÓRIO, o WOA, o PSO e o GULOSO processaram, em média, 102,5, 99,6, 89,3, 83,0, 81,6 e 61,2 tarefas, respectivamente.

Novamente, todos os mecanismos enfrentaram dificuldades para atender às solicitações devido à elevada demanda de recursos exigida pelas tarefas maiores. Ainda assim, o PRIOBILL demonstrou maior capacidade de adaptação a cenários desafiadores, mantendo o melhor desempenho e processando o maior número de tarefas ao longo de

toda a simulação.

Figura 37 – Número de tarefas pesadas processadas no horário de pico.



Fonte: Gerado pelo autor.

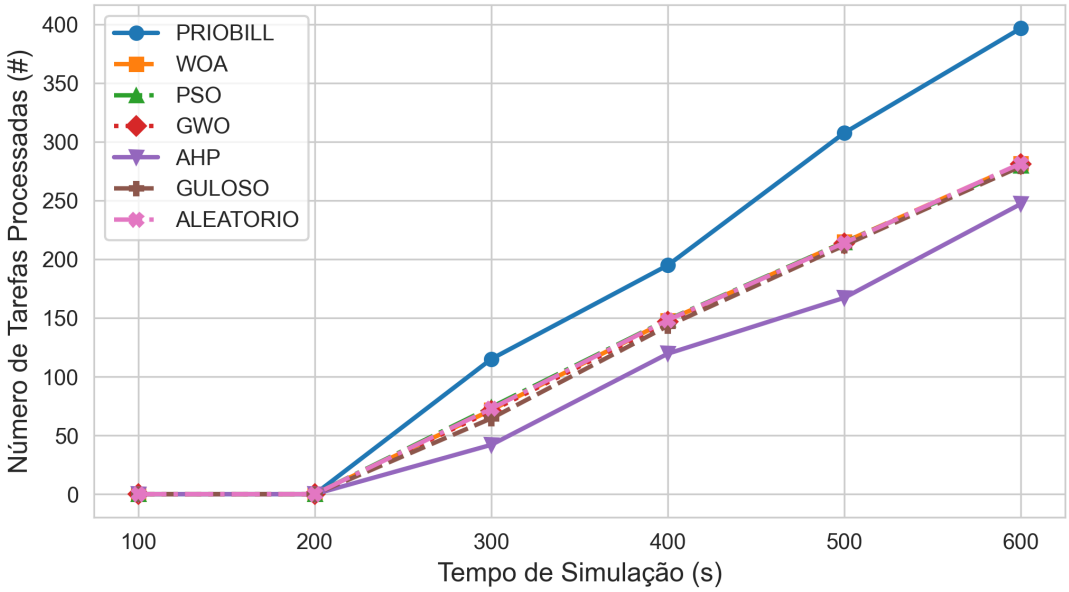
Na Figura 38 é apresentado o número de tarefas com pesos aleatórios processadas no cenário da madrugada. Nos primeiros 200 segundos de simulação, nenhum mecanismo conseguiu processar tarefas, devido à baixa quantidade de veículos no cenário. Aos 300 segundos de simulação, o PRIOBILL processou, em média, 115 tarefas, desempenho 55,36% superior ao do PSO, que apresentou o segundo melhor resultado. O ALEATÓRIO processou 57,88% menos tarefas, enquanto o WOA processou 60,69% menos. O GWO apresentou 62,26% menos tarefas processadas, o GULOSO 78,35% menos e o AHP 173,95% menos.

Aos 400 segundos de simulação, o PRIOBILL atingiu média de 194,9 tarefas processadas. O PSO, o ALEATÓRIO e o WOA apresentaram desempenhos semelhantes, processando 31,10%, 31,45% e 31,75% menos tarefas que o PRIOBILL, respectivamente. O GWO processou 32,89% menos tarefas, enquanto o GULOSO e o AHP apresentaram reduções de 36,01% e 62,96%, respectivamente.

Aos 500 segundos de simulação, o PRIOBILL manteve o melhor desempenho, com média de 307,63 tarefas processadas, superando o WOA em 43,31%, o PSO em 43,41%, o ALEATÓRIO em 43,76%, o GWO em 44,04%, o GULOSO em 45,22% e o AHP em 83,98%.

Com 600 segundos de simulação, o PRIOBILL novamente apresentou o melhor desempenho, alcançando média de 396,7 tarefas processadas. O ALEATÓRIO, o WOA, o GWO, o PSO, o GULOSO e o AHP processaram 40,87%, 41,05%, 41,22%, 41,75%, 42,14% e 60,51% menos tarefas, respectivamente.

Figura 38 – Número de tarefas com pesos aleatórios processadas no horário da madrugada.



Fonte: Gerado pelo autor.

O número de tarefas com pesos aleatórios processadas no horário da manhã é apresentado na Figura 39. Aos 700 segundos de simulação, o PRIOBILL processou, em média, 473 tarefas. O GULOSO processou 347,4 tarefas, seguido de perto pelo GWO, com 346,9, pelo WOA, com 345,4, pelo ALEATÓRIO, com 344,2, pelo PSO, com 343,9, e pelo AHP, com 347,3 tarefas processadas.

Ao atingir 800 segundos de simulação, o PRIOBILL manteve o melhor desempenho, com média de 555,5 tarefas processadas. O AHP apresentou o segundo melhor resultado, com 479,2 tarefas processadas. O GWO, o WOA, o PSO e o ALEATÓRIO obtiveram desempenhos próximos, com médias de 425,5, 423,9, 423,3 e 422,5 tarefas processadas, respectivamente. O GULOSO processou apenas 394,5 tarefas.

Aos 900 segundos, o PRIOBILL processou, em média, 598,2 tarefas. O AHP apresentou o segundo melhor desempenho, com 597,7 tarefas processadas. O GWO atingiu média de 493,1 tarefas, enquanto o ALEATÓRIO, o WOA, o PSO e o GULOSO processaram, em média, 488,7, 487,8, 486,2 e 453,4 tarefas, respectivamente.

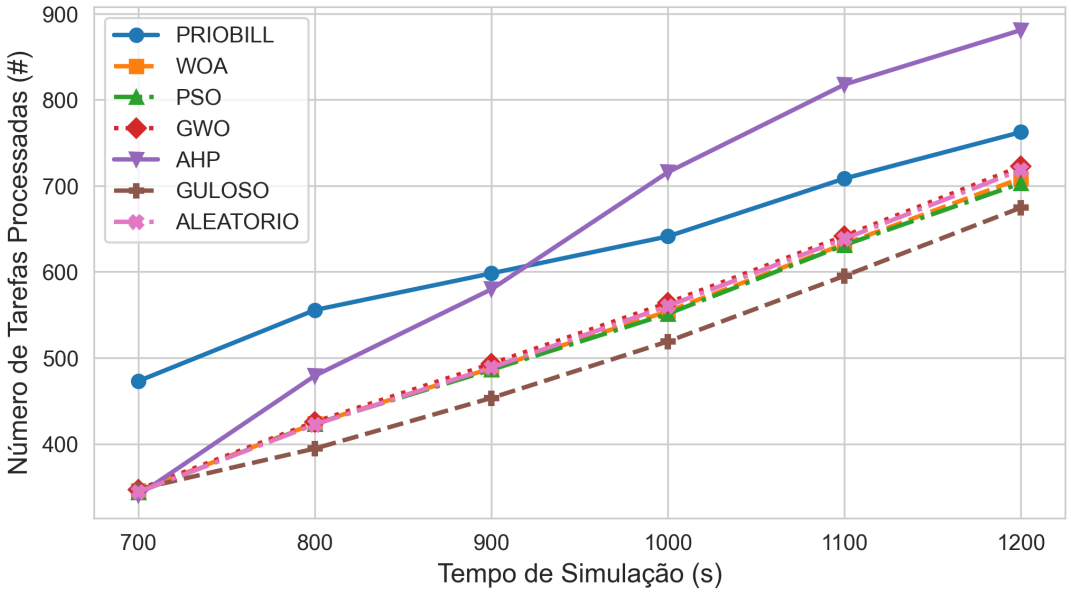
Com 1.000 segundos de simulação, o AHP e o PRIOBILL apresentaram os melhores desempenhos, com médias de 715,7 e 641,1 tarefas processadas, respectivamente. O GWO alcançou média de 564 tarefas processadas, seguido pelo ALEATÓRIO, com 560,2. O WOA, o PSO e o GULOSO processaram, em média, 554,0, 551,2 e 519 tarefas, respectivamente.

Aos 1.100 segundos, o AHP e o PRIOBILL mantiveram os melhores desempenhos, com médias de 817,6 e 708,3 tarefas processadas, respectivamente. O GWO apre-

sentou o terceiro melhor resultado, com 641,6 tarefas processadas, seguido de perto pelo ALEATÓRIO, com 638,3, pelo WOA, com 632,3, e pelo PSO, com 631,1. O GULOSO apresentou o menor desempenho, com média de 595,2 tarefas processadas.

Com 1.200 segundos de simulação, o AHP obteve o melhor desempenho, alcançando média de 880,8 tarefas processadas. O PRIOBILL ficou na segunda colocação, com média de 762,3 tarefas processadas, seguido pelo GWO, com 722,4, pelo ALEATÓRIO, com 718,4, pelo WOA, com 708,7, pelo PSO, com 703,0, e pelo GULOSO, com 674,5 tarefas processadas.

Figura 39 – Número de tarefas com pesos aleatórios processadas no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 40 apresenta-se o número de tarefas com pesos aleatórios no horário de pico. Com 1.300 segundos de simulação, o AHP e o PRIOBILL apresentaram os melhores desempenhos, com 955,7 e 820,5 tarefas processadas em média, respectivamente. O ALEATÓRIO e o GWO processaram 817 tarefas. O WOA apresentou a média de 804,3 tarefas processadas, enquanto o PSO e o GULOSO processaram 796,2 e 750,2 tarefas, respectivamente.

Aos 1.400 segundos, o AHP processou 1.001,3 tarefas. Em seguida, o ALEATÓRIO apresentou 905 tarefas e o GWO 903,5. O PSO, WOA, PRIOBILL e GULOSO processaram 887,5, 881,2, 860,5 e 813 tarefas, respectivamente.

Com 1.500 segundos de simulação, o AHP processou 1.076 tarefas, seguido de perto pelo ALEATÓRIO com 1.003,4 tarefas. O GWO, WOA, PSO, PRIOBILL e GULOSO processaram 998,3, 979,9, 969,2, 926,3 e 878,9 tarefas, respectivamente.

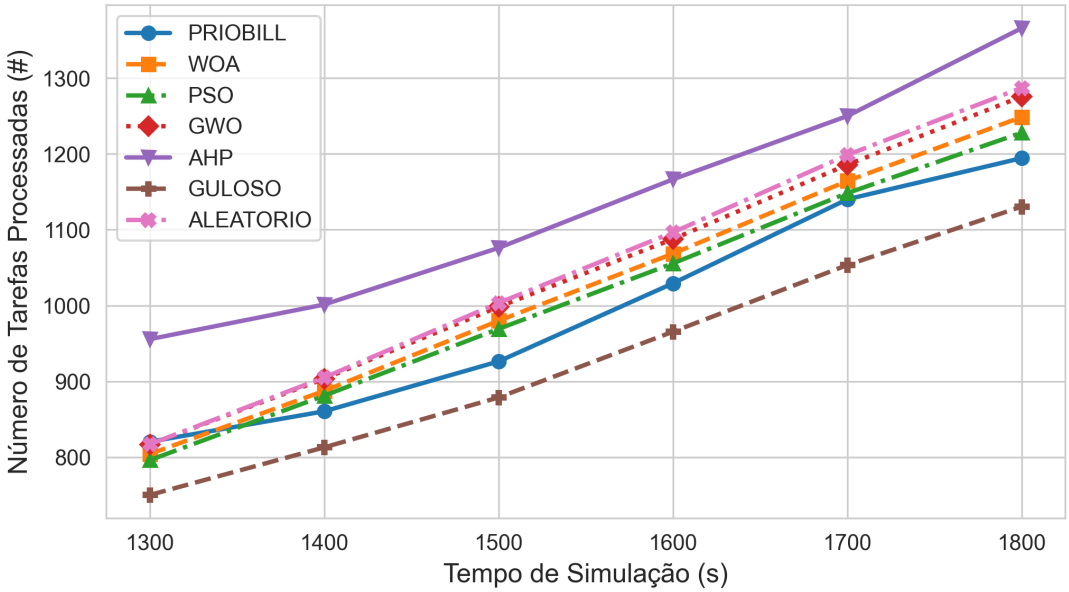
Com 1.600 segundos, o AHP processou 1.166,4 tarefas. O ALEATÓRIO processou 1.096,5 tarefas e o GWO processou 1.087,6 tarefas. Enquanto isso, o WOA, PSO,

PRIOBILL e GULOSO processaram em média 1.068,4, 1.055,6, 1.029,3 e 965,4 tarefas, respectivamente.

Com 1.700 segundos de simulação, o AHP atingiu 1.249,9 tarefas processadas. O ALEATÓRIO, GWO, WOA, PSO e PRIOBILL apresentaram resultados próximos, com 1.198,5, 1.185,6, 1.164,2, 1.148,2 e 1.140,1 tarefas processadas, respectivamente. O GULOSO apresentou um desempenho inferior, com 1.053,4 tarefas processadas.

No último intervalo, aos 1.800 segundos, o AHP alcançou 1.365,7 tarefas processadas. O ALEATÓRIO apresentou 1.287,1 tarefas, seguido pelo GWO com 1.275,5. O WOA, PSO e PRIOBILL processaram 1.248,5, 1.227,9 e 1.194,4 tarefas, respectivamente. Enquanto o GULOSO processou 1.130,4 tarefas.

Figura 40 – Número de tarefas com pesos aleatórios processadas no horário de pico.



Fonte: Gerado pelo autor.

O mecanismo desenvolvido demonstrou bom desempenho na métrica de tarefas processadas, que corresponde às tarefas atendidas integralmente durante as simulações. Em resumo, o PRIOBILL apresentou o melhor desempenho no processamento de tarefas menores no cenário da madrugada. No horário da manhã, o PRIOBILL manteve o bom desempenho nos primeiros intervalos e, posteriormente, permaneceu próximo ao algoritmo com melhor desempenho. A performance foi mantida durante o terceiro horário de simulação, até que o PRIOBILL alcançou o melhor aproveitamento ao final da simulação.

No cenário com tarefas intermediárias, o PRIOBILL apresentou o melhor desempenho em praticamente todos os intervalos no horário da madrugada, em todos os intervalos no horário da manhã e obteve o melhor resultado no primeiro intervalo do horário de pico. Nos três intervalos seguintes, apresentou algumas oscilações e voltou a alcançar o melhor desempenho nos dois últimos intervalos.

No cenário com tarefas maiores, o PRIOBILL apresentou desempenho superior aos demais mecanismos no cenário da madrugada. Posteriormente, todos os mecanismos apresentaram dificuldades de atendimento devido ao maior tamanho das tarefas. O PRIOBILL conseguiu atender mais tarefas em grande parte das simulações, mesmo em cenários mais desafiadores. Esse comportamento está relacionado ao processo de caça e adaptação às mudanças de topologia da rede.

O *framework* demonstra eficiência ao lidar com cenários mais complexos e com menor disponibilidade de recursos, por meio de uma busca mais precisa pelo nó de borda adequado, baseada no cálculo de *fitness* aplicado, possibilitando, em muitos casos, encontrar soluções próximas ao ótimo global. Mesmo quando não apresentou o maior número de tarefas processadas em determinado intervalo da simulação, o PRIOBILL permaneceu próximo ao melhor resultado, demonstrando que o processo de movimentação e caça implementado na alocação consegue manter sua competitividade, mesmo diante das limitações impostas pelos cenários avaliados.

4.10.2 Tarefas não processadas

Nesta seção são apresentados os resultados referentes às tarefas não processadas para todos os cenários avaliados.

Na Figura 41 é apresentado o número de tarefas menores não processadas no horário da madrugada. Nos primeiros 200 segundos de simulação realista durante a madrugada, a baixa quantidade de veículos, que não foi suficiente para formar clusters, impossibilitou a ativação do processo de solicitação de tarefas.

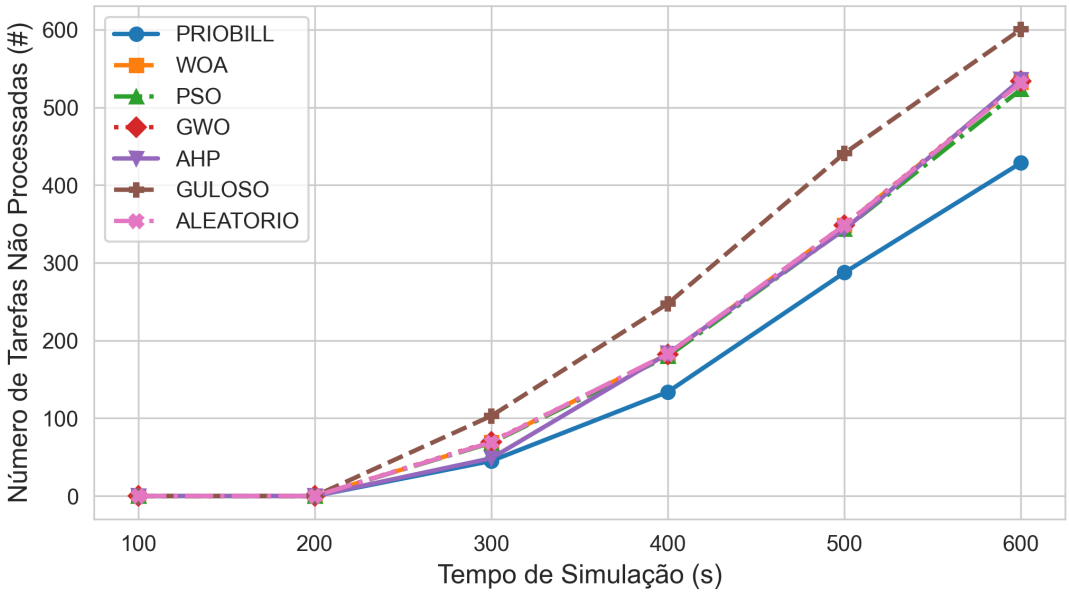
Aos 300 segundos de simulação, o PRIOBILL apresentou o menor número de negações, com uma média de 45 tarefas negadas. O AHP apresentou a segunda melhor média, com 48 negações. O PSO e o WOA tiveram média de 68 negações, enquanto o ALEATÓRIO e o GWO apresentaram média de 69. O GULOSO apresentou 103 negações.

Com 400 segundos de simulação, o PRIOBILL negou 134 tarefas, seguido pelo PSO com 179 negações, pelo WOA, GWO e ALEATÓRIO com 182 negações, pelo AHP com 183 e pelo GULOSO com 247 tarefas não processadas.

Aos 500 segundos, a média do PRIOBILL foi de 287 tarefas não processadas. O AHP e o PSO não processaram, em média, 343 tarefas, enquanto o WOA, ALEATÓRIO e GWO apresentaram média de 348 tarefas não processadas. O GULOSO apresentou 440 tarefas não processadas.

Com 600 segundos de simulação, o PRIOBILL apresentou a menor média de negações, com 428 tarefas não processadas, seguido pelo PSO, ALEATÓRIO, WOA, GWO, AHP e GULOSO, com 523, 531, 532, 533, 536 e 600 negações, respectivamente.

Figura 41 – Número de tarefas menores não processadas no horário da madrugada.



Fonte: Gerado pelo autor.

O número de tarefas menores não processadas no horário da manhã é apresentado na Figura 42. Com 700 segundos de simulação, o PRIOBILL apresentou o menor número de negações, com média de 555 tarefas não processadas. O PSO apresentou 685 negações. O WOA e o GWO tiveram média de 700 tarefas não processadas, seguidos de perto pelo ALEATÓRIO, com 705 tarefas não processadas. O AHP apresentou 738 negações, enquanto o GULOSO teve 750 tarefas não processadas.

Aos 800 segundos, o PRIOBILL atingiu 705 tarefas não processadas. O GULOSO apresentou 871 negações, seguido pelo AHP e PSO, com 756 e 876 tarefas não processadas, respectivamente. O GWO, com 891, o WOA, com 892, e o ALEATÓRIO, com 899 tarefas não processadas, completam a sequência.

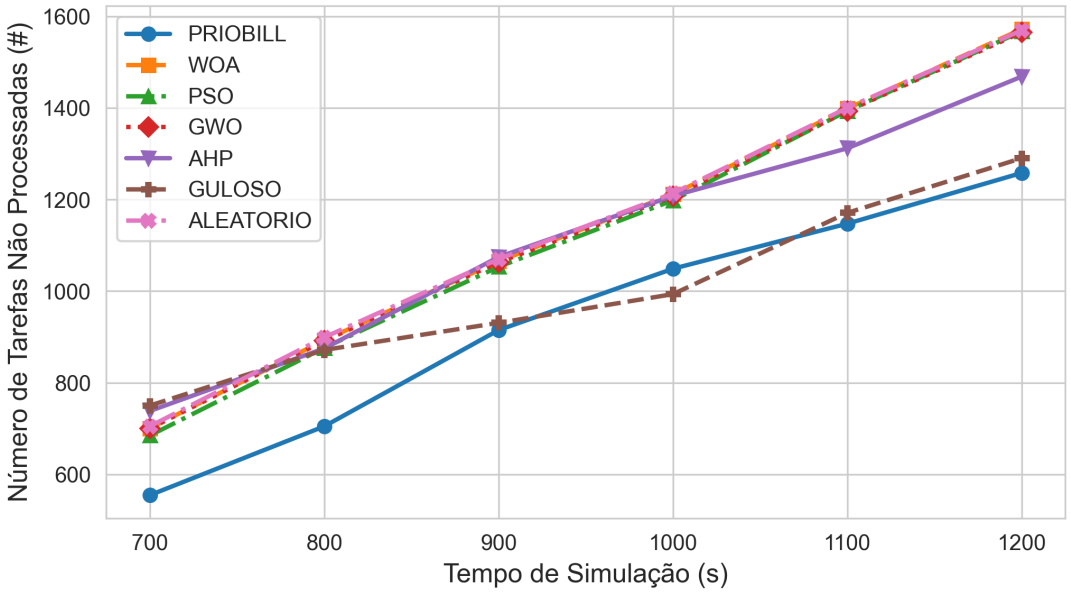
Com 900 segundos de simulação, o PRIOBILL apresentou 915 tarefas não processadas, seguido pelo GULOSO, com média de 930 negações. O PSO, GWO, WOA, ALEATÓRIO e AHP apresentaram 1.053, 1.062, 1.065, 1.069 e 1.074 tarefas não processadas, respectivamente.

Aos 1.000 segundos de simulação, o GULOSO e o PRIOBILL apresentaram os melhores resultados, com 993 e 1.048 tarefas não processadas, respectivamente. O PSO apresentou 1.198 negações. O GWO e o AHP tiveram média de 1.207 tarefas não processadas, enquanto o WOA apresentou 1.211 e o ALEATÓRIO 1.213 negações.

Com 1.100 segundos, o PRIOBILL não conseguiu processar 1.147 tarefas. O GULOSO não conseguiu processar 1.170 tarefas. Enquanto isso, o AHP, GWO, PSO, WOA e ALEATÓRIO não conseguiram processar 1.311, 1.392, 1.393, 1.398 e 1.400 tarefas, respectivamente.

Aos 1.200 segundos, o PRIOBILL chegou a 1.257 tarefas não processadas. O GULOSO apresentou 1.290 tarefas não processadas e o AHP 1.468. Enquanto isso, o GWO, PSO, ALEATÓRIO e WOA apresentaram 1.565, 1.567, 1.569 e 1.571 tarefas não processadas, respectivamente.

Figura 42 – Número de tarefas menores não processadas no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 43 apresenta-se o número de tarefas menores não processadas no cenário de pico. Aos 1.300 segundos de simulação, o PRIOBILL apresentou, em média, 1.515 tarefas não processadas. O GULOSO apresentou 1.537 tarefas não processadas. O AHP negou 1.686 tarefas. O PSO, GWO e ALEATÓRIO tiveram, em média, 1.786 tarefas não processadas, enquanto o WOA apresentou 1.794.

Com 1.400 segundos de simulação, o GULOSO e o PRIOBILL apresentaram 1.727 e 1.736 tarefas não processadas, respectivamente. Em seguida, ficaram o AHP, com 1.861, o ALEATÓRIO, com 1.954, o GWO, com 1.955, o PSO, com 1.960, e o WOA, com 1.969 tarefas não processadas.

Aos 1.500 segundos, o GULOSO e o PRIOBILL negaram 1.901 e 1.912 tarefas, respectivamente. Em seguida, ficaram o AHP, GWO e ALEATÓRIO, com 2.120, 2.137 e 2.139 tarefas não processadas, respectivamente. O WOA e o PSO apresentaram 2.146 tarefas não processadas.

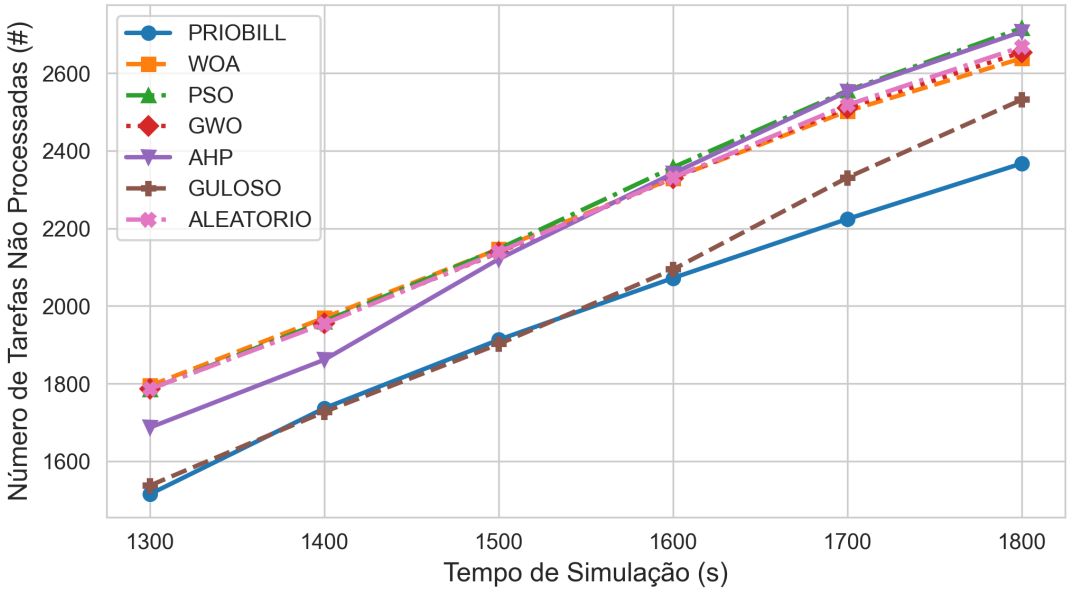
Aos 1.600 segundos de simulação, o PRIOBILL negou, em média, 2.071 tarefas, seguido pelo GULOSO com 2.094 negações. O GWO e o WOA apresentaram 2.329 tarefas não processadas, enquanto o ALEATÓRIO, AHP e PSO apresentaram 2.330, 2.341 e 2.357 tarefas não processadas, respectivamente.

Aos 1.700 segundos, o PRIOBILL apresentou 2.224 tarefas não processadas. O

GULOSO apresentou 2.330 tarefas não processadas. O WOA atingiu 2.501 tarefas não processadas, enquanto o GWO, ALEATÓRIO, AHP e PSO apresentaram 2.510, 2.517, 2.552 e 2.554 tarefas não processadas, respectivamente.

Aos 1.800 segundos de simulação, o PRIOBILL novamente apresentou o menor número de tarefas não processadas, com média de 2.367, seguido pelo GULOSO com 2.531. Em terceiro ficou o WOA, com 2.638 tarefas não processadas. O GWO apresentou 2.653, o ALEATÓRIO 2.667, o AHP 2.706 e o PSO 2.715 tarefas não processadas.

Figura 43 – Número de tarefas menores não processadas no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 44 é apresentado o número de tarefas intermediárias não processadas no horário da madrugada. Nos primeiros 200 segundos de simulação, não foram observadas requisições devido ao baixo número de veículos presentes no cenário.

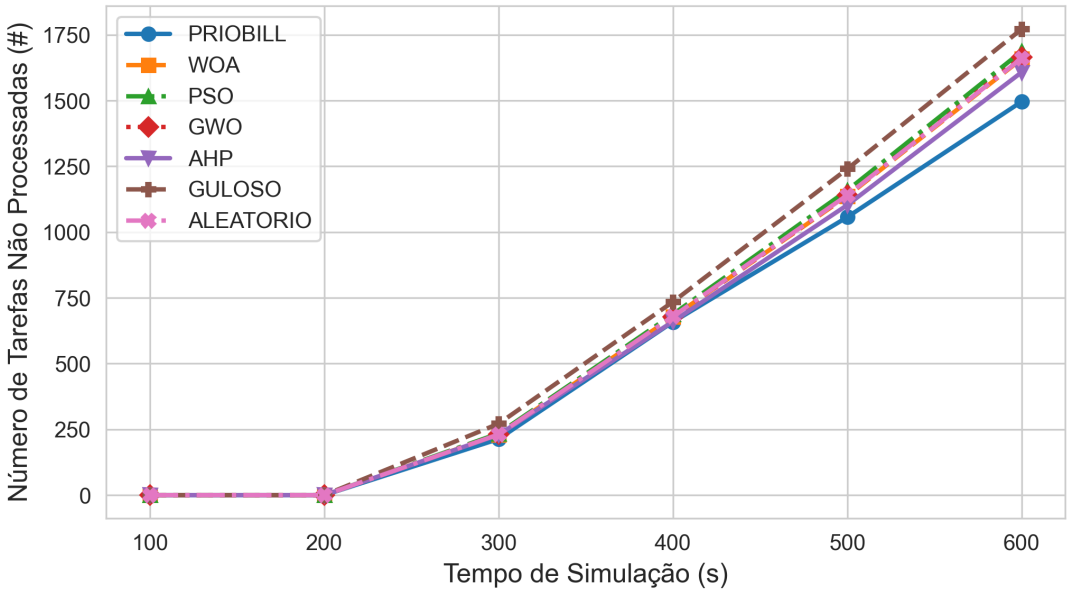
Com 300 segundos de simulação, o PRIOBILL apresentou o menor número de tarefas não processadas, com média de 213. O AHP apresentou 229 tarefas não processadas, enquanto o WOA, GWO e ALEATÓRIO apresentaram 230 tarefas não processadas. O PSO, com 324, e o GULOSO, com 271 tarefas não processadas, apresentaram os piores desempenhos.

Aos 400 segundos de simulação, o PRIOBILL apresentou 658 tarefas não processadas. O AHP apresentou resultado próximo, com 659 tarefas não processadas. O WOA, GWO e ALEATÓRIO apresentaram 677, 678 e 679 tarefas não processadas, respectivamente. Enquanto isso, o PSO apresentou 687 e o GULOSO 734 tarefas não processadas.

Com 500 segundos de simulação, o PRIOBILL atingiu 1.056 tarefas não processadas. Em seguida, ficaram o AHP, WOA, GWO, ALEATÓRIO, PSO e GULOSO, com 1.102, 1.136, 1.139, 1.140, 1.158 e 1.240 tarefas não processadas, respectivamente.

Aos 600 segundos, o PRIOBILL negou 1.495 tarefas. O AHP apresentou o segundo menor número de negações, com 1.607. O WOA apresentou 1.658 tarefas não processadas, enquanto o ALEATÓRIO apresentou 1.660. O GWO, PSO e GULOSO apresentaram 1.665, 1.686 e 1.771 tarefas não processadas, respectivamente.

Figura 44 – Número de tarefas intermediárias não processadas no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 45 apresenta-se o número de tarefas intermediárias não processadas no horário da manhã. Com 700 segundos de simulação, o PRIOBILL não processou 1.933 tarefas. O AHP deixou de processar 2.129, enquanto o GWO, WOA e ALEATÓRIO não processaram 2.150, 2.152 e 2.155 tarefas, respectivamente. O PSO não processou 2.177 tarefas e o GULOSO deixou de processar 2.216 tarefas.

Aos 800 segundos, novamente o PRIOBILL apresentou o menor número de tarefas não processadas, com 2.327. Em seguida, ficaram o WOA e o GWO, com 2.629 tarefas não processadas. O ALEATÓRIO deixou de processar 2.635 tarefas, enquanto o AHP, PSO e GULOSO não processaram 2.657, 2.660 e 2.709 tarefas, respectivamente.

Com 900 segundos de simulação, o PRIOBILL apresentou média de 2.826 tarefas não processadas. O GWO apresentou 3.087, o WOA 3.089 e o ALEATÓRIO 3.094 tarefas não processadas. O PSO, GULOSO e AHP apresentaram 3.119, 3.123 e 3.136 tarefas não processadas, respectivamente.

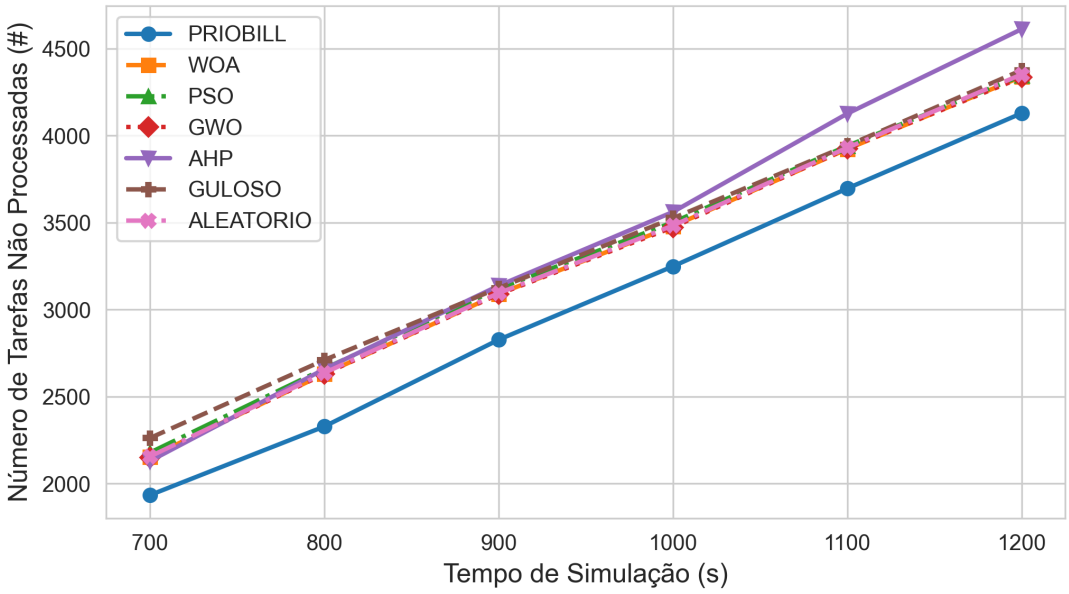
Com 1.000 segundos, o PRIOBILL continuou apresentando o menor número de negações, com 3.248 tarefas. Em seguida, ficaram o GWO, com 3.472, o WOA, com 3.478, o ALEATÓRIO, com 3.482, o PSO, com 3.501, o GULOSO, com 3.524, e o AHP, com 3.560 tarefas não processadas.

Aos 1.100 segundos de simulação, o PRIOBILL negou 3.695 tarefas, seguido pelo

WOA, GWO, ALEATÓRIO, PSO, GULOSO e AHP, com 3.922, 3.925, 3.930, 3.940, 3.941 e 4.125 tarefas negadas, respectivamente.

Com 1.200 segundos, novamente o PRIOBILL apresentou o menor número de tarefas negadas, com 4.128. O GWO apresentou 4.333 tarefas negadas, seguido pelo WOA com 4.337. Enquanto isso, o PSO, ALEATÓRIO, GULOSO e AHP apresentaram 4.343, 4.351, 4.375 e 4.611 tarefas negadas, respectivamente.

Figura 45 – Número de tarefas intermediárias não processadas no horário da manhã.

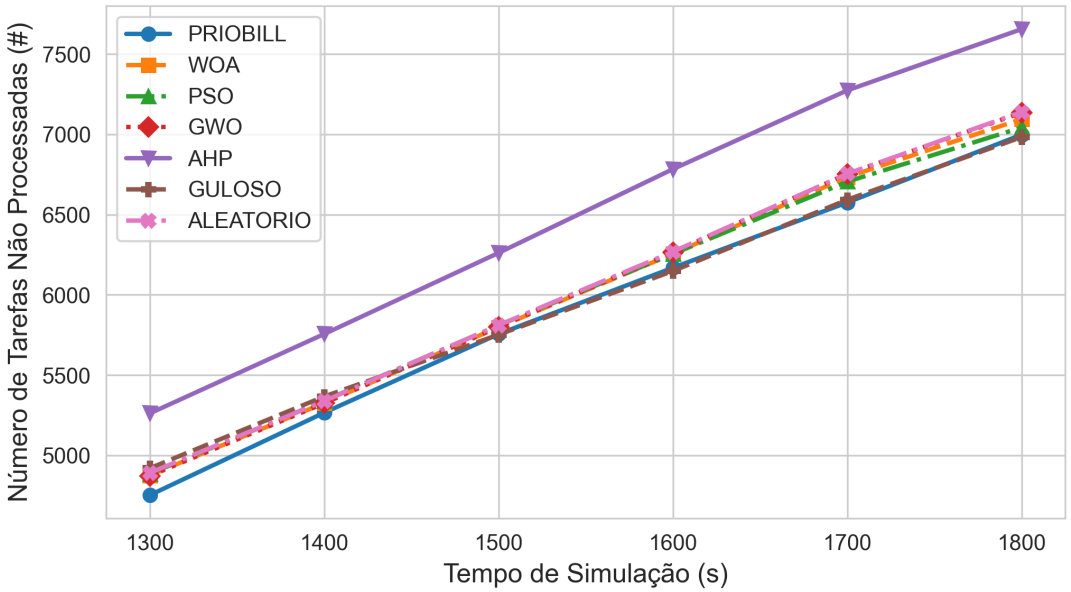


Fonte: Gerado pelo autor.

Na Figura 46 é apresentado o número de tarefas intermediárias não processadas no horário de pico. Aos 1.300 segundos de simulação, o PRIOBILL teve 4.752 tarefas não processadas. O GWO teve 4.872 tarefas não processadas e o WOA teve 4.875 tarefas não processadas. Enquanto PSO, ALEATÓRIO, GULOSO e AHP tiveram 4.885, 4.890, 4.919 e 5.261 tarefas não processadas, respectivamente. Com 1.400 segundos de simulação, o PRIOBILL chegou a 5.265 tarefas não processadas, enquanto GWO, WOA, ALEATÓRIO, PSO, GULOSO e AHP não conseguiram processar 5.324, 5.326, 5.338, 5.340, 5.366 e 5.756 tarefas, respectivamente. Aos 1.500 segundos, o GULOSO e o PRIOBILL tiveram 5.751 e 5.755 tarefas não processadas, respectivamente. O WOA não processou 5.798 tarefas, enquanto o GWO não processou 5.800 tarefas. ALEATÓRIO deixou de processar 5.810 tarefas e o AHP não processou 6.261 tarefas. Com 1.600 segundos, GULOSO e PRIOBILL não processaram 6.148 e 6.168 tarefas, respectivamente. Enquanto PSO, WOA, GWO, ALEATÓRIO e AHP tiveram 6.254, 6.255, 6.264, 6.268 e 6.783 tarefas não processadas, respectivamente. Com 1.800 segundos de simulação, novamente GULOSO e PRIOBILL deixaram de processar menos tarefas, com 6.982 e 6.997 tarefas não processadas, respectivamente. O PSO teve 7.042 tarefas não processadas. O

WOA teve 7.095 tarefas não processadas. Enquanto GWO, ALEATÓRIO e AHP tiveram 7.134, 7.139 e 7.655 tarefas não processadas, respectivamente.

Figura 46 – Número de tarefas intermediárias não processadas no horário de pico.

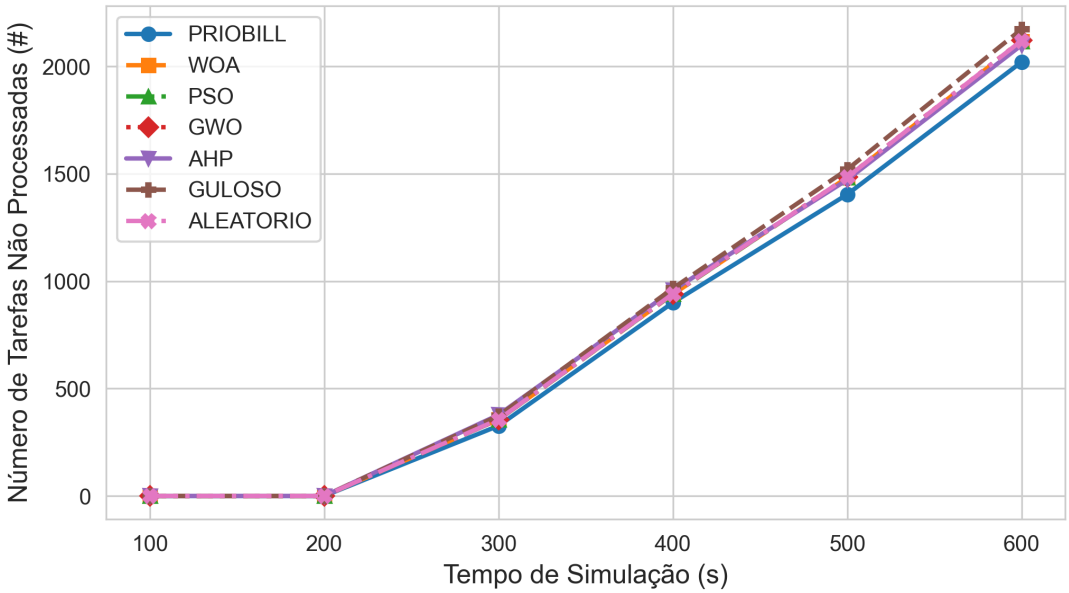


Fonte: Gerado pelo autor.

Na Figura 47 é apresentado o número de tarefas maiores não processadas no horário da madrugada. Nos primeiros 200 segundos de simulação, a pouca quantidade de veículos no cenário não possibilitou a formação de *clusters* e, consequentemente, a requisição de processamento de tarefas. Com 300 segundos de simulação, o PRIOBILL negou 326 tarefas. O PSO negou 355, enquanto WOA, GWO e ALEATÓRIO negaram, em média, 356 tarefas. O GULOSO negou 373 tarefas e o AHP negou 377. Aos 400 segundos, o PRIOBILL deixou de processar 900 tarefas. O WOA deixou de processar 938 e o PSO 939 tarefas. GWO e ALEATÓRIO deixaram de processar, em média, 940 tarefas, enquanto AHP e GULOSO não processaram 957 e 966 tarefas, respectivamente. Com 500 segundos de simulação, o PRIOBILL teve 1.404 tarefas não processadas. O AHP teve 1.472 tarefas não processadas. PSO e WOA tiveram 1.482 e 1.483 tarefas não processadas, respectivamente. ALEATÓRIO e GWO tiveram, em média, 1.484 tarefas não processadas. Enquanto o GULOSO teve 1.520 tarefas não processadas. Com 600 segundos, o PRIOBILL teve 2.020 tarefas não processadas. O AHP teve 2.099 tarefas não processadas. O PSO não processou 2.116 tarefas, enquanto o WOA não processou 2.117. ALEATÓRIO e GWO tiveram 2.120 tarefas não processadas, enquanto o GULOSO teve 2.173 tarefas não processadas.

Na Figura 48 apresenta-se o número de tarefas maiores não processadas no horário da manhã. Com 700 segundos de simulação, o PRIOBILL continuou negando menos processamentos, com 2.614 tarefas não processadas. O AHP negou 2.701 tarefas. WOA e PSO negaram, em média, 2.720 tarefas. O ALEATÓRIO negou o processamento de 2.723

Figura 47 – Número de tarefas maiores não processadas no horário da madrugada.

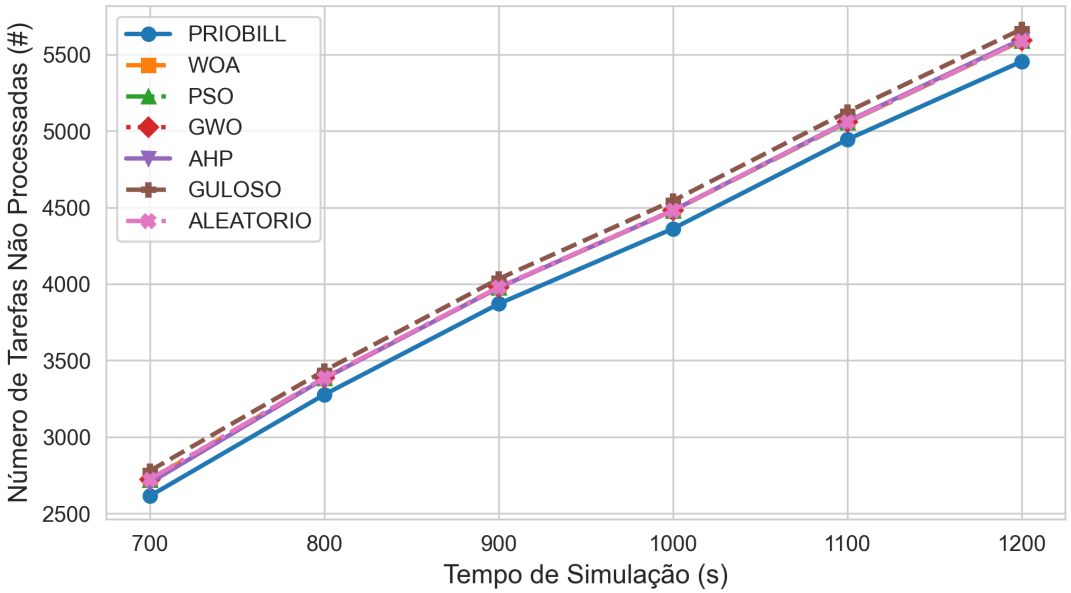


Fonte: Gerado pelo autor.

tarefas, enquanto o GWO negou 2.724 e o GULOSO negou 2.779. Aos 800 segundos, o PRIOBILL teve 3.277 tarefas não processadas. O AHP teve 3.381. WOA e PSO tiveram 3.384 tarefas não processadas, enquanto o GWO não processou 3.386 e o ALEATÓRIO não processou 3.387. O GULOSO teve 3.433 tarefas não processadas. Com 900 segundos, o PRIOBILL manteve o menor número de negações, com 3.870 tarefas não processadas. O AHP não processou 3.975 tarefas. WOA e PSO não processaram 3.976 tarefas. Seguidos de perto por GWO e ALEATÓRIO, com 3.978 e 3.979 tarefas não processadas, respectivamente. O GULOSO teve 4.033 tarefas não processadas. Ao atingir 1.000 segundos de simulação, o PRIOBILL negou 4.361 tarefas. O WOA negou 4.478 tarefas, enquanto PSO e AHP negaram 4.479 tarefas. GWO e ALEATÓRIO negaram, em média, 4.481 tarefas. O GULOSO negou 4.543 processamentos de tarefas. Com 1.100 segundos, o PRIOBILL teve 4.943 tarefas não processadas. O PSO não processou 5.055 tarefas, seguido de perto pelo WOA com 5.057, ALEATÓRIO com 5.058, GWO com 5.060 e AHP com 5.062 tarefas não processadas. O GULOSO negou 5.125 processamentos de tarefas. Aos 1.200 segundos, o PRIOBILL teve 5.453 tarefas não processadas. WOA, PSO e ALEATÓRIO tiveram 5.588 tarefas não processadas, enquanto GWO, AHP e GULOSO tiveram 5.591, 5.598 e 5.664 tarefas não processadas, respectivamente.

O número de tarefas maiores não processadas no horário de pico é apresentado na Figura 49. Aos 1.300 segundos, o PRIOBILL teve, em média, 6.150 tarefas não processadas. O WOA teve 6.286 tarefas não processadas. O ALEATÓRIO teve 6.289 tarefas não processadas, enquanto o PSO teve 6.290. O GWO teve, em média, 6.291 tarefas não processadas, enquanto o AHP teve 6.303 e o GULOSO teve 6.368. Com 1.400 segundos de simulação, o PRIOBILL teve 6.711 tarefas não processadas, seguido pelo WOA com

Figura 48 – Número de tarefas maiores não processadas no horário da manhã.

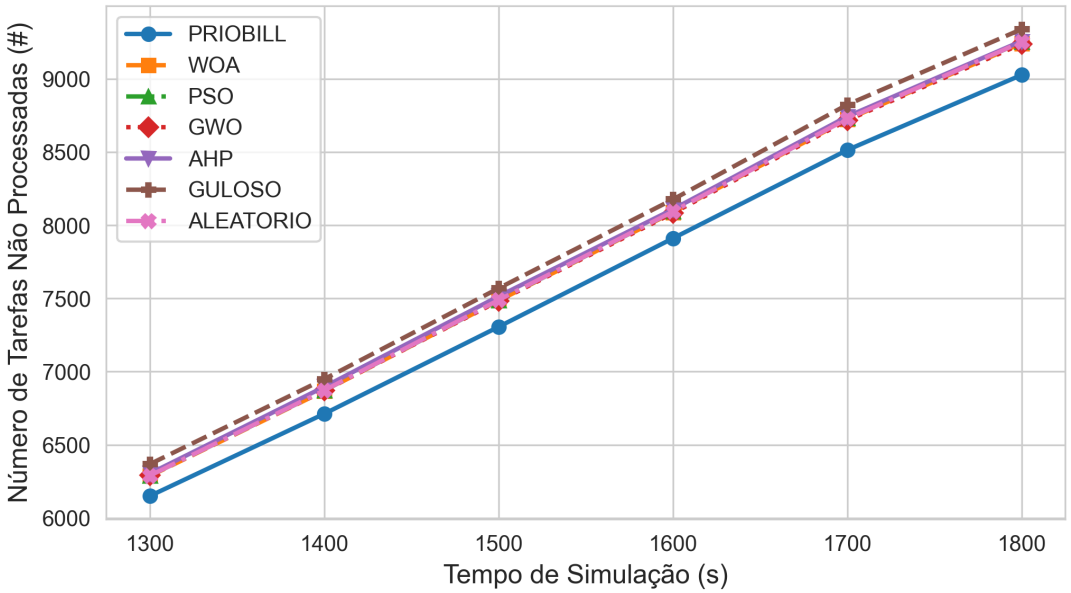


Fonte: Gerado pelo autor.

6.868 e pelo GWO com 6.870. ALEATÓRIO e PSO tiveram 6.871 tarefas não processadas. O AHP não processou 6.895 tarefas, enquanto o GULOSO deixou de processar 6.948 tarefas. Aos 1.500 segundos, o PRIOBILL negou 7.305 tarefas. Seguido por GWO, WOA, PSO, ALEATÓRIO, AHP e GULOSO, com 7.482, 7.484, 7.487, 7.488, 7.514 e 7.569 tarefas negadas, respectivamente. Aos 1.600 segundos de simulação, o PRIOBILL negou, em média, 7.910 tarefas, seguido pelo GWO com 8.083 e pelo WOA com 8.088 negações. PSO e ALEATÓRIO tiveram 8.093 tarefas não processadas, enquanto AHP e GULOSO tiveram 8.108 e 8.176, respectivamente. Aos 1.700 segundos, o PRIOBILL teve 8.513 tarefas não processadas. O GWO teve 8.717 tarefas não processadas. O WOA atingiu 8.722 tarefas não processadas. Enquanto ALEATÓRIO, PSO, AHP e GULOSO tiveram 8.727, 8.730, 8.742 e 8.823 tarefas não processadas, respectivamente. Aos 1.800 segundos de simulação, o PRIOBILL novamente negou menos tarefas, com a média de 9.026 tarefas não processadas, seguido pelo GWO com 9.238. Em terceiro ficou o WOA com 9.243 tarefas não processadas. Enquanto PSO e ALEATÓRIO tiveram 9.251 tarefas não processadas. O AHP teve 9.257 e o GULOSO teve 9.339 tarefas não processadas.

Na Figura 50 é apresentado o número de tarefas não processadas com pesos aleatórios no horário da madrugada. Nos primeiros 200 segundos de simulação, não houve requisições devido à baixa quantidade de veículos no cenário. Aos 300 segundos de simulação, o PRIOBILL teve o menor número de negações, com a média de 255 tarefas negadas. O PSO teve a segunda melhor média, com 295 negações, seguido pelo ALEATÓRIO com 297. GWO e WOA tiveram a média de 298 negações, enquanto GULOSO e AHP tiveram médias de 304 e 325 tarefas negadas, respectivamente. Com 400 segundos, o PRIOBILL negou 762 tarefas, seguido pelo PSO com 815 negações. O WOA

Figura 49 – Número de tarefas maiores não processadas no horário de pico.

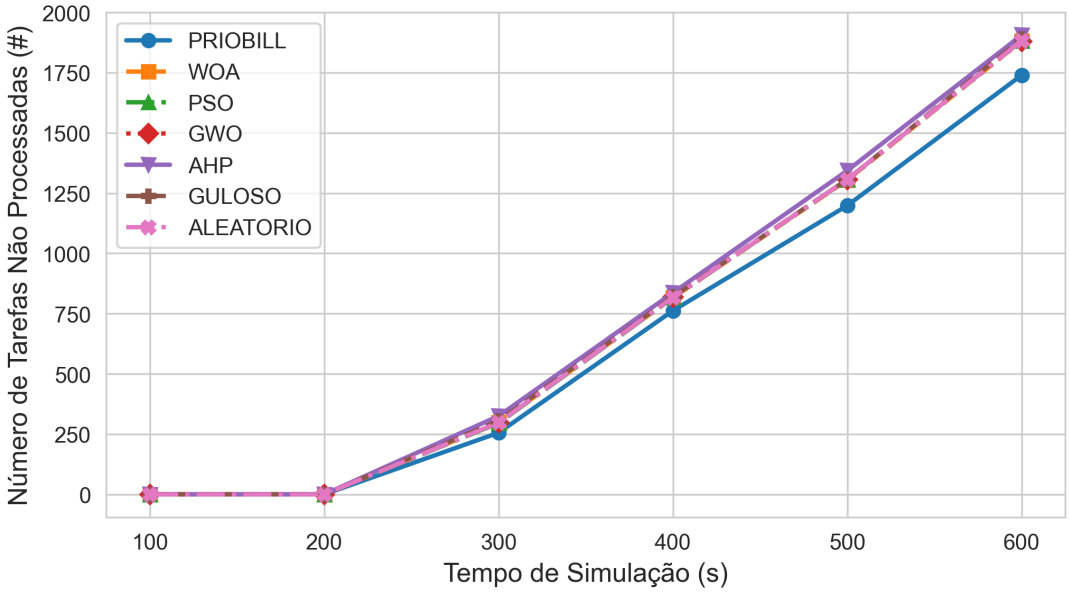


Fonte: Gerado pelo autor.

negou 816 tarefas, enquanto GWO e ALEATÓRIO negaram 817. O GULOSO teve 824 negações e o AHP teve 825 tarefas não processadas. Aos 500 segundos, a média do PRIOBILL foi de 1.198 tarefas não processadas. O WOA não processou 1.304 tarefas. PSO, GWO, GULOSO e ALEATÓRIO não processaram, em média, 1.306 tarefas, enquanto o AHP teve 1.343 tarefas não processadas. Com 600 segundos de simulação, o PRIOBILL teve a menor média de negações, com 1.739 tarefas não processadas, seguido pelo WOA e GWO com 1.880 negações. ALEATÓRIO, PSO, GULOSO e AHP tiveram 1.881, 1.882, 1.889 e 1.905 tarefas não processadas, respectivamente.

Na Figura 51 apresenta-se o número de tarefas com pesos aleatórios não processadas no horário da manhã. Com 700 segundos de simulação, o PRIOBILL continuou negando menos processamentos, com 2.259 tarefas não processadas. O AHP negou 2.415 tarefas. Seguido por GWO, WOA, PSO, ALEATÓRIO e GULOSO, com a média de 2.420, 2.421, 2.423, 2.424 e 2.433 tarefas não processadas, respectivamente. Aos 800 segundos, o PRIOBILL teve 2.839 tarefas não processadas. O AHP teve 2.929. O GWO não processou 3.008 tarefas, enquanto o WOA não processou 3.009 tarefas. ALEATÓRIO e PSO tiveram 3.012 tarefas não processadas. O GULOSO teve 3.052 tarefas não processadas. Com 900 segundos, o PRIOBILL continuou com menos tarefas recusadas, totalizando 3.389 tarefas não processadas. Seguido pelo AHP com 3.420 tarefas não processadas. GWO, WOA, PSO, ALEATÓRIO e GULOSO tiveram 3.533, 3.536, 3.538, 3.539 e 3.593 tarefas não processadas, respectivamente. Ao atingir 1.000 segundos de simulação, o AHP e o PRIOBILL tiveram 3.787 e 3.837 tarefas não processadas, respectivamente. O GWO não processou 3.966 tarefas. O ALEATÓRIO negou o processamento de 3.969 tarefas. Seguidos de perto por WOA e PSO, com 3.973 e 3.977 tarefas não processadas, respecti-

Figura 50 – Número de tarefas com pesos aleatórios não processadas no horário da madrugada.

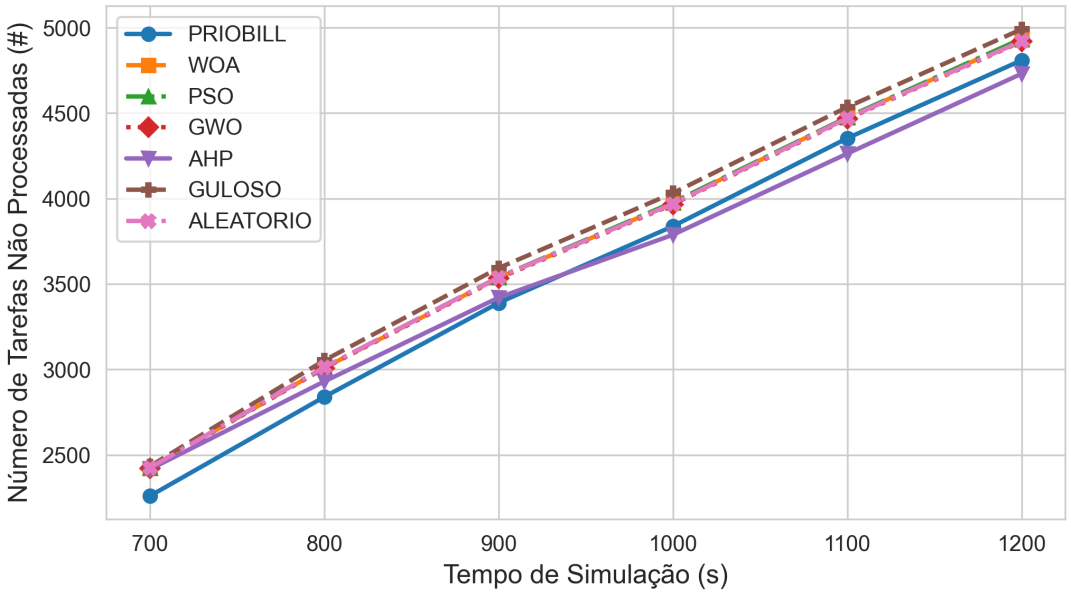


Fonte: Gerado pelo autor.

vamente. O GULOSO teve 4.032 tarefas não processadas. Ao atingir 1.100 segundos de simulação, novamente AHP e o PRIOBILL tiveram menos tarefas negadas, sendo 4.262 e 4.352, respectivamente. O GWO negou 4.466 tarefas, enquanto o ALEATÓRIO negou 4.470. O WOA não processou 4.474 tarefas. O PSO não conseguiu atender 4.475 tarefas. O GULOSO negou 4.535 processamentos de tarefas. Aos 1.200 segundos, o AHP e o PRIOBILL tiveram 4.730 e 4.809 tarefas não processadas, respectivamente. GWO, ALEATÓRIO, WOA, PSO e GULOSO tiveram 4.918, 4.921, 4.930, 4.935 e 4.990 tarefas não processadas, respectivamente.

O número de tarefas com pesos aleatórios não processadas no horário de pico é apresentado na Figura 52. Aos 1.300 segundos, o AHP teve, em média, 5.366 tarefas não processadas. O PRIOBILL teve 5.447 tarefas não processadas. O ALEATÓRIO teve 5.527 tarefas não processadas, enquanto o GWO teve 5.528. O WOA teve, em média, 5.537 tarefas não processadas, enquanto o PSO teve 5.545 e o GULOSO teve 5.626. Com 1.400 segundos de simulação, o AHP e o PRIOBILL tiveram 5.909 e 5.937 tarefas não processadas, respectivamente. Seguidos pelo ALEATÓRIO com 6.028 e pelo GWO com 6.029. WOA, PSO e GULOSO tiveram 6.039, 6.047 e 6.154 tarefas não processadas, respectivamente. Aos 1.500 segundos, o AHP negou 6.448 tarefas. Seguido pelo PRIOBILL, ALEATÓRIO, GWO, WOA, PSO e GULOSO, com 6.552, 6.567, 6.581, 6.713 tarefas negadas, respectivamente. Aos 1.600 segundos de simulação, o AHP negou, em média, 6.961 tarefas, seguido pelo PRIOBILL com 7.004, ALEATÓRIO com 7.069, GWO com 7.075 e WOA com 7.087 negações. PSO e GULOSO tiveram 7.104 e 7.236 tarefas não processadas, respectivamente. Aos 1.700 segundos, o PRIOBILL teve 7.501 tarefas não

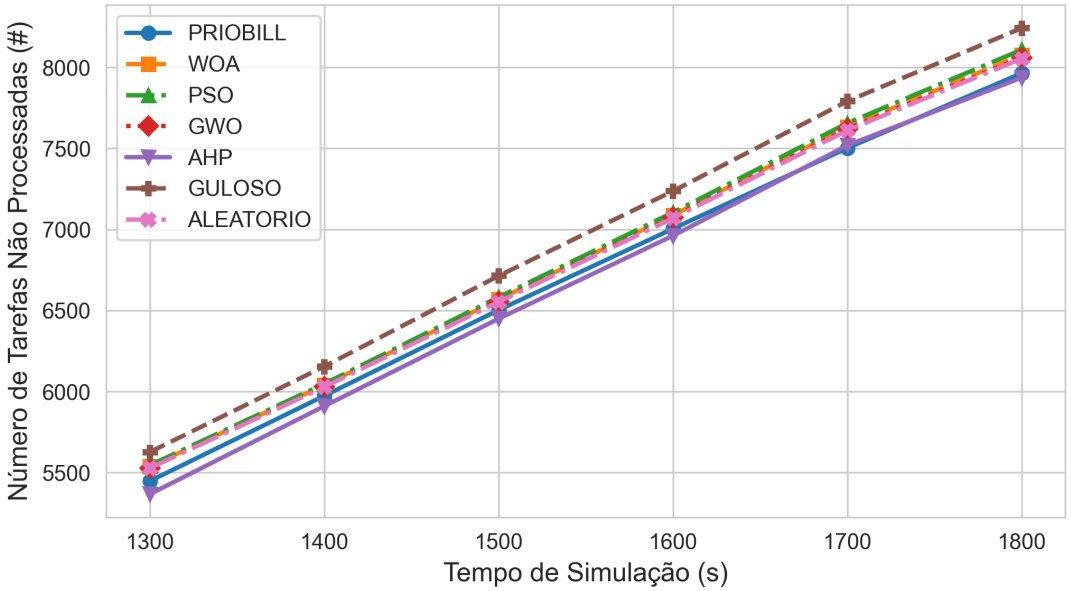
Figura 51 – Número de tarefas com pesos aleatórios não processadas no horário da manhã.



Fonte: Gerado pelo autor.

processadas. O AHP teve 7.518 tarefas não processadas. O ALEATÓRIO atingiu 7.610 tarefas não processadas. Enquanto GWO, WOA, PSO e GULOSO tiveram 7.620, 7.632, 7.655 e 7.790 tarefas não processadas, respectivamente. Aos 1.800 segundos de simulação, o AHP e o PRIOBILL negaram, em média, 7.936 e 7.963 tarefas, respectivamente. Seguidos por ALEATÓRIO, GWO, WOA, PSO e GULOSO, com 8.054, 8.060, 8.076, 8.107 e 8.241 tarefas não processadas, respectivamente.

Figura 52 – Número de tarefas com pesos aleatórios não processadas no horário de pico.



Fonte: Gerado pelo autor.

Na análise dos resultados de tarefas não processadas, que correspondem àquelas

que completaram todo o processo de seleção do nó de borda, mas não tiveram recursos suficientes para atender à solicitação, no cenário de tarefas menores, o PRIOBILL apresentou menos tarefas negadas no horário da madrugada e na maioria dos intervalos dos horários da manhã e de pico. Nas tarefas intermediárias, o melhor desempenho nos horários da madrugada, manhã e pico foi obtido pelo PRIOBILL. Para tarefas maiores, novamente o PRIOBILL demonstrou sua capacidade de adaptação ao cenário e obteve o melhor desempenho nos horários da madrugada e da manhã. No horário de pico, alternou o melhor desempenho, permanecendo pareado com os melhores algoritmos. No cenário com tarefas de pesos aleatórios, no horário da madrugada, o PRIOBILL apresentou o segundo melhor desempenho, permanecendo próximo aos melhores resultados. Nos horários da manhã e de pico, apresentou uma redução de desempenho em comparação aos demais algoritmos. O PRIOBILL demonstrou capacidade de minimizar as negações de processamento na maioria das métricas avaliadas, principalmente em cenários mais desafiadores. O PRIOBILL conseguiu negar menos tarefas em praticamente todos os cenários devido ao equilíbrio implementado com base no LOA para o processo de tomada de decisão, que avalia a capacidade dos nós de borda em relação à tarefa requisitante durante as iterações do mecanismo de decisão. Seu desempenho apresentou redução apenas nas simulações dos horários da manhã e de pico com tarefas de pesos aleatórios, devido às grandes diferenças entre os valores dos recursos exigidos por determinadas tarefas. De modo geral, o PRIOBILL consegue minimizar as negações mesmo em cenários mais complexos, com tarefas de maior demanda de recursos, devido à sua capacidade de adaptação e ao equilíbrio implementado com base no LOA para o processo de tomada de decisão, que avalia a capacidade dos nós de borda em relação à tarefa requisitante durante as iterações do mecanismo de decisão.

4.10.3 Quantidade de recursos utilizados

Nesta seção são apresentados os resultados da quantidade de recursos utilizados em todos os cenários avaliados.

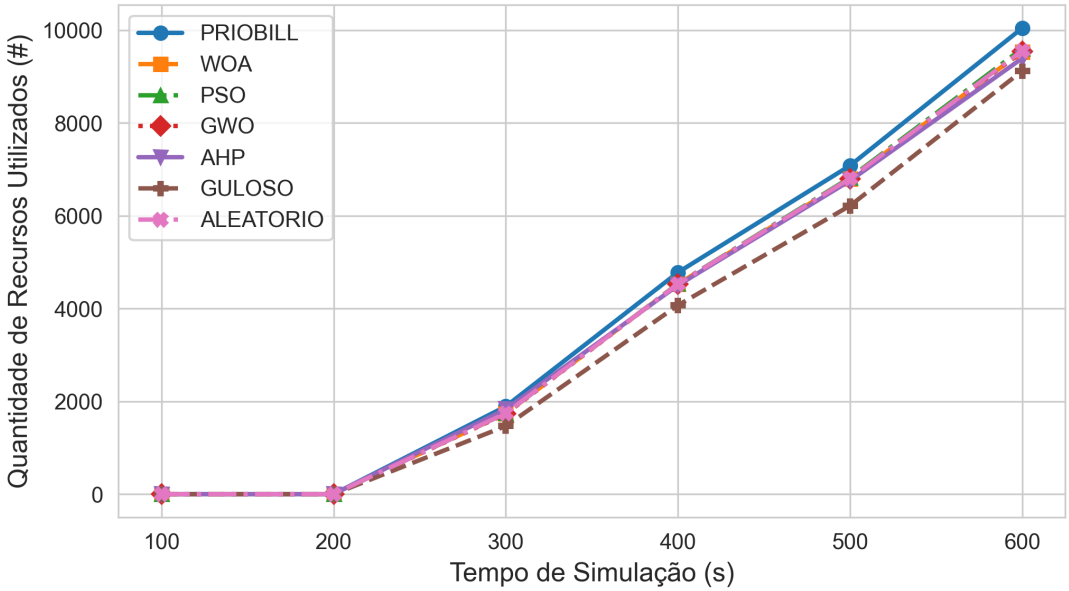
Na Figura 53, é apresentada a quantidade de recursos utilizados em tarefas menores no horário da madrugada. Durante os primeiros 200 segundos de simulação, nenhum dos algoritmos apresentou recursos utilizados. Aos 300 segundos, o PRIOBILL apresentou 1.891 recursos utilizados, seguido pelo AHP, com 1.831. O PSO registrou 1.744 recursos, enquanto o ALEATÓRIO utilizou 1.744. O WOA e o GWO apresentaram, respectivamente, 1.739 e 1.731 recursos. O GULOSO apresentou 1.467 recursos utilizados.

Aos 400 segundos, o PRIOBILL alcançou 4.778 recursos utilizados. O PSO apresentou 4.534, enquanto o ALEATÓRIO registrou 4.529. O WOA e o GWO atingiram 4.528 e 4.527, respectivamente. O AHP apresentou 4.501 recursos, enquanto o GULOSO registrou 4.070.

Aos 500 segundos, o PRIOBILL atingiu 7.082 recursos utilizados, seguido pelo PSO, com 6.820. O ALEATÓRIO apresentou 6.803, enquanto o WOA e o GWO registraram 6.795 e 6.793, respectivamente. O AHP alcançou 6.756 recursos, enquanto o GULOSO apresentou 6.216.

Ao final de 600 segundos, o PRIOBILL alcançou 10.044 recursos utilizados. O PSO apresentou 9.602, seguido pelo ALEATÓRIO, com 9.554. O GWO e o WOA registraram 9.537 e 9.536, respectivamente. O AHP atingiu 9.394 recursos, enquanto o GULOSO apresentou 9.121.

Figura 53 – Quantidade de recursos utilizados com tarefas menores no horário da madrugada.



Fonte: Gerado pelo autor.

O resultado da quantidade de recursos utilizados com tarefas menores no horário da manhã é apresentado na Figura 54. Aos 700 segundos de simulação, o PRIOBILL atingiu 12.736 recursos utilizados. O PSO apresentou 12.090, seguido pelo GWO, com 12.000. O WOA registrou 11.984, enquanto o ALEATÓRIO alcançou 11.967. O AHP apresentou 11.664 recursos, valor próximo ao GULOSO, que registrou 11.643.

Aos 800 segundos, o PRIOBILL alcançou 15.768 recursos utilizados. O GULOSO apresentou 14.960, seguido pelo PSO, com 14.856. O GWO registrou 14.784, enquanto o ALEATÓRIO e o WOA apresentaram 14.739 e 14.756, respectivamente. O AHP alcançou 14.729 recursos.

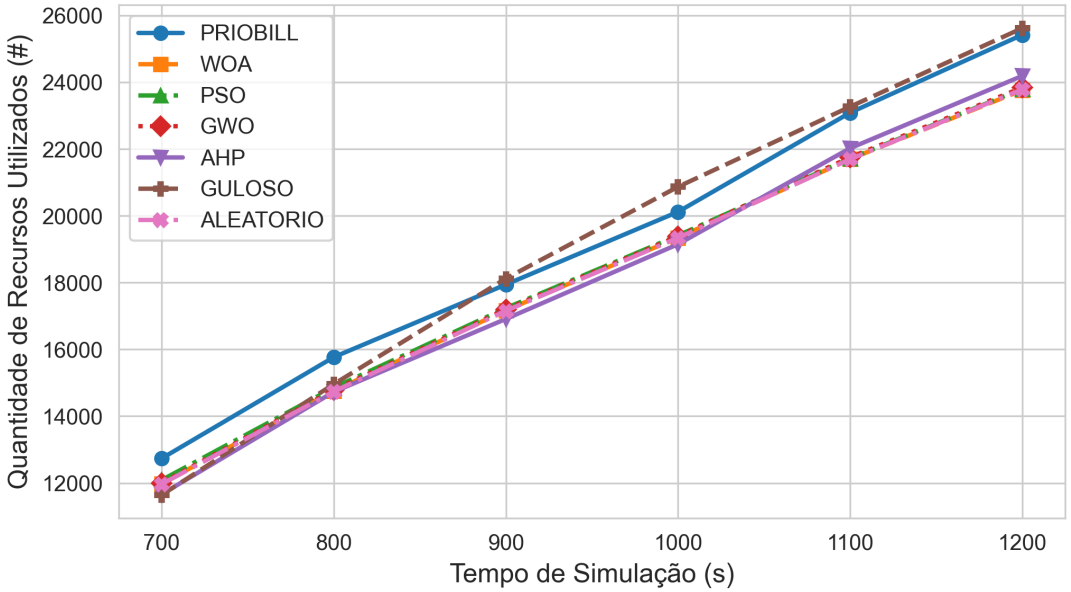
Aos 900 segundos, o PRIOBILL registrou 17.942 recursos utilizados. O GULOSO apresentou 18.121, superando os demais algoritmos nesse instante. O PSO atingiu 17.236, enquanto o GWO e o WOA registraram 17.199 e 17.170, respectivamente. O ALEATÓRIO apresentou 17.155 recursos, seguido pelo AHP, com 16.911.

Aos 1.000 segundos, o PRIOBILL alcançou 20.117 recursos utilizados. O GULOSO registrou 20.867, mantendo o maior valor entre os algoritmos. O PSO apresentou 19.410, seguido pelo GWO, com 19.384. O WOA e o ALEATÓRIO registraram 19.344 e 19.339, respectivamente. O AHP apresentou 19.148 recursos.

Aos 1.100 segundos, o PRIOBILL atingiu 23.085 recursos utilizados. O GULOSO apresentou 23.264, enquanto o AHP registrou 22.022. O GWO alcançou 21.749, seguido pelo PSO, com 21.717. O WOA e o ALEATÓRIO apresentaram 21.699 e 21.696, respectivamente.

Ao final de 1.200 segundos, o PRIOBILL registrou 25.416 recursos utilizados. O GULOSO alcançou 25.615, apresentando o maior valor nesse instante. O AHP atingiu 24.186, enquanto o GWO e o PSO registraram 23.838 e 23.796, respectivamente. O ALEATÓRIO apresentou 23.782, seguido pelo WOA, com 23.758.

Figura 54 – Quantidade de recursos utilizados com tarefas menores no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 55 apresenta-se a quantidade de recursos utilizados com tarefas menores no horário de pico. Aos 1.300 segundos de simulação, o GULOSO apresentou 28.186 recursos utilizados, seguido pelo PRIOBILL, com 27.894. O AHP registrou 26.911, enquanto o ALEATÓRIO atingiu 26.590. O GWO e o PSO apresentaram 26.620 e 26.606, respectivamente. O WOA registrou 26.518 recursos.

Aos 1.400 segundos, o GULOSO alcançou 30.385 recursos utilizados. O PRIOBILL apresentou 29.788, enquanto o AHP registrou 29.196. O GWO e o ALEATÓRIO atingiram 28.968 e 28.960, respectivamente. O PSO apresentou 28.925, seguido pelo WOA, com 28.836.

Aos 1.500 segundos, o GULOSO registrou 32.948 recursos utilizados. O PRI-

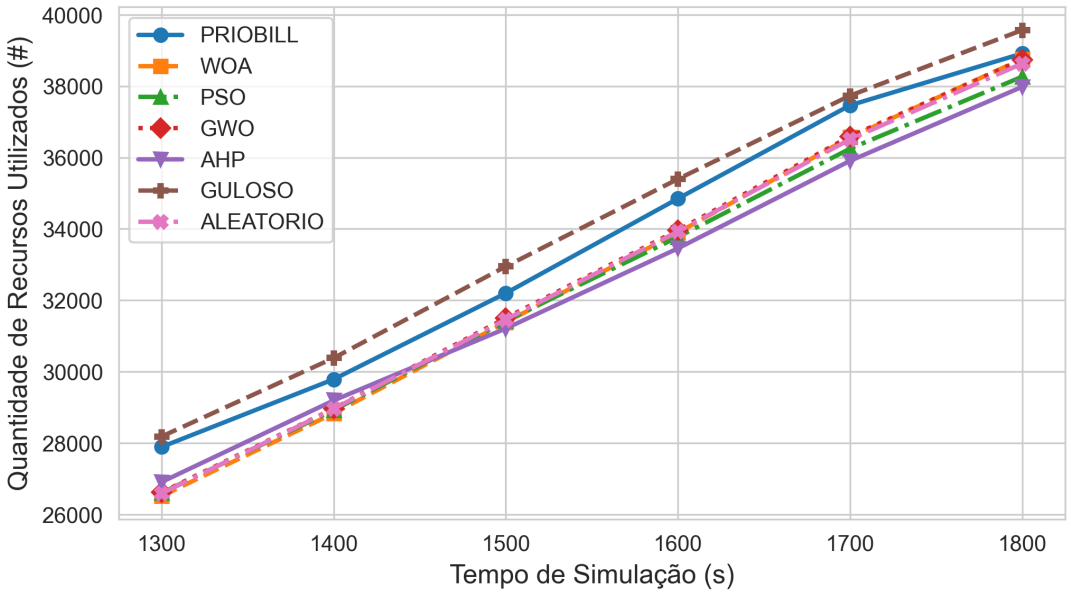
OBILL alcançou 32.200, seguido pelo PSO, com 31.420. O WOA apresentou 31.386, enquanto o GWO registrou 31.503. O ALEATÓRIO atingiu 31.463, e o AHP apresentou 31.211 recursos.

Aos 1.600 segundos, o GULOSO alcançou 35.396 recursos utilizados, seguido pelo PRIOBILL, com 34.848. O GWO apresentou 33.963, enquanto o ALEATÓRIO e o WOA registraram 33.926 e 33.896, respectivamente. O PSO atingiu 33.774, e o AHP apresentou 33.454 recursos.

Aos 1.700 segundos, o GULOSO registrou 37.732 recursos utilizados. O PRIOBILL alcançou 37.461, enquanto o GWO apresentou 36.590. O WOA e o ALEATÓRIO registraram 36.565 e 36.508, respectivamente. O PSO atingiu 36.246, enquanto o AHP apresentou 35.905 recursos.

Ao final de 1.800 segundos, o GULOSO alcançou 39.567 recursos utilizados, seguido pelo PRIOBILL, com 38.912. O WOA registrou 38.757, enquanto o GWO apresentou 38.739. O ALEATÓRIO atingiu 38.630, seguido pelo PSO, com 38.262. O AHP apresentou 37.977 recursos utilizados.

Figura 55 – Quantidade de recursos utilizados com tarefas menores no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 56 é apresentada a quantidade de recursos utilizados para tarefas intermediárias no horário da madrugada. Nos primeiros 200 segundos de simulação, nenhum dos algoritmos apresentou recursos utilizados. Aos 300 segundos, o PRIOBILL registrou 1.999 recursos utilizados, seguido pelo AHP, com 1.902. O ALEATÓRIO apresentou 1.782, enquanto o WOA e o GWO registraram 1.780 e 1.770, respectivamente. O PSO alcançou 1.711 recursos, e o GULOSO apresentou 1.298.

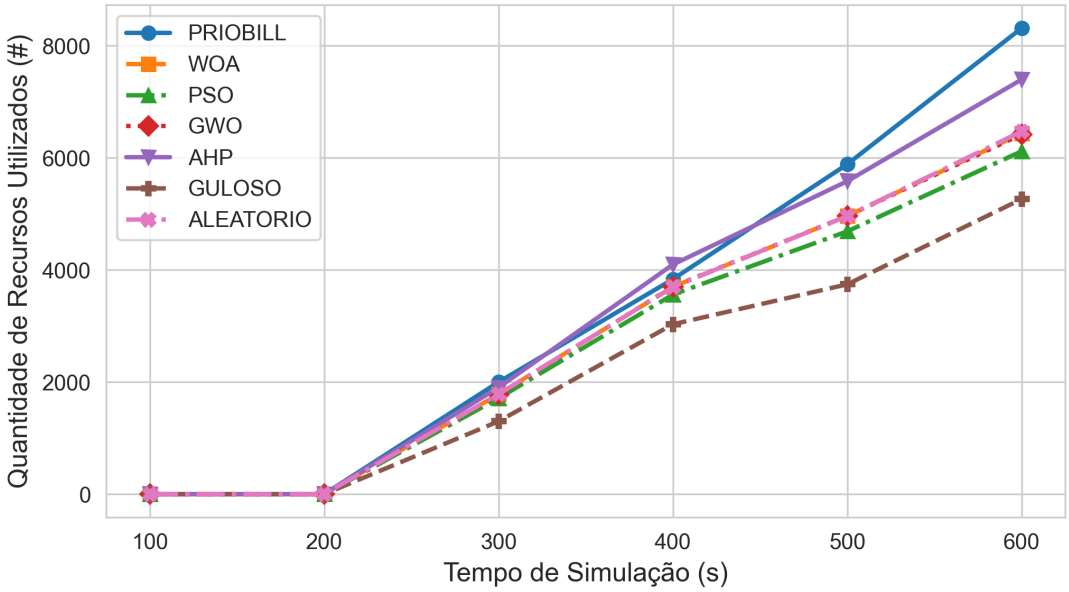
Aos 400 segundos, o PRIOBILL atingiu 3.834 recursos utilizados. O AHP apre-

sentou 4.095, superando os demais algoritmos. O WOA e o ALEATÓRIO registraram 3.697 e 3.696, respectivamente. O GWO alcançou 3.702, enquanto o PSO apresentou 3.560. O GULOSO registrou 3.030 recursos.

Aos 500 segundos, o PRIOBILL alcançou 5.884 recursos utilizados. O AHP apresentou 5.582, seguido pelo GWO, com 4.961. O ALEATÓRIO e o WOA registraram 4.960 e 4.959, respectivamente. O PSO atingiu 4.687, enquanto o GULOSO apresentou 3.742 recursos.

Ao final de 600 segundos, o PRIOBILL registrou 8.312 recursos utilizados. O AHP apresentou 7.395, enquanto o WOA e o ALEATÓRIO alcançaram 6.437 e 6.469, respectivamente. O GWO registrou 6.416, seguido pelo PSO, com 6.115. O GULOSO apresentou 5.263 recursos.

Figura 56 – Quantidade de recursos utilizados com tarefas intermediárias no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 57 é apresentada a quantidade de recursos utilizados para tarefas intermediárias no horário da manhã. Aos 700 segundos de simulação, o PRIOBILL alcançou 10.378 recursos utilizados. O AHP apresentou 8.551, seguido pelo ALEATÓRIO, com 7.879. O GWO registrou 7.956, enquanto o WOA e o PSO apresentaram 7.858 e 7.579, respectivamente. O GULOSO alcançou 6.746 recursos.

Aos 800 segundos, o PRIOBILL atingiu 13.707 recursos utilizados. O GULOSO apresentou 9.370, enquanto o AHP registrou 10.402. O GWO alcançou 10.285, seguido pelo WOA, com 10.214. O ALEATÓRIO apresentou 10.202, enquanto o PSO registrou 9.836 recursos.

Aos 900 segundos, o PRIOBILL alcançou 14.902 recursos utilizados. O GULOSO

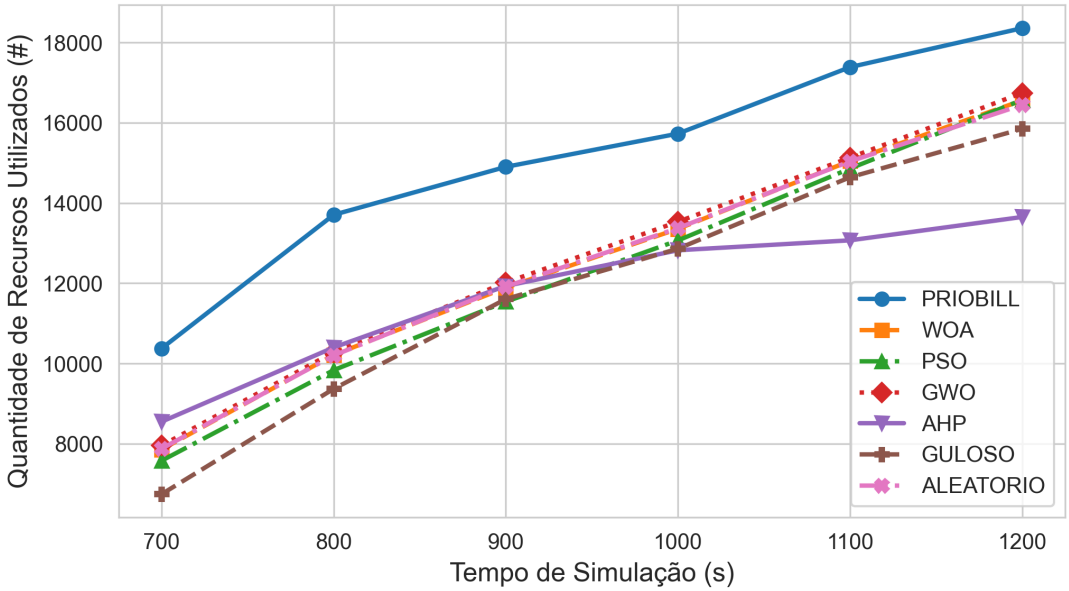
apresentou 11.607, seguido pelo PSO, com 11.541. O GWO registrou 12.019, enquanto o AHP e o WOA atingiram 11.933 e 11.891, respectivamente. O ALEATÓRIO apresentou 11.918 recursos.

Aos 1.000 segundos, o PRIOBILL registrou 15.730 recursos utilizados. O GULOSO apresentou 12.862, enquanto o AHP alcançou 12.823. O GWO atingiu 13.534, seguido pelo ALEATÓRIO, com 13.375. O WOA e o PSO registraram 13.357 e 13.067, respectivamente.

Aos 1.100 segundos, o PRIOBILL alcançou 17.387 recursos utilizados. O GULOSO apresentou 14.643, enquanto o GWO registrou 15.133. O WOA e o ALEATÓRIO atingiram 15.048 e 15.038, respectivamente. O PSO apresentou 14.860, enquanto o AHP registrou 13.071 recursos.

Ao final de 1.200 segundos, o PRIOBILL atingiu 18.358 recursos utilizados. O GWO apresentou 16.744, seguido pelo PSO, com 16.569. O WOA e o ALEATÓRIO registraram 16.557 e 16.444, respectivamente. O GULOSO alcançou 15.849, enquanto o AHP apresentou 13.653 recursos.

Figura 57 – Quantidade de recursos utilizados com tarefas intermediárias no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 58 apresenta-se a quantidade de recursos utilizados com tarefas intermediárias no horário de pico. Aos 1.300 segundos de simulação, o PRIOBILL alcançou 19.299 recursos utilizados. O WOA registrou 18.704, enquanto o GWO apresentou 18.894. O PSO e o ALEATÓRIO atingiram 18.672 e 18.608, respectivamente. O GULOSO registrou 17.771, enquanto o AHP apresentou 14.550 recursos.

Aos 1.400 segundos, o PRIOBILL atingiu 20.680 recursos utilizados. O WOA

apresentou 20.502, seguido pelo GWO, com 20.669. O ALEATÓRIO registrou 20.470, enquanto o PSO alcançou 20.426. O GULOSO apresentou 19.550, e o AHP registrou 15.733 recursos.

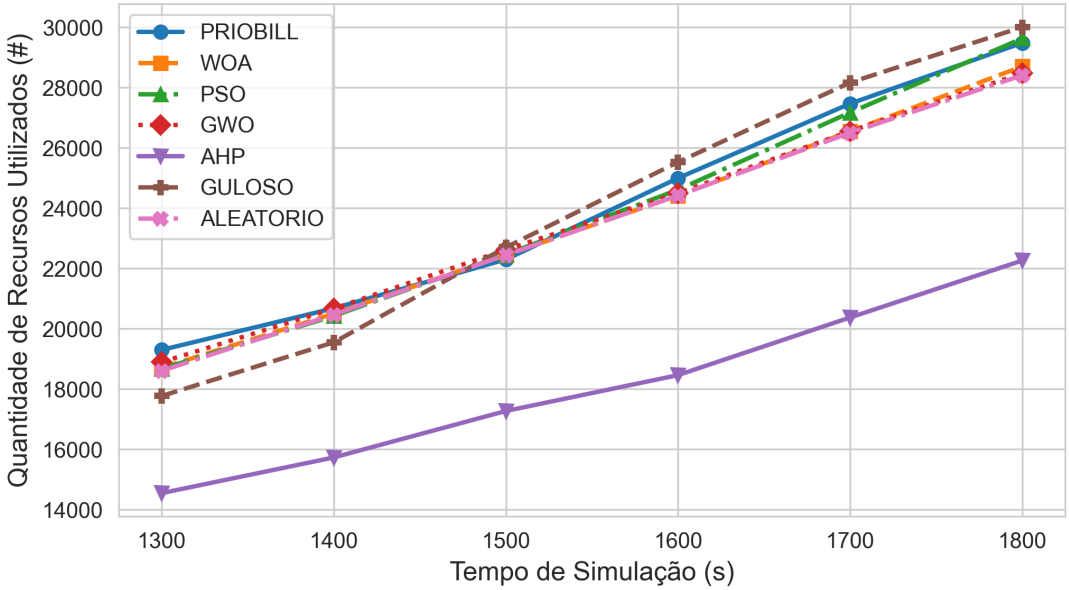
Aos 1.500 segundos, o GULOSO alcançou 22.700 recursos utilizados, enquanto o PRIOBILL apresentou 22.299. O GWO registrou 22.619, seguido pelo PSO, com 22.465. O ALEATÓRIO e o WOA atingiram 22.444 e 22.442, respectivamente. O AHP apresentou 17.273 recursos.

Aos 1.600 segundos, o GULOSO registrou 25.526 recursos utilizados. O PRIOBILL alcançou 24.994, enquanto o PSO apresentou 24.609. O ALEATÓRIO registrou 24.418, seguido pelo WOA, com 24.416. O GWO atingiu 24.498, enquanto o AHP apresentou 18.462 recursos.

Aos 1.700 segundos, o GULOSO alcançou 28.165 recursos utilizados. O PRIOBILL apresentou 27.465, seguido pelo PSO, com 27.170. O ALEATÓRIO registrou 26.500, enquanto o GWO e o WOA apresentaram 26.552 e 26.546, respectivamente. O AHP atingiu 20.373 recursos.

Ao final de 1.800 segundos, o GULOSO registrou 30.002 recursos utilizados, seguido pelo PSO, com 29.627. O PRIOBILL alcançou 29.479, enquanto o WOA apresentou 28.698. O GWO e o ALEATÓRIO registraram 28.471 e 28.405, respectivamente. O AHP apresentou 22.261 recursos.

Figura 58 – Quantidade de recursos utilizados com tarefas intermediárias no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 59 é apresentada a quantidade de recursos utilizados para tarefas maiores no horário da madrugada. Nos primeiros 200 segundos de simulação, nenhum

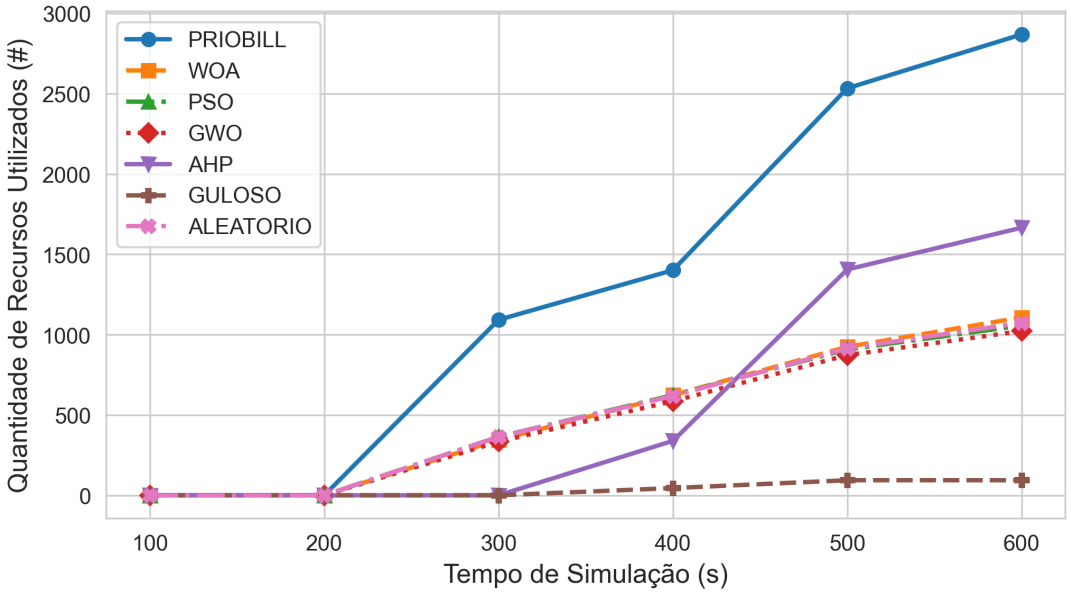
dos algoritmos apresentou recursos utilizados. Aos 300 segundos, o PRIOBILL registrou 1.092 recursos utilizados. O PSO apresentou 364, enquanto o ALEATÓRIO atingiu 363. O WOA e o GWO registraram 345 e 334, respectivamente. O AHP e o GULOSO não apresentaram recursos utilizados nesse instante.

Aos 400 segundos, o PRIOBILL alcançou 1.401 recursos utilizados. O AHP apresentou 338, enquanto o WOA e o PSO registraram 623 e 622, respectivamente. O ALEATÓRIO atingiu 618, seguido pelo GWO, com 585. O GULOSO apresentou 45 recursos.

Aos 500 segundos, o PRIOBILL atingiu 2.533 recursos utilizados. O AHP apresentou 1.405, enquanto o WOA e o ALEATÓRIO registraram 924 e 914, respectivamente. O PSO alcançou 908, seguido pelo GWO, com 872. O GULOSO apresentou 93 recursos.

Ao final de 600 segundos, o PRIOBILL registrou 2.868 recursos utilizados. O AHP alcançou 1.665, enquanto o WOA apresentou 1.107. O ALEATÓRIO e o PSO registraram 1.073 e 1.056, respectivamente. O GWO atingiu 1.022, enquanto o GULOSO permaneceu com 93 recursos utilizados.

Figura 59 – Quantidade de recursos utilizados com tarefas maiores no horário da madrugada.



Fonte: Gerado pelo autor.

A quantidade de recursos utilizados de tarefas maiores no cenário da manhã é apresentada na Figura 60. Neste cenário, pode-se observar uma dificuldade da simulação realista com tarefas pesadas, com poucas ou até nenhuma alocação realizada neste intervalo, conforme apresentado na Figura 36. Dessa forma, também são observadas poucas modificações na quantidade de recursos utilizados.

Aos 700 segundos de simulação, o PRIOBILL alcançou 2.925 recursos utilizados.

O AHP apresentou 1.665, seguido pelo WOA, com 1.148. O ALEATÓRIO registrou 1.119, enquanto o PSO e o GWO atingiram 1.101 e 1.069, respectivamente. O GULOSO apresentou 246 recursos.

Aos 800 segundos, o PRIOBILL manteve 2.925 recursos utilizados. O AHP também permaneceu com 1.665. O WOA alcançou 1.207, enquanto o ALEATÓRIO registrou 1.179. O PSO apresentou 1.163, seguido pelo GWO, com 1.126. O GULOSO permaneceu com 246 recursos.

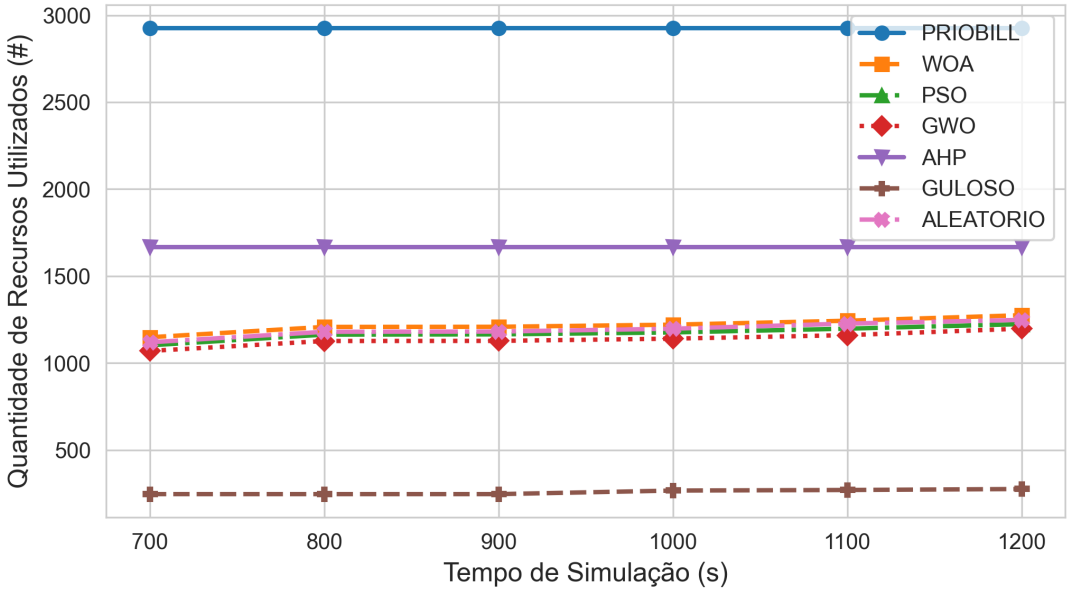
Aos 900 segundos, o PRIOBILL continuou com 2.925 recursos utilizados, assim como o AHP, que permaneceu com 1.665. O WOA apresentou 1.208, enquanto o ALEATÓRIO registrou 1.180. O PSO atingiu 1.164, seguido pelo GWO, com 1.127. O GULOSO manteve 246 recursos.

Aos 1.000 segundos, o PRIOBILL permaneceu com 2.925 recursos utilizados. O AHP registrou 1.665, enquanto o WOA alcançou 1.220. O ALEATÓRIO apresentou 1.197, seguido pelo PSO, com 1.175. O GWO registrou 1.140, enquanto o GULOSO atingiu 266 recursos.

Aos 1.100 segundos, o PRIOBILL manteve 2.925 recursos utilizados, seguido pelo AHP, com 1.665. O WOA alcançou 1.243, enquanto o ALEATÓRIO registrou 1.226. O PSO apresentou 1.197, seguido pelo GWO, com 1.160. O GULOSO atingiu 269 recursos.

Ao final de 1.200 segundos, o PRIOBILL permaneceu com 2.925 recursos utilizados. O AHP registrou 1.665, enquanto o WOA alcançou 1.274. O ALEATÓRIO apresentou 1.247, seguido pelo PSO, com 1.223. O GWO atingiu 1.197, enquanto o GULOSO apresentou 275 recursos.

Figura 60 – Quantidade de recursos utilizados com tarefas maiores no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 61 apresenta-se a quantidade de recursos utilizados com tarefas maiores no horário de pico. Aos 1.300 segundos de simulação, o PRIOBILL manteve 2.925 recursos utilizados. O WOA registrou 1.371, enquanto o AHP permaneceu com 1.665. O ALEATÓRIO alcançou 1.352, seguido pelo PSO, com 1.293. O GWO apresentou 1.289 recursos, enquanto o GULOSO registrou 404.

Aos 1.400 segundos, o PRIOBILL atingiu 3.022 recursos utilizados. O AHP apresentou 1.690, enquanto o WOA registrou 1.481. O ALEATÓRIO alcançou 1.492, seguido pelo GWO, com 1.509. O PSO apresentou 1.412, enquanto o GULOSO registrou 606 recursos.

Aos 1.500 segundos, o PRIOBILL alcançou 3.045 recursos utilizados. O AHP apresentou 1.771, enquanto o GWO registrou 1.737. O WOA e o ALEATÓRIO atingiram 1.591 e 1.618, respectivamente. O PSO apresentou 1.533, enquanto o GULOSO registrou 752 recursos.

Aos 1.600 segundos, o PRIOBILL registrou 3.048 recursos utilizados. O AHP alcançou 1.948, enquanto o GWO apresentou 1.942. O ALEATÓRIO registrou 1.746, seguido pelo WOA, com 1.693. O PSO apresentou 1.645, enquanto o GULOSO atingiu 1.020 recursos.

Aos 1.700 segundos, o PRIOBILL alcançou 3.166 recursos utilizados. O GWO apresentou 2.140, seguido pelo AHP, com 2.237. O ALEATÓRIO registrou 1.967, enquanto o WOA atingiu 1.854. O PSO apresentou 1.817, enquanto o GULOSO alcançou 1.179 recursos.

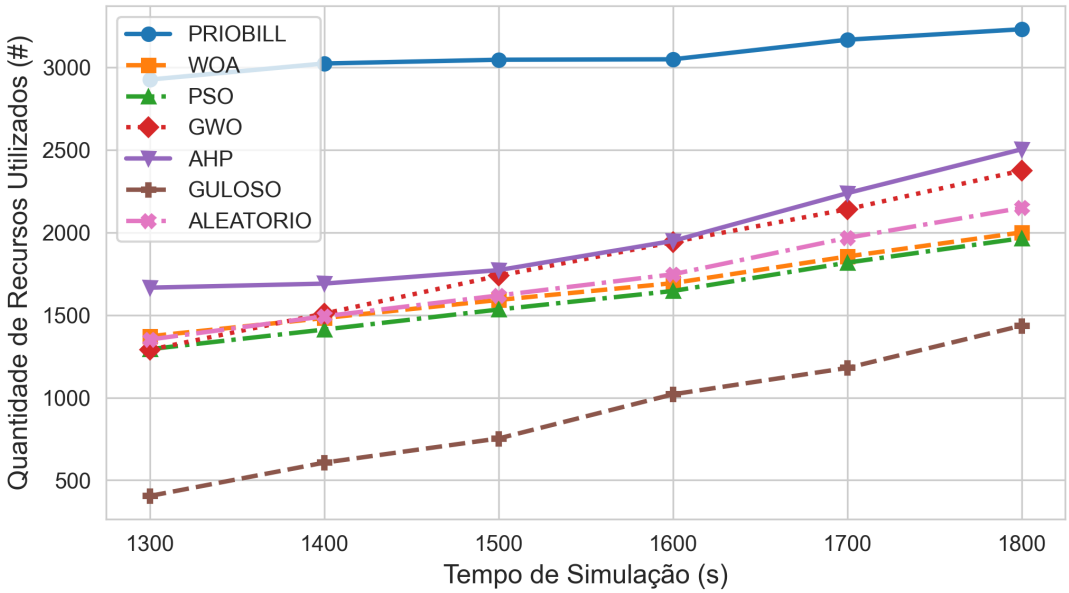
Ao final de 1.800 segundos, o PRIOBILL atingiu 3.230 recursos utilizados. O AHP apresentou 2.503, seguido pelo GWO, com 2.374. O ALEATÓRIO registrou 2.148, enquanto o WOA alcançou 2.000. O PSO apresentou 1.965, enquanto o GULOSO registrou 1.435 recursos.

Na Figura 62 é apresentada a quantidade de recursos utilizados de tarefas com pesos aleatórios no horário da madrugada. Nos primeiros 200 segundos de simulação, nenhum dos algoritmos apresentou recursos utilizados. Aos 300 segundos, o PRIOBILL alcançou 1.623 recursos utilizados. O PSO apresentou 967, enquanto o ALEATÓRIO registrou 935. O WOA e o GWO atingiram 923 e 909, respectivamente. O GULOSO apresentou 742, enquanto o AHP registrou 483 recursos.

Aos 400 segundos, o PRIOBILL atingiu 2.623 recursos utilizados. O WOA e o ALEATÓRIO registraram 1.881 e 1.884, respectivamente. O PSO apresentou 1.902, enquanto o GWO alcançou 1.853. O GULOSO registrou 1.732, seguido pelo AHP, com 1.472 recursos.

Aos 500 segundos, o PRIOBILL alcançou 4.147 recursos utilizados. O PSO apresentou 2.692, seguido pelo ALEATÓRIO, com 2.669. O GWO registrou 2.654, enquanto

Figura 61 – Quantidade de recursos utilizados com tarefas maiores no horário de pico.

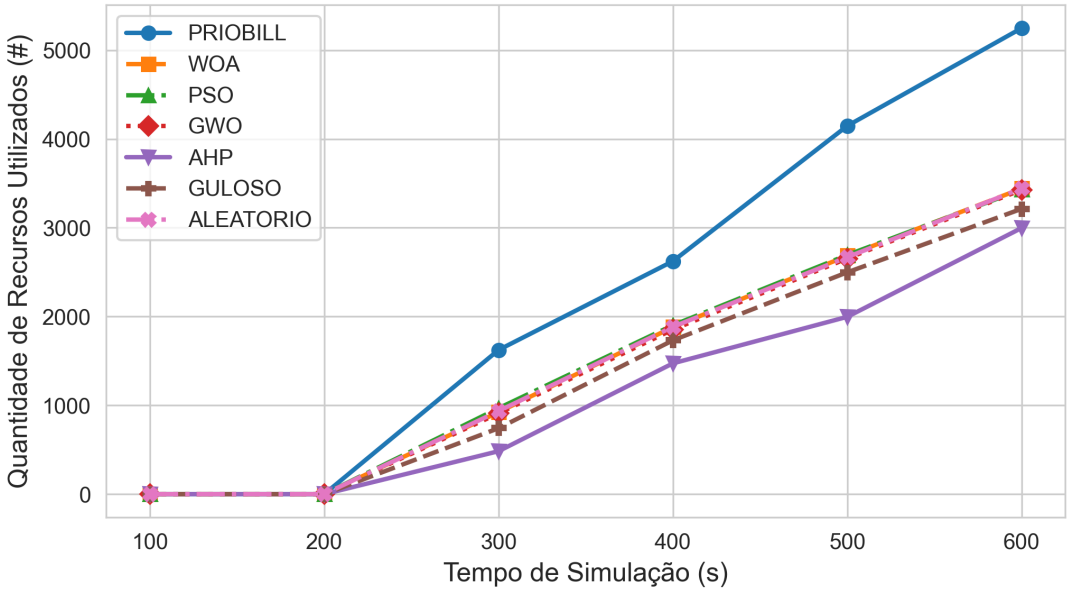


Fonte: Gerado pelo autor.

o WOA atingiu 2.684. O GULOSO apresentou 2.498, enquanto o AHP registrou 1.996 recursos.

Ao final de 600 segundos, o PRIOBILL atingiu 5.248 recursos utilizados. O WOA apresentou 3.441, enquanto o ALEATÓRIO registrou 3.442. O PSO alcançou 3.430, seguido pelo GWO, com 3.427. O GULOSO apresentou 3.213, enquanto o AHP registrou 2.997 recursos.

Figura 62 – Quantidade de recursos utilizados com tarefas com pesos aleatórios no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 63 é apresentada a quantidade de recursos utilizados de tarefas com pesos aleatórios no horário da manhã. Aos 700 segundos de simulação, o PRIOBILL alcançou 6.076 recursos utilizados. O GWO apresentou 4.163, seguido pelo WOA, com 4.157. O AHP registrou 4.155, enquanto o PSO e o ALEATÓRIO atingiram 4.144 e 4.140, respectivamente. O GULOSO apresentou 3.980 recursos.

Aos 800 segundos, o PRIOBILL atingiu 7.043 recursos utilizados. O AHP apresentou 5.964, enquanto o GWO registrou 5.150. O WOA alcançou 5.137, seguido pelo PSO, com 5.137. O ALEATÓRIO apresentou 5.126, enquanto o GULOSO registrou 4.558 recursos.

Aos 900 segundos, o PRIOBILL alcançou 7.446 recursos utilizados. O AHP apresentou 7.248, enquanto o GWO registrou 5.942. O ALEATÓRIO atingiu 5.888, seguido pelo WOA, com 5.874. O PSO apresentou 5.850, enquanto o GULOSO registrou 5.201 recursos.

Aos 1.000 segundos, o PRIOBILL registrou 7.849 recursos utilizados. O AHP alcançou 9.033, superando os demais algoritmos. O GWO apresentou 6.764, enquanto o ALEATÓRIO registrou 6.708. O WOA e o PSO atingiram 6.634 e 6.583, respectivamente. O GULOSO apresentou 5.940 recursos.

Aos 1.100 segundos, o PRIOBILL alcançou 8.551 recursos utilizados. O AHP apresentou 10.283, enquanto o GWO registrou 7.656. O ALEATÓRIO atingiu 7.598, seguido pelo WOA, com 7.526. O PSO apresentou 7.483, enquanto o GULOSO registrou 6.773 recursos.

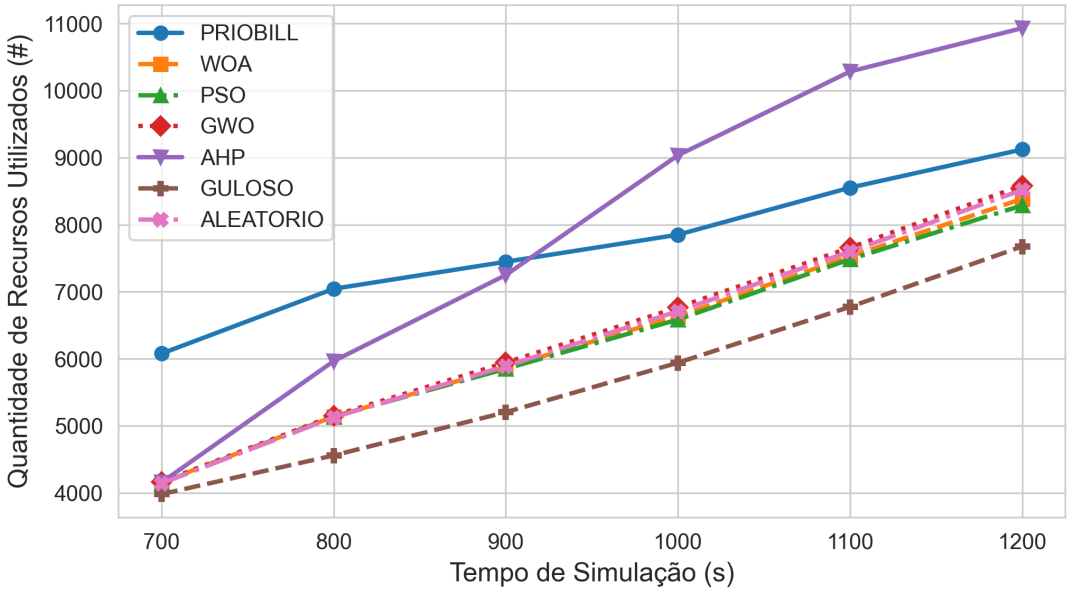
Ao final de 1.200 segundos, o PRIOBILL atingiu 9.121 recursos utilizados. O AHP apresentou 10.930, enquanto o GWO registrou 8.584. O ALEATÓRIO alcançou 8.516, seguido pelo WOA, com 8.383. O PSO apresentou 8.285, enquanto o GULOSO registrou 7.674 recursos.

A quantidade de recursos utilizados no cenário de tarefas com pesos aleatórios no horário de pico é apresentada na Figura 64. Aos 1.300 segundos de simulação, o PRIOBILL alcançou 9.773 recursos utilizados. O AHP apresentou 11.697, enquanto o WOA registrou 9.513. O GWO e o ALEATÓRIO atingiram 9.684 e 9.692, respectivamente. O PSO apresentou 9.370, enquanto o GULOSO registrou 8.527 recursos.

Aos 1.400 segundos, o PRIOBILL atingiu 10.234 recursos utilizados. O AHP apresentou 12.143, enquanto o ALEATÓRIO registrou 10.768. O GWO alcançou 10.750, seguido pelo WOA, com 10.529. O PSO apresentou 10.401, enquanto o GULOSO registrou 9.280 recursos.

Aos 1.500 segundos, o PRIOBILL alcançou 10.967 recursos utilizados. O AHP apresentou 13.014, enquanto o ALEATÓRIO registrou 11.936. O GWO atingiu 11.862, seguido pelo WOA, com 11.615. O PSO apresentou 11.447, enquanto o GULOSO registrou

Figura 63 – Quantidade de recursos utilizados com tarefas com pesos aleatórios no horário da manhã.



Fonte: Gerado pelo autor.

10.048 recursos.

Aos 1.600 segundos, o PRIOBILL atingiu 12.146 recursos utilizados. O AHP apresentou 14.046, enquanto o ALEATÓRIO registrou 13.026. O GWO alcançou 12.892, seguido pelo WOA, com 12.649. O PSO apresentou 12.427, enquanto o GULOSO registrou 11.009 recursos.

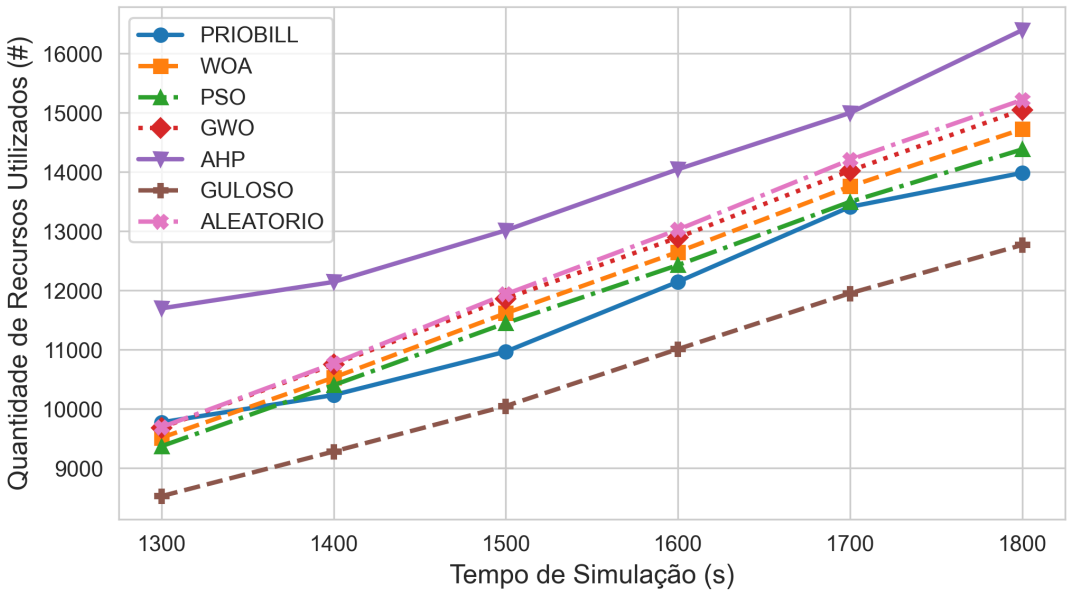
Aos 1.700 segundos, o PRIOBILL alcançou 13.410 recursos utilizados. O AHP apresentou 14.997, enquanto o ALEATÓRIO registrou 14.206. O GWO atingiu 14.021, seguido pelo WOA, com 13.757. O PSO apresentou 13.496, enquanto o GULOSO registrou 11.952 recursos.

Ao final de 1.800 segundos, o AHP alcançou 16.394 recursos utilizados. O ALEATÓRIO apresentou 15.217, enquanto o GWO registrou 15.047. O WOA atingiu 14.722, seguido pelo PSO, com 14.381. O PRIOBILL apresentou 13.985 recursos, enquanto o GULOSO registrou 12.767.

Em resumo, para a métrica de recursos utilizados, no cenário com tarefas menores, o PRIOBILL apresentou o melhor desempenho no primeiro horário de simulação, manteve um desempenho competitivo no segundo horário e, no terceiro, permaneceu próximo ao mecanismo com maior aproveitamento.

No cenário com tarefas intermediárias, os algoritmos iniciaram com desempenhos semelhantes em relação ao aproveitamento de recursos durante o horário da madrugada. Entretanto, ao longo da simulação, o PRIOBILL apresentou evolução e alcançou o melhor desempenho. No horário da manhã, o PRIOBILL obteve o maior aproveitamento em todos

Figura 64 – Quantidade de recursos utilizados com tarefas com pesos aleatórios no horário de pico.



Fonte: Gerado pelo autor.

os intervalos avaliados, enquanto no horário de pico manteve um desempenho consistente, permanecendo próximo aos demais algoritmos avaliados.

No cenário mais complexo, composto por tarefas maiores, o PRIOBILL apresentou desempenho superior no horário da madrugada. Nos demais períodos, assim como os outros mecanismos comparados, apresentou dificuldades de atendimento devido à maior demanda computacional das tarefas e à limitação dos recursos disponíveis nos veículos. Para tarefas com pesos aleatórios, o PRIOBILL apresentou o melhor desempenho no primeiro cenário de simulação e permaneceu competitivo até aproximadamente a metade do segundo cenário, porém apresentou redução de desempenho no cenário com maior quantidade de veículos.

Na avaliação da métrica de recursos utilizados, que busca maximizar o aproveitamento dos recursos disponibilizados pelas VECs, considerando a escassez desses recursos no ambiente veicular, o PRIOBILL demonstrou maior capacidade de utilização dos recursos disponíveis na maioria dos intervalos avaliados. A capacidade de adaptação do LOA, incorporada ao processo de seleção do nó de borda do PRIOBILL, contribuiu para o desempenho obtido mesmo diante das variações nas características das tarefas.

Além disso, o cálculo de *fitness* aplicado possibilitou a busca por um equilíbrio entre os requisitos da tarefa requisitante e a capacidade computacional do nó de borda selecionado. Dessa forma, evitou-se a alocação de tarefas com baixa demanda em veículos com grande disponibilidade de recursos, reduzindo a possibilidade de ociosidade dos recursos computacionais. Assim, o PRIOBILL conseguiu maximizar o aproveitamento dos

recursos disponibilizados pelas VECs, evitando desperdícios e contribuindo para uma utilização mais eficiente da infraestrutura veicular.

4.10.4 Latência média

Na Figura 65 apresenta-se os resultapresentam-seados de latência média no processamento de tarefas menores durante o horário da madrugada. Nos primeiros 200 segundos de simulação, não foram observadas tarefas processadas devido à baixa quantidade de veículos presentes no cenário, o que impossibilitou a formação adequada da infraestrutura de comunicação e processamento.

Aos 300 segundos de simulação, o GULOSO apresentou a menor latência média, com 0,088354 segundos. O GWO obteve latência de 0,088615 segundos, seguido pelo PSO com 0,088733 segundos. WOA, AHP, ALEATÓRIO e PRIOBILL apresentaram latências de 0,089524, 0,090791, 0,090888 e 0,092741 segundos, respectivamente.

Com 400 segundos de simulação, os valores de latência permaneceram semelhantes aos observados no intervalo anterior. O GULOSO manteve o menor tempo de processamento, com 0,088354 segundos, seguido pelo GWO, PSO, WOA, AHP, ALEATÓRIO e PRIOBILL, com latências de 0,088615, 0,088733, 0,089524, 0,090791, 0,090888 e 0,092741 segundos, respectivamente.

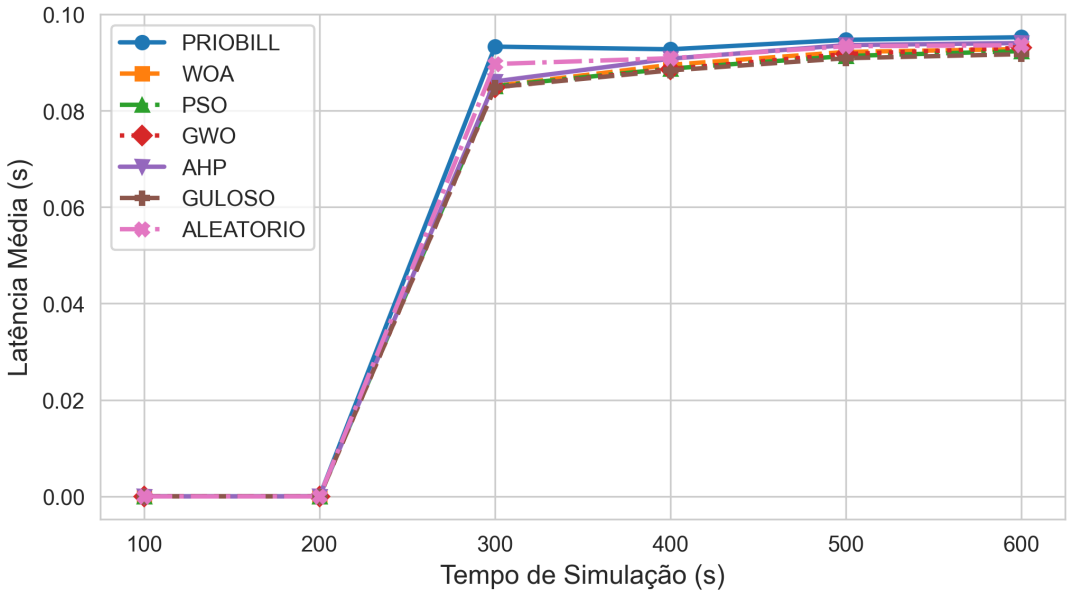
Aos 500 segundos de simulação, o GULOSO continuou apresentando a menor latência média, com 0,090887 segundos no processo de seleção do nó de borda. O PSO e o WOA apresentaram tempos de 0,091348 e 0,091765 segundos, respectivamente. Em seguida, o GWO, ALEATÓRIO, AHP e PRIOBILL obtiveram latências médias de 0,092193, 0,093367, 0,093608 e 0,094705 segundos, respectivamente.

Com 600 segundos de simulação, o GULOSO novamente apresentou o menor tempo de seleção do nó de borda, com latência média de 0,091733 segundos. O PSO, WOA, GWO, ALEATÓRIO, AHP e PRIOBILL apresentaram latências de 0,092358, 0,092786, 0,093037, 0,093591, 0,094042 e 0,095242 segundos, respectivamente.

Na Figura 66 apresenta-se a latência média obtida no processamento de tarefas menores durante o horário da manhã. Observa-se que, neste cenário, o PRIOBILL apresentou evolução no tempo de seleção do nó de borda em relação ao período anterior, mantendo valores competitivos de latência ao longo de todos os intervalos avaliados.

Com 700 segundos de simulação, o PRIOBILL alcançou uma latência média de 0,092368 segundos no processo de seleção. O PSO apresentou 0,093190 segundos, seguido pelo WOA com 0,093343 segundos. ALEATÓRIO e GWO obtiveram tempos de 0,093664 e 0,093896 segundos, respectivamente. O AHP apresentou 0,094276 segundos, enquanto o GULOSO obteve o maior tempo, com 0,096081 segundos.

Figura 65 – Latência média para processamento de tarefas menores no horário da madrugada.



Fonte: Gerado pelo autor.

Aos 800 segundos de simulação, o PRIOBILL manteve o melhor desempenho em relação à latência, com 0,092993 segundos para a seleção do nó de borda. O ALEATÓRIO apresentou 0,093732 segundos, seguido pelo WOA e PSO, com 0,093839 e 0,093934 segundos, respectivamente. O GWO e o AHP apresentaram latências de 0,094228 e 0,094282 segundos, enquanto o GULOSO permaneceu com o maior tempo, com 0,096081 segundos.

Com 900 segundos de simulação, o PRIOBILL continuou apresentando a menor latência média, com 0,093485 segundos. O ALEATÓRIO obteve 0,093881 segundos, seguido pelo AHP, WOA, PSO e GWO, com tempos de 0,094042, 0,094112, 0,094310 e 0,094442 segundos, respectivamente. O GULOSO apresentou 0,096611 segundos.

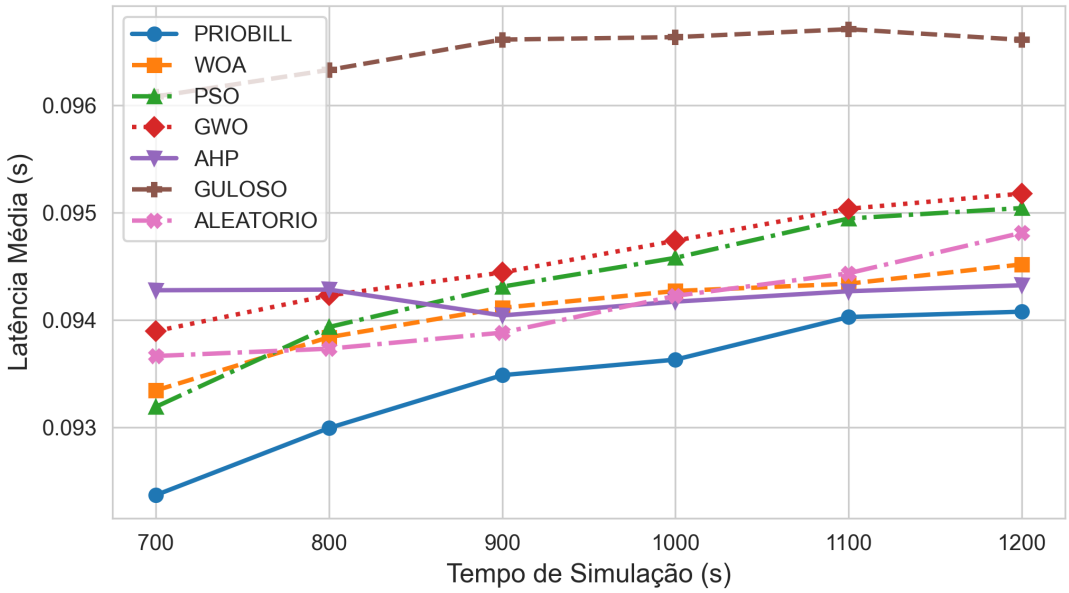
Aos 1.000 segundos de simulação, o PRIOBILL realizou a seleção do nó de borda em 0,093630 segundos, mantendo o melhor desempenho entre os mecanismos avaliados. O AHP apresentou o segundo menor tempo, com 0,094170 segundos, seguido pelo ALEATÓRIO, WOA, PSO, GWO e GULOSO, com latências médias de 0,094224, 0,094269, 0,094578, 0,094738 e 0,096634 segundos, respectivamente.

Com 1.100 segundos, o PRIOBILL manteve a menor latência média, com 0,094027 segundos. AHP, WOA, ALEATÓRIO, PSO, GWO e GULOSO apresentaram tempos de 0,094267, 0,094337, 0,094435, 0,094945, 0,095036 e 0,096708 segundos, respectivamente.

No último intervalo avaliado, aos 1.200 segundos de simulação, o PRIOBILL novamente apresentou o menor tempo de seleção, com 0,094077 segundos. O AHP obteve 0,094323 segundos, seguido pelo WOA e ALEATÓRIO com 0,094517 e 0,094812 segundos, respectivamente. PSO, GWO e GULOSO apresentaram latências médias de 0,095042,

0,095177 e 0,096609 segundos, respectivamente.

Figura 66 – Latência média para processamento de tarefas menores no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 67 apresenta-se a latência média obtida no processamento de tarefas menores durante o horário de pico. Neste cenário, observa-se que os mecanismos avaliados apresentaram valores próximos de latência, indicando que o aumento da densidade de veículos e da quantidade de requisições não provocou grande impacto no tempo de seleção do nó de borda.

Com 1.300 segundos de simulação, o WOA apresentou a menor latência média, com 0,094494 segundos. O AHP e o PRIOBILL apresentaram tempos de 0,094591 e 0,094795 segundos, respectivamente. O ALEATÓRIO obteve latência de 0,095015 segundos, seguido pelo PSO com 0,095248 segundos e pelo GWO com 0,095287 segundos. O GULOSO apresentou a maior latência, com 0,096663 segundos.

Aos 1.400 segundos de simulação, o WOA manteve o menor tempo de seleção, com 0,094605 segundos. O AHP e o PRIOBILL apresentaram latências médias de 0,094861 e 0,095121 segundos, respectivamente. ALEATÓRIO, PSO, GWO e GULOSO obtiveram tempos de 0,095122, 0,095393, 0,095457 e 0,096626 segundos, respectivamente.

Com 1.500 segundos de simulação, o WOA novamente apresentou o melhor desempenho em latência, com 0,094661 segundos. O AHP obteve 0,094993 segundos, seguido pelo PRIOBILL com 0,095230 segundos. O ALEATÓRIO apresentou 0,095280 segundos, enquanto PSO, GWO e GULOSO alcançaram latências de 0,095486, 0,095652 e 0,096548 segundos, respectivamente.

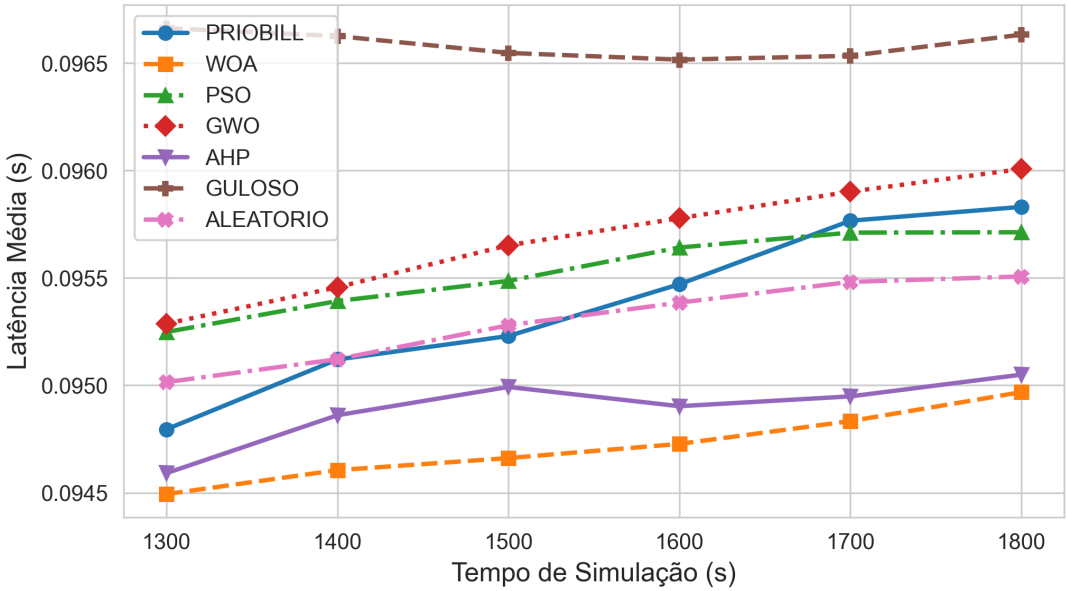
Aos 1.600 segundos, o WOA continuou apresentando a menor latência média, com 0,094727 segundos. O AHP apresentou 0,094993 segundos, seguido pelo ALEATÓRIO

com 0,095385 segundos. O PRIOBILL, PSO, GWO e GULOSO apresentaram tempos de 0,095471, 0,095642, 0,095778 e 0,096516 segundos, respectivamente.

Com 1.700 segundos de simulação, o WOA manteve o menor tempo de seleção, com 0,094834 segundos, seguido pelo AHP com 0,094949 segundos. O ALEATÓRIO apresentou 0,095482 segundos, enquanto o PSO obteve 0,095711 segundos. O PRIOBILL apresentou latência média de 0,095766 segundos, seguido pelo GWO com 0,095902 segundos. O GULOSO novamente apresentou o maior tempo, com 0,096534 segundos.

No último intervalo avaliado, aos 1.800 segundos de simulação, o WOA permaneceu com a menor latência média, com 0,094968 segundos. AHP, ALEATÓRIO, PSO, PRIOBILL, GWO e GULOSO apresentaram latências de 0,095050, 0,095507, 0,095713, 0,095831, 0,096007 e 0,096633 segundos, respectivamente.

Figura 67 – Latência média para processamento de tarefas menores no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 68 apresenta-se a latência média obtida no processamento de tarefas intermediárias durante o horário da madrugada. Nos primeiros intervalos de 100 e 200 segundos de simulação, não foram observados processamentos devido à baixa densidade de veículos presente no cenário, impossibilitando a formação adequada de clusters e, consequentemente, a execução das solicitações de tarefas.

Com 300 segundos de simulação, o GULOSO apresentou a menor latência média, com 0,085140 segundos. O GWO e o PRIOBILL apresentaram tempos de 0,086132 e 0,086371 segundos, respectivamente. WOA, AHP, PSO e ALEATÓRIO obtiveram latências de 0,086563, 0,086637, 0,086993 e 0,087050 segundos, respectivamente.

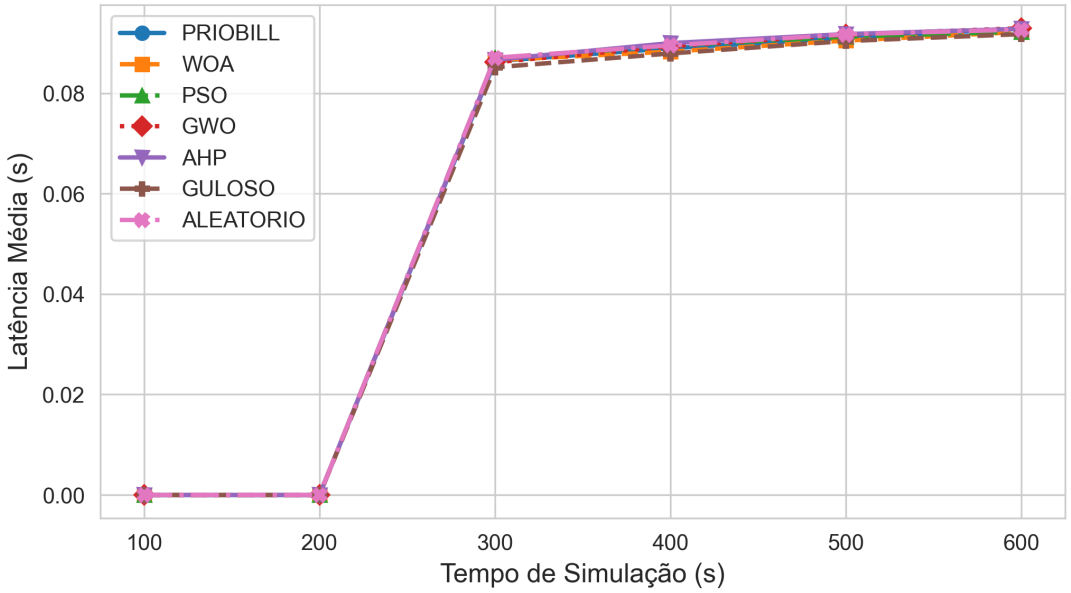
Aos 400 segundos de simulação, o GULOSO manteve o menor tempo de seleção, com latência média de 0,087894 segundos. O WOA apresentou 0,088261 segundos, seguido

pelo PRIOBILL com 0,088935 segundos. O GWO obteve 0,089345 segundos, enquanto PSO, ALEATÓRIO e AHP apresentaram tempos de 0,089487, 0,089504 e 0,089961 segundos, respectivamente.

Com 500 segundos de simulação, GULOSO e WOA apresentaram as menores latências, com valores de 0,090363 e 0,090454 segundos, respectivamente. O PRIOBILL obteve uma latência média de 0,091027 segundos. PSO e GWO apresentaram tempos de 0,091226 e 0,091629 segundos, enquanto ALEATÓRIO e AHP alcançaram 0,091709 e 0,091758 segundos, respectivamente.

Aos 600 segundos de simulação, o GULOSO novamente apresentou a menor latência média, com 0,091745 segundos. O WOA obteve 0,092176 segundos, seguido pelo PSO com 0,092220 segundos. ALEATÓRIO e AHP apresentaram latências de 0,092660 e 0,092800 segundos, respectivamente. O PRIOBILL obteve 0,092824 segundos, enquanto o GWO apresentou a maior latência entre os mecanismos avaliados, com 0,092921 segundos.

Figura 68 – Latência média para processamento de tarefas intermediárias no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 69 apresenta-se a latência média obtida no processamento de tarefas intermediárias durante o horário da manhã. Neste cenário, observa-se que os mecanismos apresentaram valores próximos de latência, com o GULOSO obtendo os menores tempos na maioria dos intervalos avaliados. O PRIOBILL apresentou comportamento competitivo, mantendo valores próximos aos demais mecanismos, apesar da maior complexidade do processo de seleção baseado no LOA.

Com 700 segundos de simulação, o GULOSO apresentou a menor latência média, com 0,092417 segundos. PSO e WOA obtiveram tempos de 0,092801 e 0,092869 segundos,

respectivamente. ALEATÓRIO, AHP, GWO e PRIOBILL apresentaram latências de 0,093033, 0,093057, 0,093443 e 0,093675 segundos, respectivamente.

Aos 800 segundos de simulação, o GULOSO novamente apresentou o menor tempo de processamento, com 0,092449 segundos. O PSO obteve latência de 0,093111 segundos, seguido pelo ALEATÓRIO com 0,093123 segundos. O AHP apresentou 0,093250 segundos, enquanto WOA, GWO e PRIOBILL tiveram latências de 0,093359, 0,093662 e 0,094183 segundos, respectivamente.

Com 900 segundos de simulação, o GULOSO manteve o melhor desempenho em latência, com 0,092604 segundos. PSO, ALEATÓRIO, AHP, WOA, GWO e PRIOBILL apresentaram tempos de 0,093285, 0,093423, 0,093586, 0,093818, 0,093927 e 0,094541 segundos, respectivamente.

Aos 1.000 segundos de simulação, o GULOSO apresentou latência média de 0,092904 segundos. O PSO obteve 0,093542 segundos, seguido pelo ALEATÓRIO com 0,093672 segundos e pelo AHP com 0,093766 segundos. O WOA apresentou latência de 0,094017 segundos, enquanto o GWO obteve 0,094135 segundos. O PRIOBILL apresentou tempo médio de 0,094703 segundos.

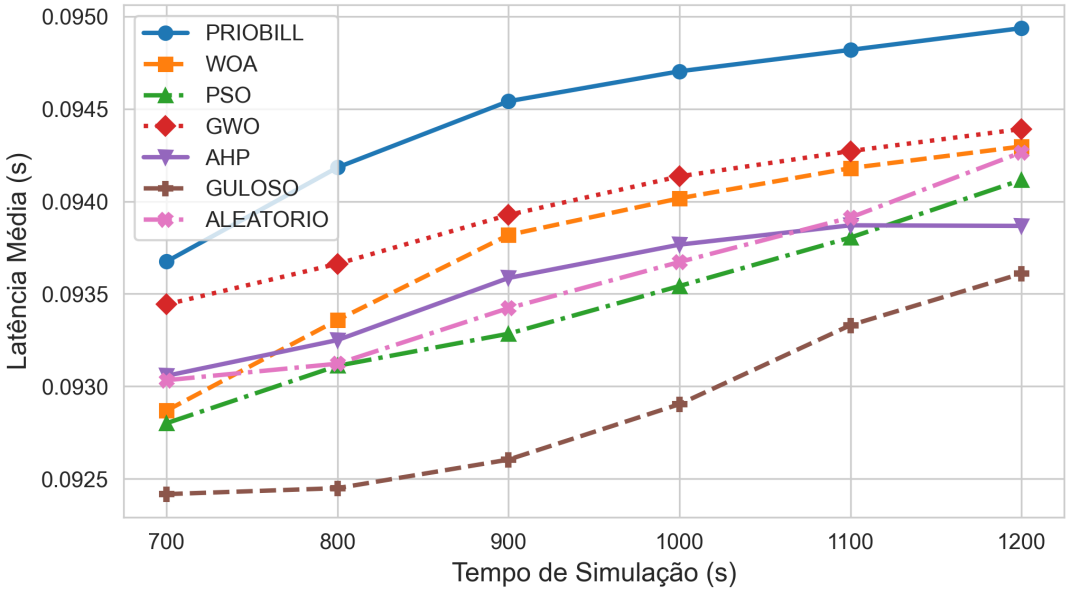
Com 1.100 segundos de simulação, o GULOSO continuou apresentando a menor latência, com 0,093331 segundos. PSO, AHP, ALEATÓRIO, WOA, GWO e PRIOBILL obtiveram valores de 0,093804, 0,093871, 0,093913, 0,094180, 0,094272 e 0,094819 segundos, respectivamente.

No último intervalo avaliado, aos 1.200 segundos de simulação, o GULOSO apresentou latência de 0,093609 segundos. O AHP obteve 0,093867 segundos, seguido pelo PSO com 0,094116 segundos. ALEATÓRIO, WOA e GWO apresentaram latências de 0,094265, 0,094297 e 0,094391 segundos, respectivamente. O PRIOBILL obteve 0,094936 segundos.

Na Figura 70 apresenta-se a latência média obtida no processamento de tarefas intermediárias durante o horário de pico. Neste cenário, o PRIOBILL apresentou o melhor desempenho na maioria dos intervalos avaliados, obtendo as menores latências entre os mecanismos comparados. Esse resultado evidencia que, mesmo com o aumento da densidade de veículos e da demanda por processamento, o processo de seleção baseado no LOA conseguiu manter baixa sobrecarga computacional durante a escolha do nó de borda.

Com 1.300 segundos de simulação, o PRIOBILL apresentou a menor latência média, com 0,093788 segundos. O AHP e o PSO obtiveram tempos de 0,094103 e 0,094385 segundos, respectivamente. O WOA apresentou latência de 0,094401 segundos, seguido pelo ALEATÓRIO com 0,094446 segundos, pelo GWO com 0,094615 segundos e pelo GULOSO com 0,095075 segundos.

Figura 69 – Latência média para processamento de tarefas intermediárias no horário da manhã.



Fonte: Gerado pelo autor.

Aos 1.400 segundos de simulação, o PRIOBILL novamente apresentou o menor tempo de seleção, com latência média de 0,093988 segundos. O WOA, PSO, ALEATÓRIO e GWO obtiveram valores de 0,094503, 0,094574, 0,094611 e 0,094728 segundos, respectivamente. O AHP apresentou latência de 0,095199 segundos, enquanto o GULOSO obteve o mesmo valor de 0,095199 segundos.

Com 1.500 segundos de simulação, o PRIOBILL manteve o melhor desempenho, com latência de 0,094119 segundos. O AHP apresentou 0,094408 segundos, seguido pelo WOA com 0,094632 segundos. O PSO obteve 0,094727 segundos, enquanto GWO, ALEATÓRIO e GULOSO apresentaram latências de 0,094765, 0,094771 e 0,095281 segundos, respectivamente.

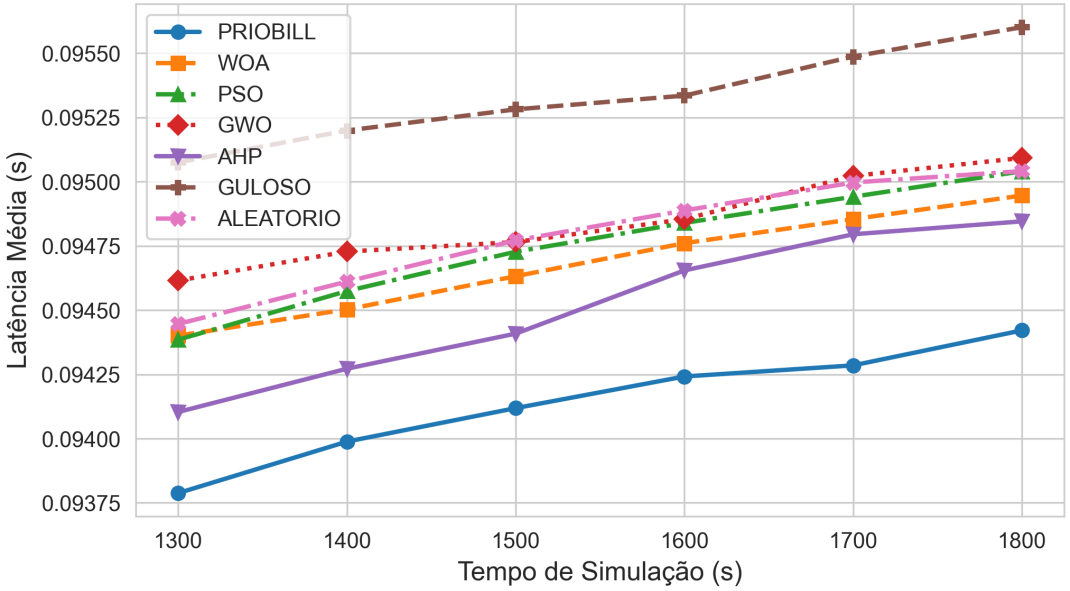
Aos 1.600 segundos de simulação, o PRIOBILL novamente apresentou a menor latência média, com 0,094242 segundos. O AHP obteve 0,094654 segundos, seguido por WOA, PSO, GWO, ALEATÓRIO e GULOSO, com valores de 0,094760, 0,094839, 0,094854, 0,094888 e 0,095335 segundos, respectivamente.

Com 1.700 segundos de simulação, o PRIOBILL manteve a menor latência, com 0,094285 segundos. O AHP apresentou 0,094795 segundos, seguido pelo WOA com 0,094854 segundos. O PSO obteve 0,094941 segundos, enquanto o ALEATÓRIO apresentou 0,094997 segundos. O GWO alcançou 0,095022 segundos e o GULOSO apresentou 0,095487 segundos de latência.

No último intervalo avaliado, aos 1.800 segundos de simulação, o PRIOBILL apresentou latência média de 0,094421 segundos. Na sequência, AHP, WOA, PSO,

ALEATÓRIO, GWO e GULOSO obtiveram valores de 0,094845, 0,094946, 0,095040, 0,095041, 0,095092 e 0,095601 segundos, respectivamente.

Figura 70 – Latência média para processamento de tarefas intermediárias no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 71 apresenta-se a latência média obtida no processamento de tarefas maiores durante o horário da madrugada. Nos primeiros 200 segundos de simulação, não foram observadas alocações de tarefas devido à baixa quantidade de veículos presentes no cenário, o que impossibilitou a formação adequada dos clusters e a execução das solicitações de processamento.

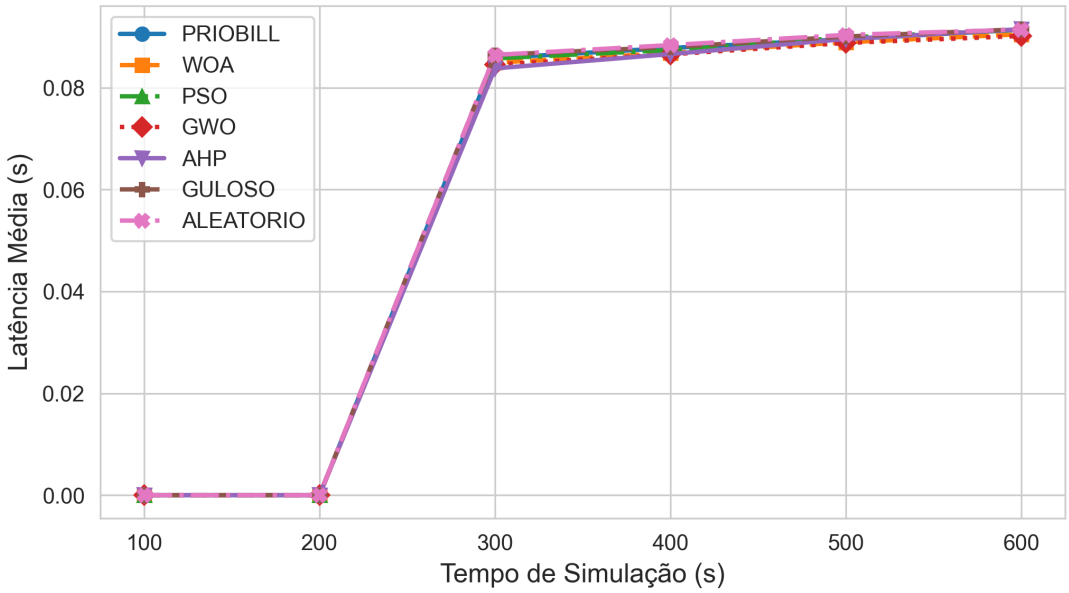
Com 300 segundos de simulação, o AHP apresentou a menor latência média, com 0,083796 segundos. O GWO e o WOA obtiveram tempos de 0,084542 e 0,084918 segundos, respectivamente. O PSO apresentou latência de 0,085762 segundos, seguido pelo PRIOBILL com 0,085836 segundos. ALEATÓRIO e GULOSO apresentaram os maiores tempos entre os mecanismos avaliados, com 0,086452 e 0,086464 segundos, respectivamente.

Aos 400 segundos de simulação, o GWO apresentou a menor latência média, com 0,086556 segundos. O AHP obteve 0,086619 segundos, seguido pelo WOA com 0,086842 segundos. O PSO apresentou latência de 0,087404 segundos, enquanto o PRIOBILL alcançou 0,087816 segundos. GULOSO e ALEATÓRIO apresentaram tempos de 0,087987 e 0,088386 segundos, respectivamente.

Com 600 segundos de simulação, o GWO manteve o menor tempo de processamento, com latência média de 0,090170 segundos. O WOA apresentou 0,090506 segundos, seguido pelo PRIOBILL com 0,091398 segundos. PSO, ALEATÓRIO, GULOSO e AHP obtiveram latências de 0,091406, 0,091428, 0,091498 e 0,091507 segundos, respectiva-

mente.

Figura 71 – Latência média para processamento de tarefas maiores no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 72 apresenta-se a latência média obtida no processamento de tarefas maiores durante o horário da manhã. Neste cenário, observa-se que o WOA apresentou os menores valores de latência na maioria dos intervalos avaliados, seguido pelo GWO. O PRIOBILL manteve comportamento competitivo, apresentando valores próximos aos demais mecanismos, mesmo considerando o processo adicional de otimização empregado na seleção do nó de borda.

Com 700 segundos de simulação, o WOA apresentou a menor latência média, com 0,091109 segundos. O GWO obteve 0,091164 segundos, seguido pelo ALEATÓRIO com 0,092074 segundos e pelo PRIOBILL com 0,092214 segundos. GULOSO, PSO e AHP apresentaram latências médias de 0,092262, 0,092268 e 0,092352 segundos, respectivamente.

Aos 800 segundos de simulação, o WOA manteve o melhor desempenho, com latência de 0,091617 segundos. O GWO, ALEATÓRIO, PSO, PRIOBILL, GULOSO e AHP apresentaram valores de 0,091964, 0,092074, 0,092664, 0,092659, 0,092700 e 0,092881 segundos, respectivamente.

Com 900 segundos de simulação, o WOA apresentou novamente a menor latência média, com 0,092101 segundos. O GWO obteve 0,092386 segundos, seguido pelo ALEATÓRIO com 0,092682 segundos. PSO, PRIOBILL, GULOSO e AHP apresentaram latências de 0,092878, 0,093103, 0,093216 e 0,093304 segundos, respectivamente.

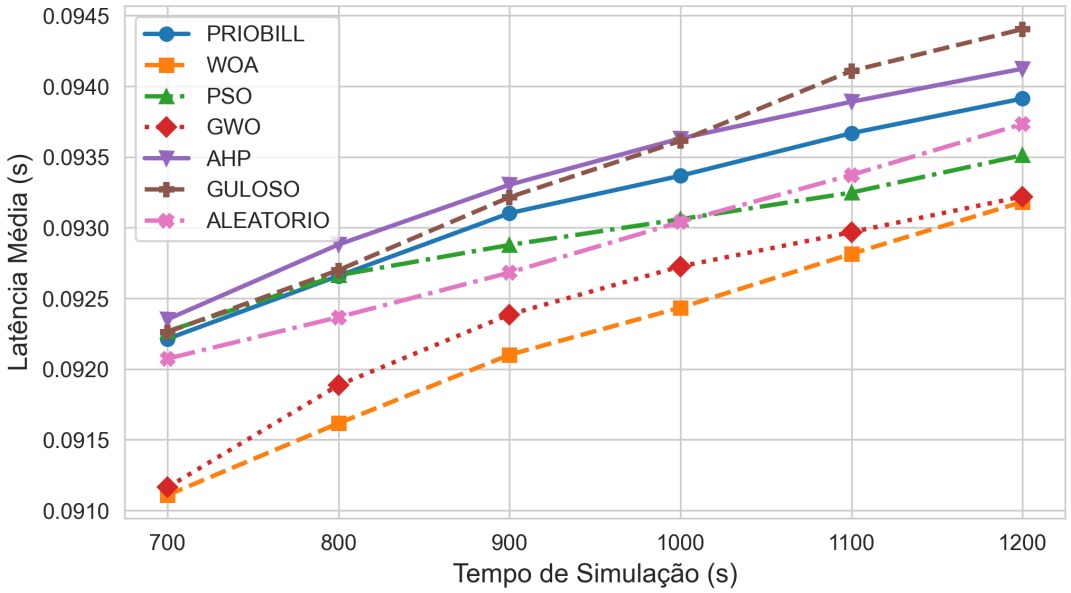
Aos 1.000 segundos de simulação, o WOA apresentou latência média de 0,092435

segundos. O GWO obteve 0,092726 segundos, enquanto ALEATÓRIO e PSO apresentaram valores de 0,093041 e 0,093060 segundos, respectivamente. O PRIOBILL alcançou 0,093367 segundos, seguido pelo GULOSO com 0,093613 segundos e pelo AHP com 0,093631 segundos.

Com 1.100 segundos de simulação, o WOA continuou apresentando a menor latência, com 0,092815 segundos. O GWO obteve 0,092968 segundos, seguido pelo PSO com 0,093248 segundos e pelo ALEATÓRIO com 0,093373 segundos. O PRIOBILL apresentou latência média de 0,093668 segundos, enquanto AHP e GULOSO obtiveram 0,093890 e 0,094108 segundos, respectivamente.

No último intervalo avaliado, aos 1.200 segundos de simulação, o WOA apresentou latência média de 0,093182 segundos. O GWO obteve 0,093220 segundos, seguido pelo PSO com 0,093512 segundos e pelo ALEATÓRIO com 0,093733 segundos. O PRIOBILL apresentou latência de 0,093913 segundos, enquanto AHP e GULOSO obtiveram 0,094124 e 0,094404 segundos, respectivamente.

Figura 72 – Latência média para processamento de tarefas maiores no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 73 apresenta-se a latência média obtida no processamento de tarefas maiores durante o horário de pico. Neste cenário, observa-se que o WOA apresentou o menor tempo de processamento na maioria dos intervalos avaliados, seguido pelo GWO e PSO. O PRIOBILL manteve um comportamento competitivo, apresentando latências próximas aos melhores resultados, mesmo considerando o processo de seleção baseado no LOA.

Com 1.300 segundos de simulação, o WOA apresentou a menor latência média, com 0,093379 segundos. O GWO obteve 0,093496 segundos, enquanto PSO e ALEATÓRIO

apresentaram valores de 0,093773 e 0,093973 segundos, respectivamente. O PRIOBILL alcançou latência de 0,094111 segundos, seguido pelo AHP com 0,094312 segundos e pelo GULOSO com 0,094655 segundos.

Aos 1.400 segundos de simulação, o WOA manteve o melhor desempenho, com latência média de 0,093633 segundos. O GWO apresentou 0,093661 segundos, seguido pelo PSO com 0,093972 segundos e pelo ALEATÓRIO com 0,094157 segundos. O PRIOBILL, AHP e GULOSO obtiveram latências de 0,094320, 0,094519 e 0,094918 segundos, respectivamente.

Com 1.500 segundos de simulação, o WOA apresentou latência de 0,093882 segundos, seguido de perto pelo GWO com 0,093893 segundos. O PSO alcançou 0,094126 segundos, enquanto o ALEATÓRIO apresentou 0,094328 segundos. O PRIOBILL, AHP e GULOSO obtiveram valores de 0,094476, 0,094696 e 0,095085 segundos, respectivamente.

Aos 1.600 segundos de simulação, o WOA continuou apresentando a menor latência, com 0,094012 segundos. O GWO obteve 0,094059 segundos, seguido pelo PSO com 0,094248 segundos e pelo ALEATÓRIO com 0,094487 segundos. O PRIOBILL apresentou latência média de 0,094566 segundos, enquanto AHP e GULOSO alcançaram 0,094766 e 0,095279 segundos, respectivamente.

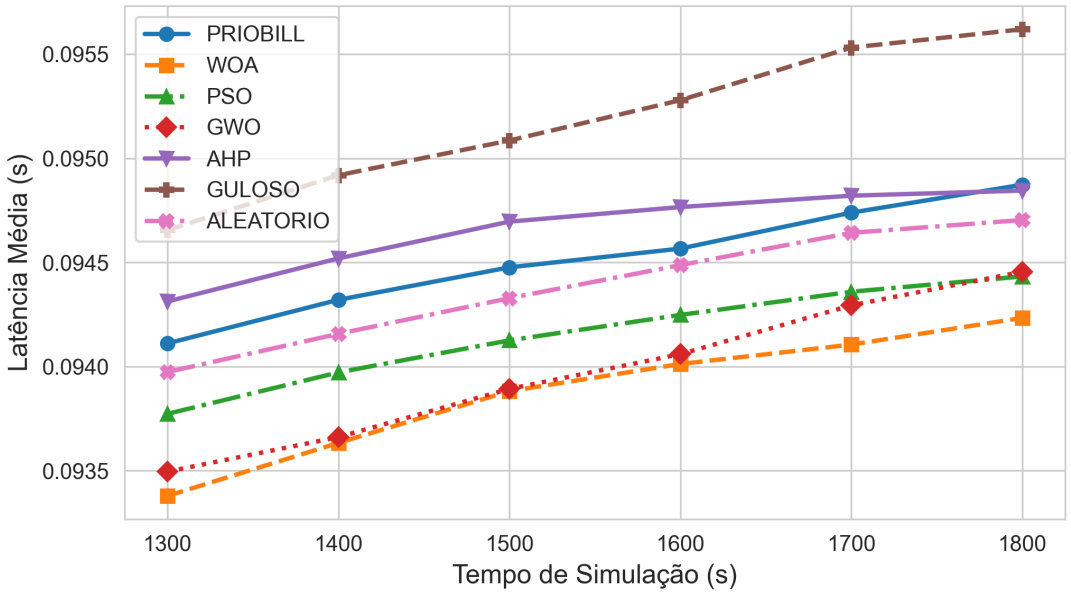
Com 1.700 segundos de simulação, o WOA apresentou latência média de 0,094105 segundos. O GWO obteve 0,094294 segundos, enquanto PSO e ALEATÓRIO apresentaram 0,094359 e 0,094642 segundos, respectivamente. O PRIOBILL alcançou 0,094739 segundos, seguido pelo AHP com 0,094820 segundos e pelo GULOSO com 0,095531 segundos.

No último intervalo avaliado, aos 1.800 segundos de simulação, o WOA manteve a menor latência, com 0,094234 segundos. O PSO e o GWO apresentaram valores de 0,094432 e 0,094455 segundos, respectivamente. O ALEATÓRIO obteve 0,094703 segundos, enquanto o PRIOBILL, AHP e GULOSO apresentaram latências de 0,094873, 0,094845 e 0,095619 segundos, respectivamente.

Na Figura 74 apresenta-se a latência média obtida no processamento de tarefas com pesos aleatórios durante o horário da madrugada. Nos primeiros 200 segundos de simulação, não ocorreram alocações devido à baixa densidade de veículos no cenário avaliado. A partir do momento em que houve maior disponibilidade de nós de borda, os mecanismos passaram a apresentar valores de latência próximos, indicando baixo impacto da variação dos pesos das tarefas no tempo de seleção.

Com 300 segundos de simulação, o GWO apresentou a menor latência média, com 0,084516 segundos. O WOA e o PRIOBILL obtiveram valores de 0,085740 e 0,086130 segundos, respectivamente. O PSO apresentou latência de 0,086577 segundos, seguido pelo ALEATÓRIO com 0,086665 segundos e pelo GULOSO com 0,087105 segundos. O

Figura 73 – Latência média para processamento de tarefas maiores no horário de pico.



Fonte: Gerado pelo autor.

AHP apresentou a maior latência no intervalo, com 0,087831 segundos.

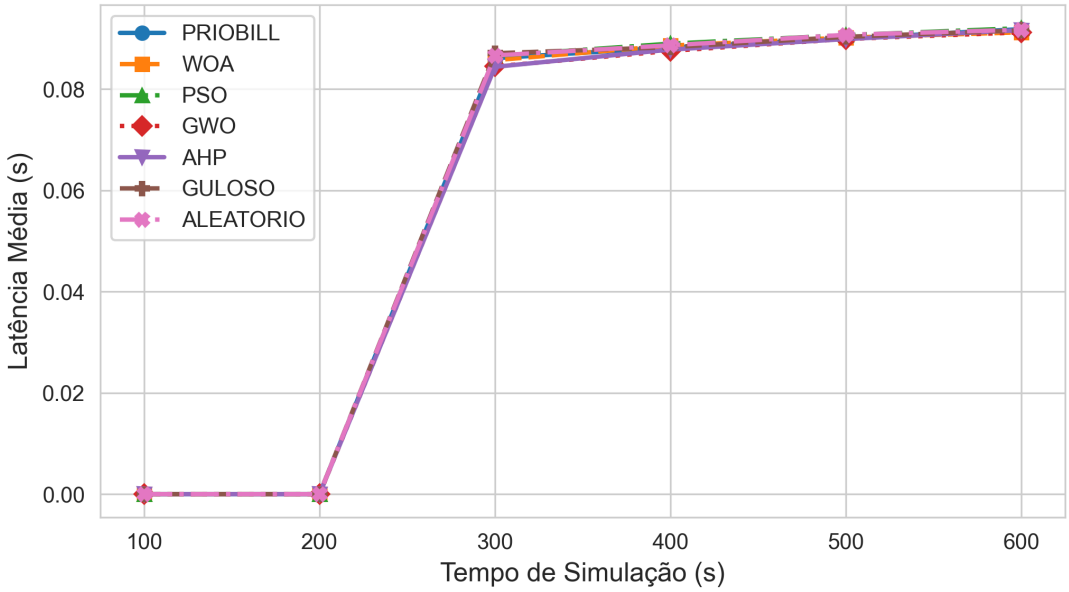
Aos 400 segundos de simulação, o GWO manteve o melhor desempenho, com latência média de 0,087573 segundos. O AHP apresentou 0,087831 segundos, seguido pelo PRIOBILL com 0,088248 segundos. O GULOSO obteve latência de 0,088305 segundos, enquanto WOA, ALEATÓRIO e PSO apresentaram valores de 0,088623, 0,088682 e 0,089003 segundos, respectivamente.

Com 500 segundos de simulação, AHP e GWO apresentaram os menores valores de latência, com 0,089906 e 0,089924 segundos, respectivamente. O PRIOBILL obteve latência média de 0,090024 segundos, seguido pelo WOA com 0,090192 segundos. O GULOSO apresentou 0,090351 segundos, enquanto PSO e ALEATÓRIO alcançaram 0,090677 e 0,090744 segundos, respectivamente.

No último intervalo avaliado, aos 600 segundos de simulação, o GWO apresentou novamente o menor tempo de processamento, com latência média de 0,091226 segundos. O WOA obteve 0,091237 segundos, seguido pelo GULOSO com 0,091649 segundos e pelo AHP com 0,091677 segundos. O ALEATÓRIO apresentou latência de 0,091759 segundos, enquanto o PRIOBILL e o PSO obtiveram 0,091870 e 0,092061 segundos, respectivamente.

Na Figura 75 apresenta-se a latência média obtida no processamento de tarefas com pesos aleatórios durante o horário da manhã. Neste cenário, observa-se que o GWO apresentou os menores valores de latência em todos os intervalos avaliados, seguido principalmente pelo WOA. O PRIOBILL manteve comportamento competitivo, apresentando valores próximos aos demais mecanismos, mesmo com a aplicação do processo de seleção baseado no LOA.

Figura 74 – Latência média para processamento de tarefas com pesos aleatórios no horário da madrugada.



Fonte: Gerado pelo autor.

Com 700 segundos de simulação, o GWO apresentou a menor latência média, com 0,091915 segundos. O WOA obteve 0,091996 segundos, seguido pelo GULOSO com 0,092295 segundos e pelo ALEATÓRIO com 0,092321 segundos. O AHP, PRIOBILL e PSO apresentaram latências de 0,092643, 0,092699 e 0,092899 segundos, respectivamente.

Aos 800 segundos de simulação, o GWO manteve o melhor desempenho, com latência de 0,092406 segundos. WOA e ALEATÓRIO apresentaram valores de 0,092517 e 0,092569 segundos, respectivamente. O GULOSO obteve 0,092828 segundos, enquanto AHP, PRIOBILL e PSO alcançaram latências de 0,093029, 0,093109 e 0,093575 segundos, respectivamente.

Com 900 segundos de simulação, o GWO apresentou novamente a menor latência, com 0,092736 segundos. O ALEATÓRIO obteve 0,092879 segundos, seguido pelo WOA com 0,092997 segundos. GULOSO, AHP, PRIOBILL e PSO apresentaram valores de 0,093281, 0,093345, 0,093521 e 0,093969 segundos, respectivamente.

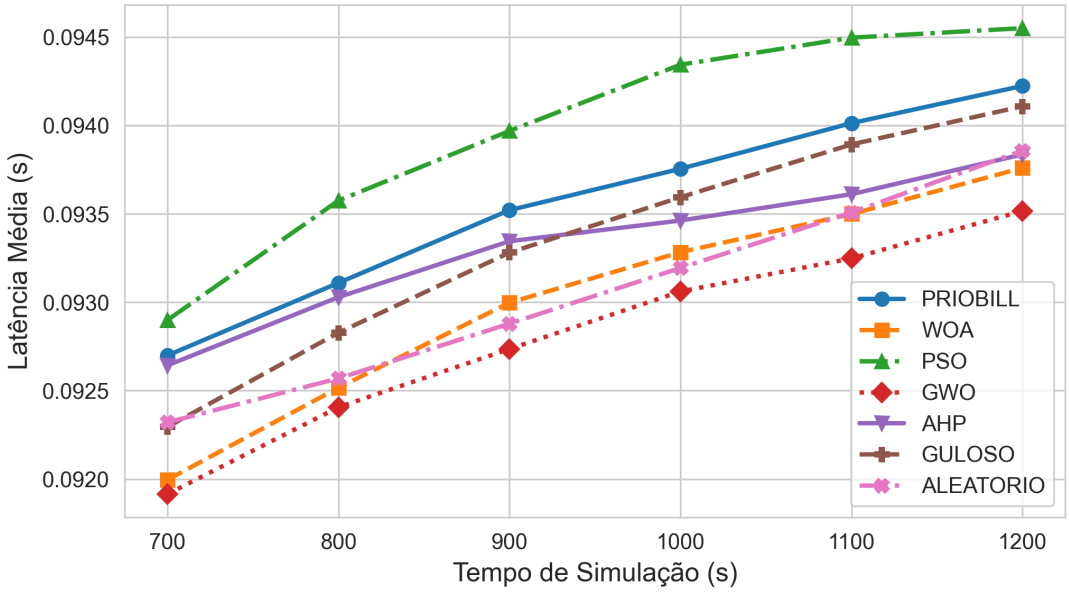
Aos 1.000 segundos de simulação, o GWO continuou apresentando o menor tempo de processamento, com 0,093061 segundos. O ALEATÓRIO apresentou 0,093194 segundos, seguido pelo WOA com 0,093283 segundos. O AHP, GULOSO, PRIOBILL e PSO obtiveram latências de 0,093462, 0,093594, 0,093754 e 0,094343 segundos, respectivamente.

Com 1.100 segundos de simulação, o GWO manteve a menor latência média, com 0,093248 segundos. O WOA apresentou 0,093499 segundos, seguido pelo ALEATÓRIO com 0,093507 segundos e pelo AHP com 0,093611 segundos. O GULOSO alcançou

0,093892 segundos, enquanto o PRIOBILL e o PSO apresentaram valores de 0,094013 e 0,094496 segundos, respectivamente.

No último intervalo avaliado, aos 1.200 segundos de simulação, o GWO apresentou latência média de 0,093516 segundos. O WOA obteve 0,093759 segundos, seguido pelo AHP com 0,093836 segundos e pelo ALEATÓRIO com 0,093857 segundos. O GULOSO apresentou 0,094108 segundos, enquanto o PRIOBILL e o PSO alcançaram 0,094223 e 0,094550 segundos, respectivamente.

Figura 75 – Latência média para processamento de tarefas com pesos aleatórios no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 76 apresenta-se a latência média obtida no processamento de tarefas com pesos aleatórios durante o horário de pico. Neste cenário, observa-se que o GWO apresentou os menores valores de latência em todos os intervalos avaliados, seguido pelo WOA e pelo AHP. O PRIOBILL manteve valores próximos aos demais mecanismos, demonstrando estabilidade no processo de seleção de nós de borda mesmo em cenários com maior densidade de veículos e maior variação dos requisitos das tarefas.

Com 1.300 segundos de simulação, o GWO apresentou a menor latência média, com 0,093755 segundos. O WOA obteve 0,093853 segundos, seguido pelo AHP com 0,093997 segundos. O ALEATÓRIO apresentou latência de 0,094077 segundos, enquanto o PRIOBILL, PSO e GULOSO obtiveram 0,094429, 0,094808 e 0,198619 segundos, respectivamente.

Aos 1.400 segundos de simulação, o GWO manteve o melhor desempenho, com latência média de 0,093951 segundos. O WOA apresentou 0,094065 segundos, seguido pelo AHP com 0,094137 segundos e pelo ALEATÓRIO com 0,094264 segundos. O PRIOBILL

obteve latência de 0,094620 segundos, enquanto PSO e GULOSO apresentaram valores de 0,094920 e 0,198679 segundos, respectivamente.

Com 1.500 segundos de simulação, o GWO e o AHP apresentaram as menores latências, com 0,094096 e 0,094250 segundos, respectivamente. O WOA obteve 0,094285 segundos, seguido pelo ALEATÓRIO com 0,094430 segundos. O PRIOBILL apresentou latência de 0,094756 segundos, enquanto PSO e GULOSO alcançaram 0,095000 e 0,198698 segundos, respectivamente.

Aos 1.600 segundos de simulação, o GWO continuou apresentando a menor latência, com 0,094252 segundos. O WOA apresentou 0,094391 segundos, seguido pelo AHP com 0,094473 segundos e pelo ALEATÓRIO com 0,094584 segundos. O PRIOBILL obteve latência média de 0,094851 segundos, enquanto PSO e GULOSO apresentaram 0,095069 e 0,198712 segundos, respectivamente.

Com 1.700 segundos de simulação, o GWO manteve o menor tempo de processamento, com latência de 0,094400 segundos. O WOA apresentou 0,094543 segundos, seguido pelo AHP com 0,094647 segundos. O ALEATÓRIO e o PRIOBILL obtiveram 0,094711 e 0,095016 segundos, respectivamente. O PSO apresentou latência de 0,095109 segundos, enquanto o GULOSO alcançou 0,198739 segundos.

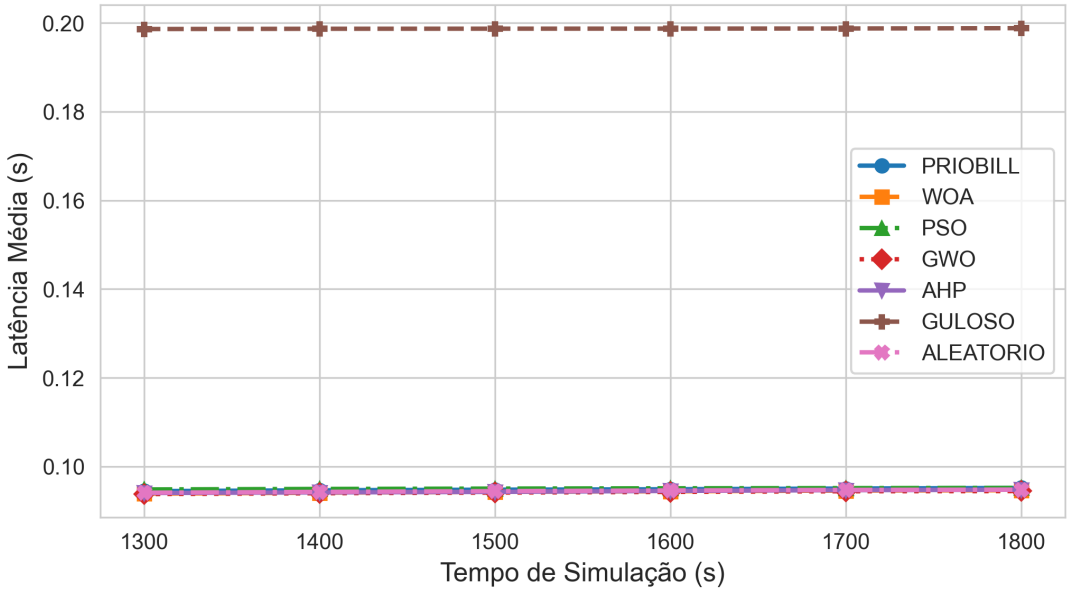
No último intervalo avaliado, aos 1.800 segundos de simulação, o GWO apresentou novamente a menor latência média, com 0,094434 segundos. O WOA obteve 0,094636 segundos, seguido pelo AHP com 0,094747 segundos e pelo ALEATÓRIO com 0,094767 segundos. O PRIOBILL apresentou latência de 0,095143 segundos, enquanto PSO e GULOSO alcançaram 0,095146 e 0,198815 segundos, respectivamente.

Na análise da métrica de latência, o PRIOBILL apresentou variações ao longo dos cenários avaliados, alternando entre momentos de melhor desempenho e intervalos com valores intermediários. Entretanto, em todos os cenários analisados, o mecanismo permaneceu próximo aos demais métodos da literatura utilizados na comparação, sem apresentar valores discrepantes de latência. Esse comportamento demonstra que o processo de seleção baseado no LOA introduz uma pequena sobrecarga computacional, porém mantém tempos de resposta adequados para o ambiente de computação de borda veicular.

No cenário com tarefas menores, o PRIOBILL iniciou a simulação com uma latência média equilibrada em relação aos demais mecanismos avaliados. Com o aumento da quantidade de veículos no cenário, apresentou melhoria no tempo de seleção e, posteriormente, manteve um desempenho intermediário quando a densidade de veículos aumentou, indicando estabilidade no processo de tomada de decisão.

Para tarefas intermediárias, a latência do PRIOBILL apresentou comportamento semelhante aos demais mecanismos no início da simulação. Durante a evolução do cenário, houve um pequeno aumento no tempo médio de seleção, porém o mecanismo recuperou

Figura 76 – Latência média para processamento de tarefas com pesos aleatórios no horário de pico.



Fonte: Gerado pelo autor.

desempenho no último terço da simulação, aproximando-se novamente dos métodos com menores valores de latência.

No cenário com tarefas maiores, o PRIOBILL apresentou latência equilibrada nos primeiros intervalos avaliados e permaneceu entre os mecanismos com valores intermediários durante o restante da simulação. Mesmo com o aumento da complexidade das tarefas e a maior exigência de recursos dos nós de borda, o mecanismo conseguiu manter tempos de seleção adequados, sem comprometer o desempenho geral do sistema.

Para tarefas com pesos aleatórios, o comportamento do PRIOBILL novamente apresentou equilíbrio em relação aos demais mecanismos. Inicialmente, obteve valores de latência próximos aos métodos comparados, apresentou uma redução de desempenho durante o período intermediário da simulação e, posteriormente, estabilizou seus resultados nos últimos intervalos avaliados.

O processo de seleção de nó de borda do PRIOBILL, baseado no algoritmo metaheurístico bioinspirado LOA, realiza uma busca iterativa pelo melhor nó disponível, considerando as etapas de movimentação, caça, ataque, migração e equilíbrio. Esse processo aumenta a capacidade de exploração do espaço de soluções e permite encontrar nós de borda mais adequados para atender às demandas das tarefas. Entretanto, devido à característica iterativa do algoritmo, em determinados momentos pode ocorrer um pequeno aumento na latência de seleção.

Apesar disso, os resultados demonstram que essa sobrecarga permanece controlada e próxima aos valores obtidos pelos mecanismos da literatura. O PRIOBILL mantém

um equilíbrio entre tempo de seleção e qualidade da decisão, conseguindo identificar nós de borda mais adequados e, consequentemente, melhorar a capacidade de atendimento das tarefas sem causar impactos significativos na latência do sistema.

4.10.5 Throughput

Nesta seção são apresentados os resultados obtidos para a métrica de throughput em todos os cenários avaliados.

Na Figura 77 apresenta-se o *throughput* obtido no processamento de tarefas menores durante o horário da madrugada. Nos primeiros 200 segundos de simulação, não ocorreram alocações devido à baixa quantidade de veículos presentes no cenário. Com 300 segundos de simulação, o PRIOBILL apresentou *throughput* de 1,0859. O AHP alcançou 1,0547, enquanto o ALEATÓRIO e o PSO apresentaram 1,0058 e 1,0072, respectivamente. O WOA e o GWO obtiveram *throughput* de 1,0033 e 0,9998, enquanto o GULOSO apresentou 0,8634.

Aos 400 segundos de simulação, o PRIOBILL manteve o maior *throughput*, atingindo 2,0576. O PSO apresentou 1,9589, seguido pelo AHP, com 1,9570, pelo WOA, com 1,9552, pelo GWO, com 1,9557, e pelo ALEATÓRIO, com 1,9563. O GULOSO apresentou *throughput* de 1,7795.

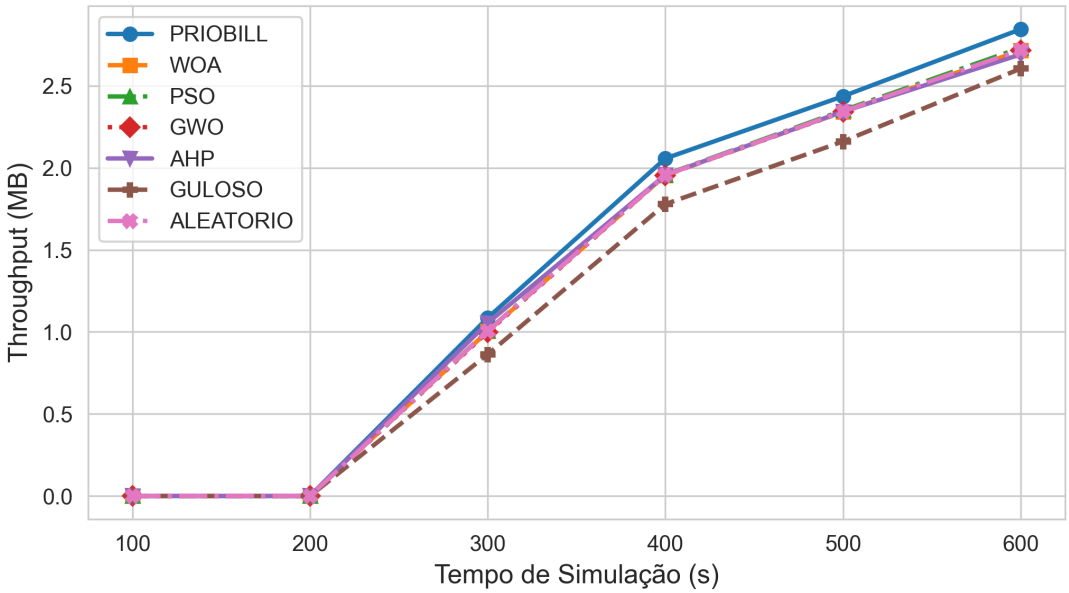
Com 500 segundos de simulação, o PRIOBILL alcançou *throughput* de 2,4368. O PSO apresentou 2,3505, seguido pelo ALEATÓRIO, com 2,3445, pelo AHP, com 2,3431, pelo WOA, com 2,3422, e pelo GWO, com 2,3415. O GULOSO obteve *throughput* de 2,1613.

No último intervalo avaliado, aos 600 segundos de simulação, o PRIOBILL apresentou *throughput* de 2,8453, mantendo o maior resultado entre os mecanismos analisados. O PSO alcançou 2,7319, enquanto o ALEATÓRIO apresentou 2,7192. O WOA, o GWO e o AHP obtiveram *throughput* de 2,7151, 2,7147 e 2,6920, respectivamente. O GULOSO apresentou 2,6052.

Na Figura 78 apresenta-se o *throughput* obtido no processamento de tarefas menores durante o horário da manhã. Neste cenário, observa-se que o PRIOBILL apresentou desempenho competitivo ao longo de todos os intervalos avaliados, mantendo-se entre os mecanismos com maior taxa de processamento e demonstrando capacidade de adaptação ao aumento da densidade de veículos no ambiente.

Com 700 segundos de simulação, o PRIOBILL apresentou *throughput* de 3,1110. O PSO alcançou 2,9770, seguido pelo WOA, com 2,9529, pelo GWO, com 2,9548, pelo ALEATÓRIO, com 2,9481, e pelo AHP, com 2,8922. O GULOSO apresentou *throughput* de 2,8748.

Figura 77 – Throughput para tarefas menores no horário da madrugada.



Fonte: Gerado pelo autor.

Aos 800 segundos de simulação, o PRIOBILL atingiu *throughput* de 3,3622. O PSO apresentou 3,1966, enquanto o GULOSO alcançou 3,2077. O GWO, o WOA, o AHP e o ALEATÓRIO obtiveram *throughput* de 3,1795, 3,1753, 3,1767 e 3,1703, respectivamente.

Com 900 segundos de simulação, o GULOSO apresentou *throughput* de 3,4447, enquanto o PRIOBILL alcançou 3,4141. O PSO obteve 3,3032, seguido pelo GWO, com 3,2934, pelo WOA, com 3,2892, e pelo ALEATÓRIO, com 3,2855. O AHP apresentou *throughput* de 3,2530.

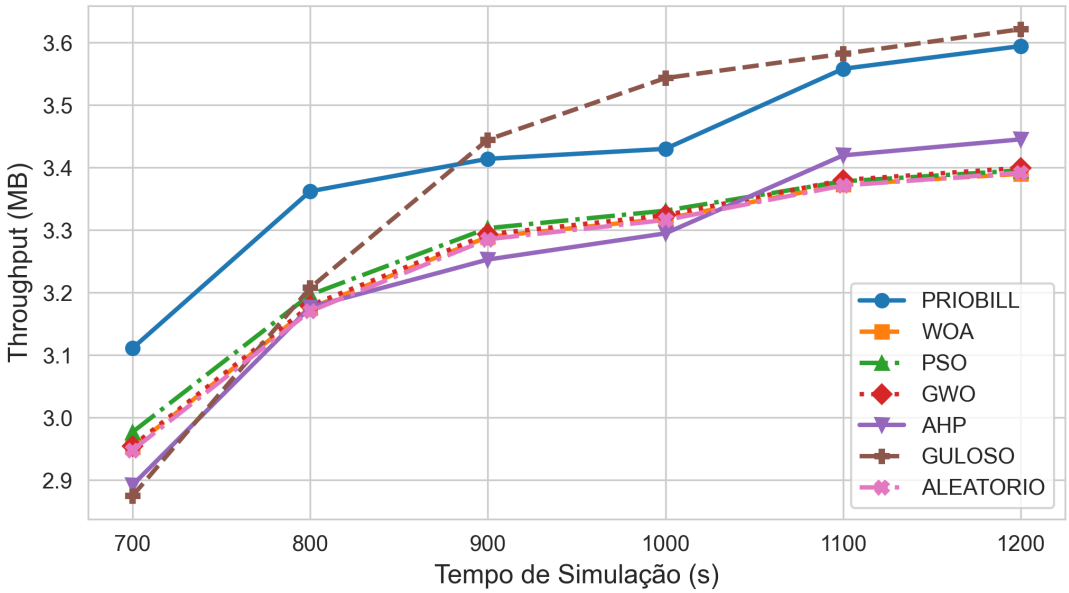
Aos 1.000 segundos de simulação, o GULOSO manteve o maior *throughput*, com 3,5433. O PRIOBILL apresentou 3,4301, enquanto o PSO alcançou 3,3312. O GWO, o ALEATÓRIO, o WOA e o AHP obtiveram *throughput* de 3,3240, 3,3162, 3,3185 e 3,2948, respectivamente.

Com 1.100 segundos de simulação, o GULOSO apresentou *throughput* de 3,5821, seguido pelo PRIOBILL, com 3,5581. O AHP alcançou 3,4196, enquanto o GWO apresentou 3,3802. O PSO, o WOA e o ALEATÓRIO obtiveram *throughput* de 3,3781, 3,3729 e 3,3714, respectivamente.

No último intervalo avaliado, aos 1.200 segundos de simulação, o GULOSO alcançou *throughput* de 3,6214, mantendo o maior resultado. O PRIOBILL apresentou 3,5944, enquanto o AHP obteve 3,4453. O GWO, o PSO, o ALEATÓRIO e o WOA apresentaram *throughput* de 3,3997, 3,3956, 3,3919 e 3,3901, respectivamente.

Na Figura 79 apresenta-se o *throughput* obtido no processamento de tarefas menores durante o horário de pico. Neste cenário, observa-se que o PRIOBILL apresentou

Figura 78 – Throughput para tarefas menores no horário da manhã.



Fonte: Gerado pelo autor.

desempenho competitivo ao longo de todos os intervalos avaliados, permanecendo próximo ao mecanismo com maior *throughput* e demonstrando capacidade de adaptação mesmo em um ambiente com maior quantidade de veículos e maior demanda de processamento.

Com 1.300 segundos de simulação, o GULOSO apresentou o maior *throughput*, atingindo 3,6881. O PRIOBILL alcançou 3,6559, enquanto o WOA apresentou 3,4989. O GWO, o ALEATÓRIO e o PSO obtiveram *throughput* de 3,5094, 3,5066 e 3,5089, respectivamente. O AHP apresentou 3,5448.

Aos 1.400 segundos de simulação, o GULOSO manteve o maior *throughput*, com 3,7078. O PRIOBILL apresentou 3,6408, seguido pelo AHP, com 3,5810. O GWO, o ALEATÓRIO, o PSO e o WOA alcançaram *throughput* de 3,5564, 3,5562, 3,5523 e 3,5432, respectivamente.

Com 1.500 segundos de simulação, o GULOSO atingiu *throughput* de 3,7590. O PRIOBILL apresentou 3,6772, enquanto o GWO alcançou 3,6130. O ALEATÓRIO, o PSO e o WOA obtiveram *throughput* de 3,6109, 3,6056 e 3,6033, respectivamente. O AHP apresentou 3,5806.

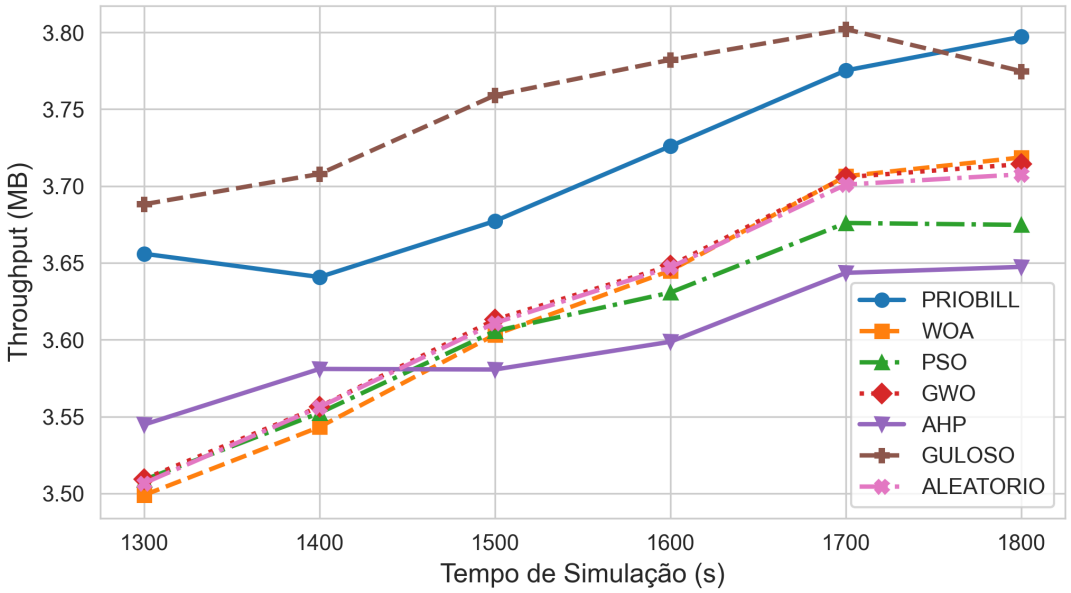
Aos 1.600 segundos de simulação, o GULOSO apresentou *throughput* de 3,7821, enquanto o PRIOBILL alcançou 3,7259. O GWO obteve 3,6482, seguido pelo ALEATÓRIO, com 3,6471, pelo WOA, com 3,6450, e pelo PSO, com 3,6307. O AHP apresentou *throughput* de 3,5987.

Com 1.700 segundos de simulação, o GULOSO alcançou *throughput* de 3,8020, seguido pelo PRIOBILL, com 3,7752. O WOA apresentou 3,7064, enquanto o GWO

obteve 3,7058. O ALEATÓRIO e o PSO alcançaram 3,7009 e 3,6760, respectivamente. O AHP apresentou *throughput* de 3,6435.

No último intervalo avaliado, aos 1.800 segundos de simulação, o GULOSO apresentou *throughput* de 3,7747. O PRIOBILL alcançou 3,7971, superando os demais mecanismos. O WOA apresentou 3,7186, enquanto o GWO obteve 3,7143. O ALEATÓRIO, o AHP e o PSO apresentaram *throughput* de 3,7076, 3,6474 e 3,6747, respectivamente.

Figura 79 – Throughput para tarefas menores no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 80 apresenta-se o *throughput* obtido no processamento de tarefas intermediárias durante o horário da madrugada. Nos primeiros 200 segundos de simulação, não ocorreram alocações devido à baixa quantidade de veículos presentes no cenário. Com 300 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,5248. O AHP alcançou 0,5046, enquanto o ALEATÓRIO e o WOA obtiveram 0,4668 e 0,4661, respectivamente. O GWO apresentou 0,4616, seguido pelo PSO, com 0,4476, e pelo GULOSO, com 0,3406.

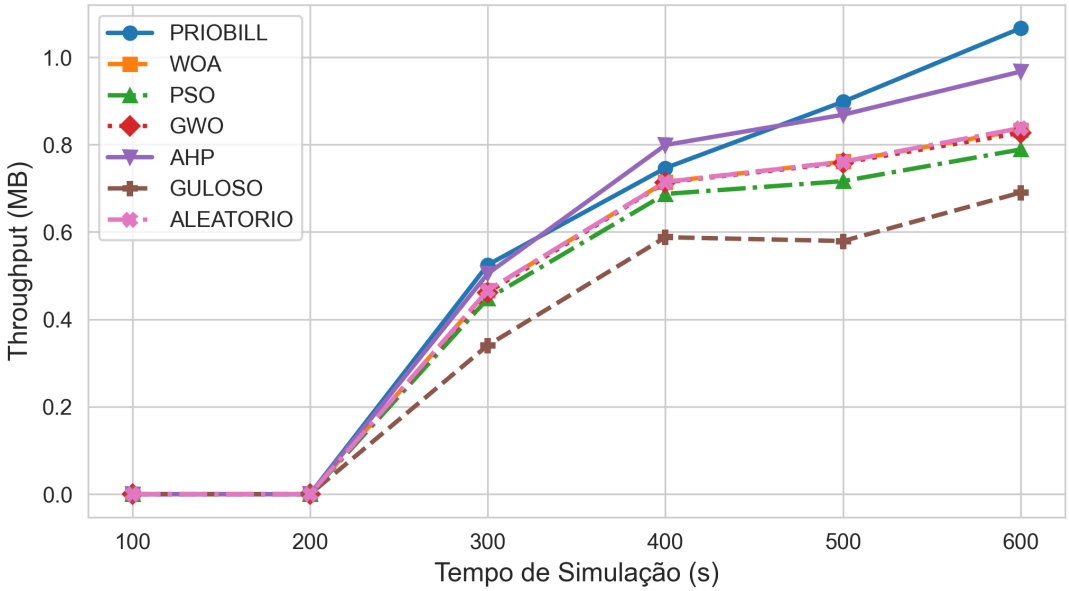
Aos 400 segundos de simulação, o AHP apresentou o maior *throughput*, atingindo 0,7992. O PRIOBILL alcançou 0,7464, enquanto o WOA, o ALEATÓRIO e o GWO apresentaram 0,7149, 0,7146 e 0,7136, respectivamente. O PSO obteve 0,6870, seguido pelo GULOSO, com 0,5880.

Com 500 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,8981, alcançando o maior resultado. O AHP obteve 0,8684, enquanto o WOA apresentou 0,7610. O ALEATÓRIO e o GWO alcançaram 0,7608 e 0,7585, respectivamente. O PSO apresentou 0,7164, seguido pelo GULOSO, com 0,5793.

No último intervalo avaliado, aos 600 segundos de simulação, o PRIOBILL manteve o maior *throughput*, com 1,0666. O AHP apresentou 0,9672, enquanto o ALEATÓRIO

alcançou 0,8380. O WOA, o GWO e o PSO obtiveram *throughput* de 0,8336, 0,8268 e 0,7890, respectivamente. O GULOSO apresentou 0,6903.

Figura 80 – Throughput para tarefas intermediárias no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 81 apresenta-se o *throughput* obtido no processamento de tarefas intermediárias durante o horário da manhã. Neste cenário, o PRIOBILL apresentou desempenho superior aos demais mecanismos avaliados em todos os intervalos de simulação, demonstrando maior capacidade de aproveitamento dos recursos disponíveis e maior eficiência no atendimento das tarefas intermediárias em um ambiente com maior densidade veicular.

Com 700 segundos de simulação, o PRIOBILL apresentou *throughput* de 1,1420. O AHP alcançou 0,9568, enquanto o GWO obteve 0,8803. O ALEATÓRIO e o WOA apresentaram 0,8768 e 0,8740, respectivamente. O PSO alcançou 0,8402, seguido pelo GULOSO, com 0,7599.

Aos 800 segundos de simulação, o PRIOBILL atingiu *throughput* de 1,3341. O AHP apresentou 1,0185, enquanto o GWO alcançou 1,0025. O WOA, o ALEATÓRIO e o PSO obtiveram *throughput* de 1,0007, 1,0000 e 0,9608, respectivamente. O GULOSO apresentou 0,9323.

Com 900 segundos de simulação, o PRIOBILL apresentou *throughput* de 1,2906. O GWO alcançou 1,0397, seguido pelo ALEATÓRIO, com 1,0364, pelo WOA, com 1,0339, pelo AHP, com 1,0337, e pelo GULOSO, com 1,0274. O PSO apresentou *throughput* de 1,0008.

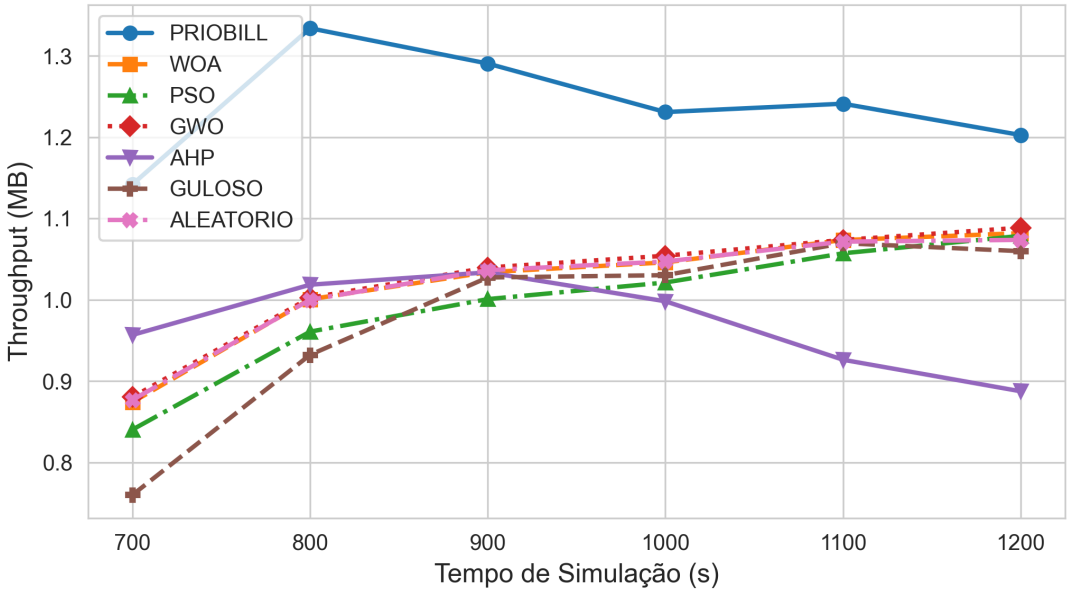
Aos 1.000 segundos de simulação, o PRIOBILL atingiu *throughput* de 1,2309. O GWO apresentou 1,0542, enquanto o ALEATÓRIO alcançou 1,0471. O WOA obteve

1,0460, seguido pelo GULOSO, com 1,0303, pelo PSO, com 1,0212, e pelo AHP, com 0,9981.

Com 1.100 segundos de simulação, o PRIOBILL apresentou *throughput* de 1,2412. O WOA alcançou 1,0735, enquanto o GWO obteve 1,0737. O ALEATÓRIO apresentou 1,0715, seguido pelo GULOSO, com 1,0696, pelo PSO, com 1,0572, e pelo AHP, com 0,9262.

No último intervalo avaliado, aos 1.200 segundos de simulação, o PRIOBILL atingiu *throughput* de 1,2027. O GWO apresentou 1,0887, enquanto o WOA alcançou 1,0819. O PSO, o ALEATÓRIO e o GULOSO obtiveram *throughput* de 1,0786, 1,0737 e 1,0598, respectivamente. O AHP apresentou 0,8874.

Figura 81 – Throughput para tarefas intermediárias no horário da manhã.



Fonte: Gerado pelo autor.

Na Figura 82 apresenta-se o *throughput* obtido no processamento de tarefas intermediárias durante o horário de pico. Neste cenário, observa-se que o PRIOBILL apresentou desempenho competitivo ao longo dos intervalos avaliados, mantendo-se entre os mecanismos com maior taxa de processamento e alcançando o melhor resultado nos últimos intervalos da simulação.

Com 1.300 segundos de simulação, o PRIOBILL apresentou *throughput* de 1,1660. O WOA alcançou 1,1249, enquanto o PSO obteve 1,1195. O ALEATÓRIO apresentou 1,1185, seguido pelo GWO, com 1,1314, pelo GULOSO, com 1,0897, e pelo AHP, com 0,8738.

Aos 1.400 segundos de simulação, o PRIOBILL apresentou *throughput* de 1,1200. O GULOSO alcançou 1,1087, enquanto o GWO obteve 1,1465. O WOA, o ALEATÓRIO

e o PSO apresentaram 1,1418, 1,1392 e 1,1341, respectivamente. O AHP apresentou 0,8776.

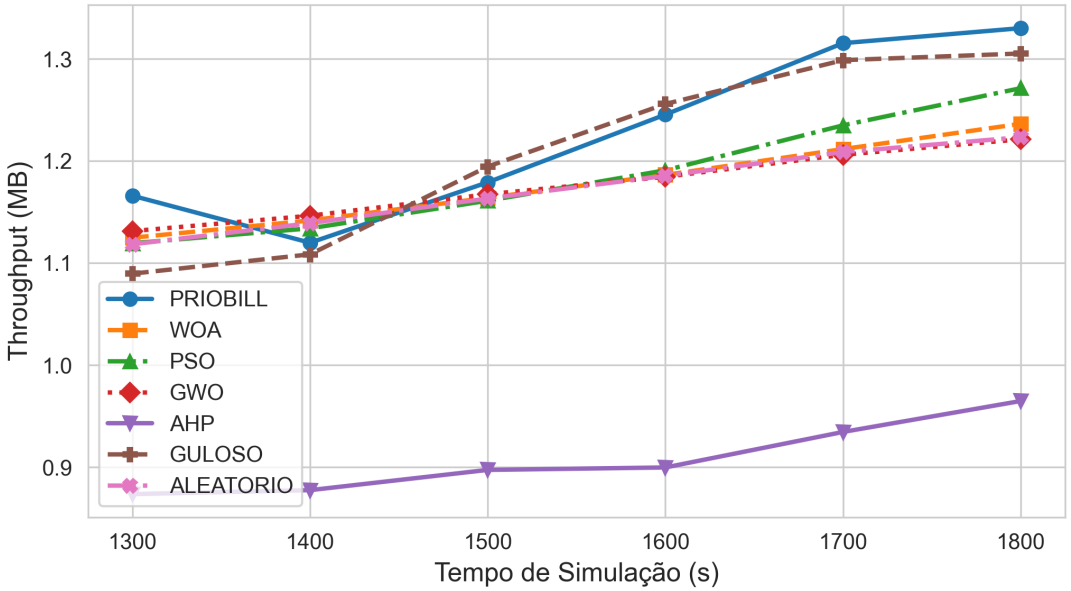
Com 1.500 segundos de simulação, o GULOSO atingiu *throughput* de 1,1946, enquanto o PRIOBILL apresentou 1,1792. O GWO alcançou 1,1675, seguido pelo WOA, com 1,1641, pelo ALEATÓRIO, com 1,1634, e pelo PSO, com 1,1612. O AHP apresentou 0,8975.

Aos 1.600 segundos de simulação, o GULOSO apresentou *throughput* de 1,2559. O PRIOBILL alcançou 1,2455, enquanto o PSO obteve 1,1910. O WOA, o ALEATÓRIO e o GWO apresentaram *throughput* de 1,1865, 1,1855 e 1,1846, respectivamente. O AHP apresentou 0,8999.

Com 1.700 segundos de simulação, o PRIOBILL atingiu *throughput* de 1,3155. O GULOSO apresentou 1,2988, enquanto o PSO alcançou 1,2350. O WOA obteve 1,2117, seguido pelo ALEATÓRIO, com 1,2088, e pelo GWO, com 1,2061. O AHP apresentou 0,9347.

No último intervalo avaliado, aos 1.800 segundos de simulação, o PRIOBILL apresentou *throughput* de 1,3302. O GULOSO alcançou 1,3054, enquanto o PSO obteve 1,2714. O WOA apresentou 1,2365, seguido pelo ALEATÓRIO, com 1,2234, pelo GWO, com 1,2213, e pelo AHP, com 0,9650.

Figura 82 – Throughput para tarefas intermediárias no horário de pico.



Fonte: Gerado pelo autor.

Na Figura 83 apresenta-se o *throughput* obtido no processamento de tarefas maiores durante o horário da madrugada. Neste cenário, observa-se uma maior dificuldade dos mecanismos avaliados devido ao elevado custo computacional das tarefas, principalmente

nos primeiros intervalos da simulação, quando a quantidade de veículos disponíveis ainda é reduzida.

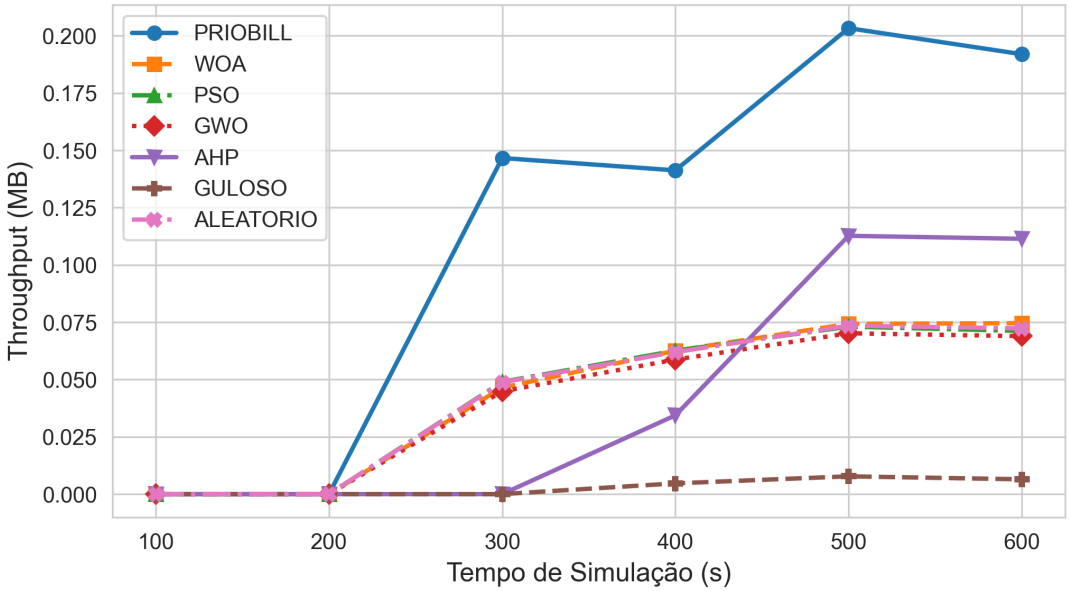
Nos primeiros 200 segundos de simulação, não ocorreram alocações devido à baixa quantidade de veículos presentes no cenário. Com 300 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,1467, enquanto o PSO alcançou 0,0490. O ALEATÓRIO apresentou 0,0488, seguido pelo WOA, com 0,0465, e pelo GWO, com 0,0448. O AHP e o GULOSO não realizaram alocações, apresentando *throughput* igual a zero.

Aos 400 segundos de simulação, o PRIOBILL manteve o maior resultado, com *throughput* de 0,1413. O WOA e o PSO apresentaram 0,0626, enquanto o ALEATÓRIO alcançou 0,0620. O GWO obteve 0,0587, seguido pelo AHP, com 0,0344, e pelo GULOSO, com 0,0047.

Com 500 segundos de simulação, o PRIOBILL novamente apresentou o maior *throughput*, atingindo 0,2033. O AHP alcançou 0,1127, enquanto o WOA apresentou 0,0743. O ALEATÓRIO, o PSO e o GWO obtiveram *throughput* de 0,0735, 0,0731 e 0,0702, respectivamente. O GULOSO apresentou 0,0078.

No último intervalo avaliado, aos 600 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,1920, mantendo o maior resultado entre os mecanismos analisados. O AHP alcançou 0,1114, seguido pelo WOA, com 0,0746, pelo ALEATÓRIO, com 0,0723, pelo PSO, com 0,0712, e pelo GWO, com 0,0689. O GULOSO apresentou *throughput* de 0,0065.

Figura 83 – Throughput para tarefas maiores no horário da madrugada.



Fonte: Gerado pelo autor.

Na Figura 84 apresenta-se o *throughput* obtido para tarefas maiores no horário da manhã. Com 700 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,1680.

O AHP alcançou 0,0955, enquanto o ALEATÓRIO obteve 0,0647. O WOA, o PSO e o GWO apresentaram *throughput* de 0,0664, 0,0637 e 0,0619, respectivamente. O GULOSO apresentou 0,0143.

Aos 800 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,1470. O AHP apresentou 0,0836, enquanto o ALEATÓRIO alcançou 0,0597. O WOA, o PSO e o GWO obtiveram *throughput* de 0,0612, 0,0589 e 0,0572, respectivamente. O GULOSO apresentou 0,0125.

Com 900 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,1307. O AHP alcançou 0,0743, enquanto o WOA obteve 0,0544. O ALEATÓRIO apresentou 0,0532, seguido pelo PSO, com 0,0525, e pelo GWO, com 0,0508. O GULOSO apresentou *throughput* de 0,0111.

Aos 1.000 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,1176. O AHP apresentou 0,0668, enquanto o WOA alcançou 0,0495. O ALEATÓRIO obteve 0,0486, seguido pelo PSO, com 0,0477, e pelo GWO, com 0,0463. O GULOSO apresentou 0,0108.

Com 1.100 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,1069. O AHP alcançou 0,0608, enquanto o WOA obteve 0,0459. O ALEATÓRIO apresentou 0,0453, seguido pelo PSO, com 0,0442, e pelo GWO, com 0,0429. O GULOSO apresentou *throughput* de 0,0100.

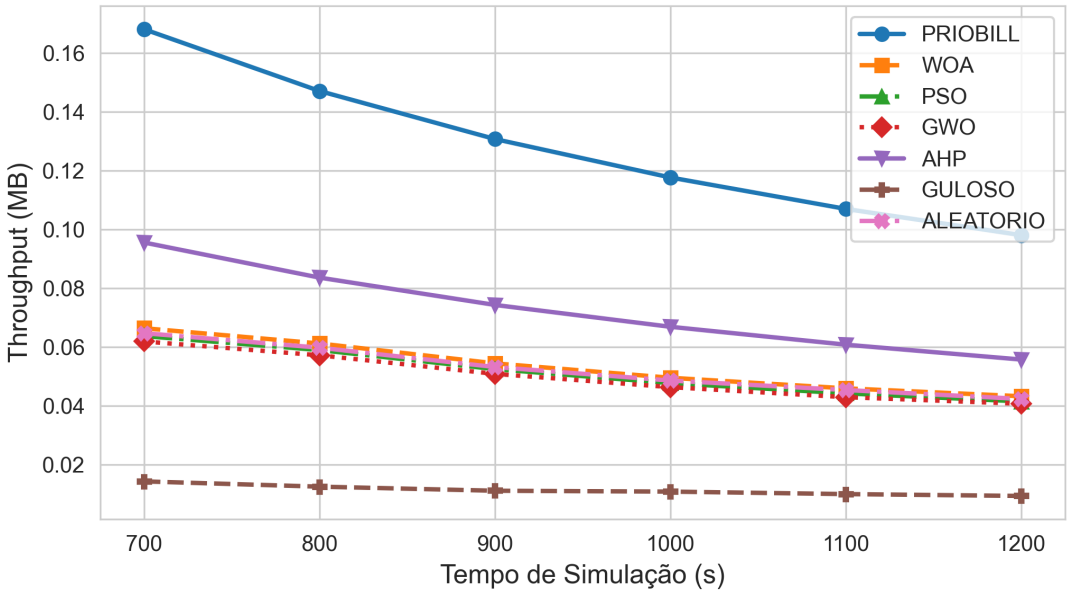
No último intervalo avaliado, aos 1.200 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,0980. O AHP apresentou 0,0557, enquanto o WOA alcançou 0,0432. O ALEATÓRIO obteve 0,0423, seguido pelo PSO, com 0,0415, e pelo GWO, com 0,0407. O GULOSO apresentou 0,0093.

Na Figura 85 apresenta-se o *throughput* obtido para tarefas maiores no horário de pico. Aos 1.300 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,0905. O AHP alcançou 0,0514, enquanto o GWO obteve 0,0406. O WOA apresentou 0,0431, seguido pelo ALEATÓRIO, com 0,0425, e pelo PSO, com 0,0406. O GULOSO apresentou *throughput* de 0,0129.

Aos 1.400 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,0870. O AHP apresentou 0,0485, enquanto o GWO alcançou 0,0446. O ALEATÓRIO obteve 0,0439, seguido pelo WOA, com 0,0435, pelo PSO, com 0,0414, e pelo GULOSO, com 0,0185.

Com 1.500 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,0819. O GWO alcançou 0,0484, enquanto o AHP obteve 0,0476. O ALEATÓRIO apresentou 0,0446, seguido pelo WOA, com 0,0438, pelo PSO, com 0,0422, e pelo GULOSO, com 0,0217.

Figura 84 – Throughput para tarefas maiores no horário da manhã.



Fonte: Gerado pelo autor.

Aos 1.600 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,0769. O GWO apresentou 0,0510, enquanto o AHP alcançou 0,0495. O ALEATÓRIO obteve 0,0453, seguido pelo WOA, com 0,0438, pelo PSO, com 0,0426, e pelo GULOSO, com 0,0276.

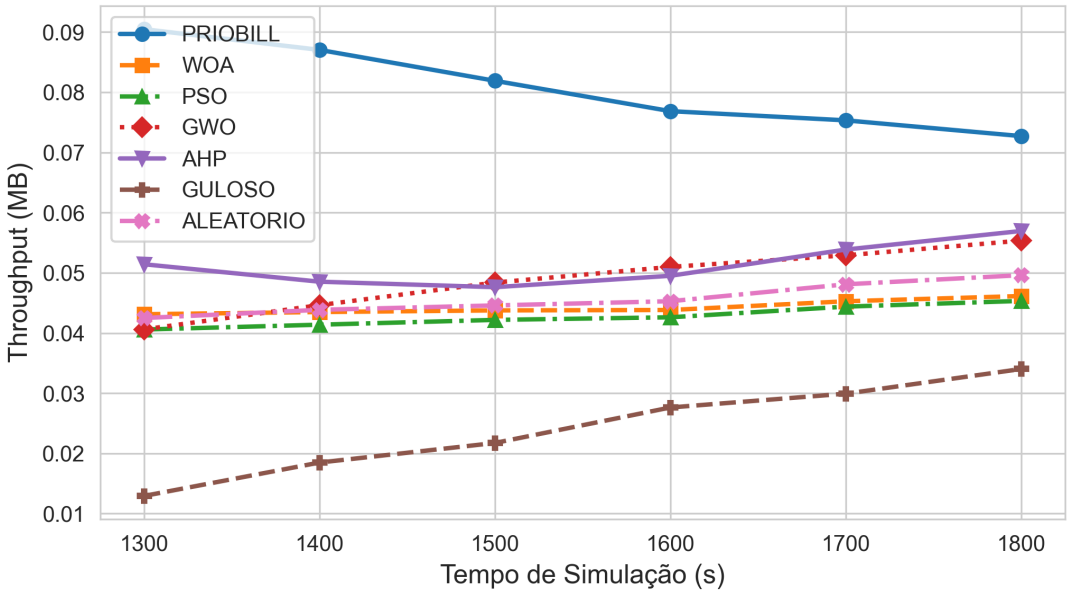
Com 1.700 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,0753. O AHP alcançou 0,0539, enquanto o GWO obteve 0,0529. O ALEATÓRIO apresentou 0,0481, seguido pelo WOA, com 0,0453, pelo PSO, com 0,0444, e pelo GULOSO, com 0,0299.

No último intervalo avaliado, aos 1.800 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,0727. O AHP apresentou 0,0570, enquanto o GWO alcançou 0,0554. O ALEATÓRIO obteve 0,0496, seguido pelo WOA, com 0,0461, pelo PSO, com 0,0453, e pelo GULOSO, com 0,0340.

Na Figura 86 apresenta-se o *throughput* obtido para tarefas com pesos aleatórios no horário da madrugada. Nos primeiros 200 segundos de simulação, não ocorreram alocações devido à baixa quantidade de veículos presentes no cenário. Com 300 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,3835. O AHP alcançou 0,1400, enquanto o GULOSO obteve 0,2151. O ALEATÓRIO, o PSO, o WOA e o GWO apresentaram *throughput* de 0,2429, 0,2469, 0,2387 e 0,2364, respectivamente.

Aos 400 segundos de simulação, o PRIOBILL manteve o maior *throughput*, atingindo 0,4873. O AHP apresentou 0,2990, enquanto o GULOSO alcançou 0,3583. O PSO obteve 0,3717, seguido pelo ALEATÓRIO, com 0,3707, pelo WOA, com 0,3698, e pelo GWO, com 0,3667.

Figura 85 – Throughput para tarefas maiores no horário de pico.



Fonte: Gerado pelo autor.

Com 500 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,6153. O AHP alcançou 0,3344, enquanto o GULOSO obteve 0,4237. O WOA, o PSO, o GWO e o ALEATÓRIO apresentaram *throughput* de 0,4293, 0,4290, 0,4272 e 0,4281, respectivamente.

No último intervalo avaliado, aos 600 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,6612. O WOA apresentou 0,4687, enquanto o ALEATÓRIO alcançou 0,4693. O GWO obteve 0,4682, seguido pelo PSO, com 0,4664, e pelo GULOSO, com 0,4652. O AHP apresentou *throughput* de 0,4119.

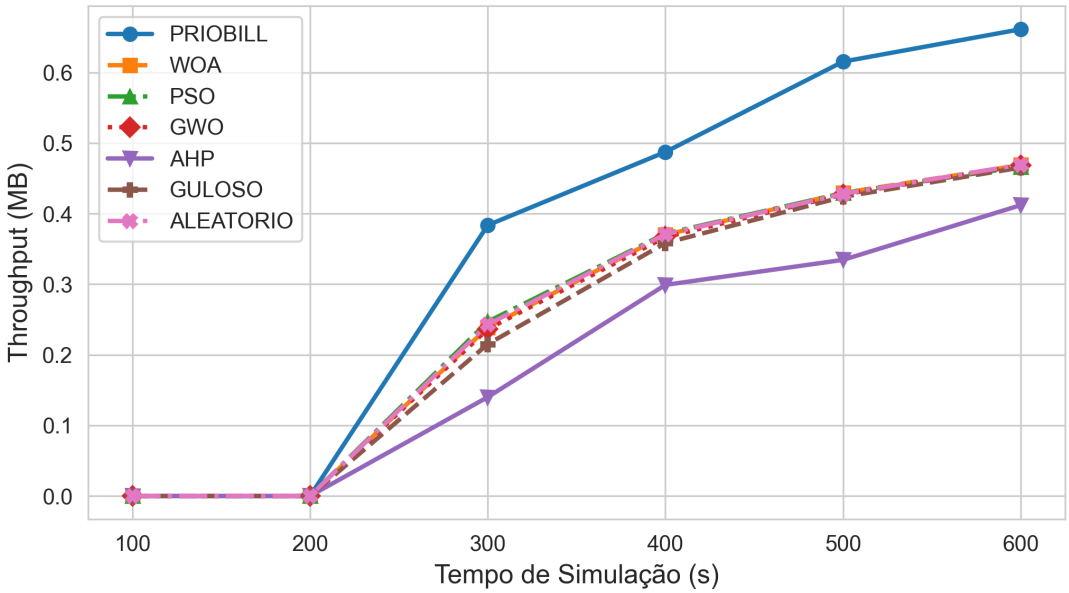
Na Figura 87 apresenta-se o *throughput* para tarefas com pesos aleatórios no horário da manhã. De maneira geral, o PRIOBILL apresentou desempenho superior na maior parte dos intervalos avaliados, demonstrando maior capacidade de processamento das tarefas quando comparado aos mecanismos analisados.

Com 700 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,6757. O GULOSO alcançou 0,4962, enquanto o GWO obteve 0,4956. O WOA, o ALEATÓRIO e o PSO apresentaram *throughput* de 0,4934, 0,4917 e 0,4913, respectivamente. O AHP apresentou 0,4865.

Aos 800 segundos de simulação, o PRIOBILL atingiu *throughput* de 0,6944. O AHP apresentou 0,5990, enquanto o GWO alcançou 0,5319. O WOA, o PSO e o ALEATÓRIO obtiveram *throughput* de 0,5299, 0,5292 e 0,5282, respectivamente. O GULOSO apresentou 0,4932.

Com 900 segundos de simulação, o PRIOBILL apresentou *throughput* de 0,6646.

Figura 86 – Throughput para tarefas com pesos aleatórios no horário da madrugada.



Fonte: Gerado pelo autor.

O AHP alcançou 0,6441, enquanto o GWO obteve 0,5479. O ALEATÓRIO apresentou 0,5431, seguido pelo WOA, com 0,5421, e pelo PSO, com 0,5402. O GULOSO apresentou *throughput* de 0,5037.

Aos 1.000 segundos de simulação, o AHP apresentou o maior *throughput*, atingindo 0,7157. O PRIOBILL alcançou 0,6411, enquanto o GWO obteve 0,5639. O ALEATÓRIO apresentou 0,5602, seguido pelo WOA, com 0,5541, pelo PSO, com 0,5512, e pelo GULOSO, com 0,5190.

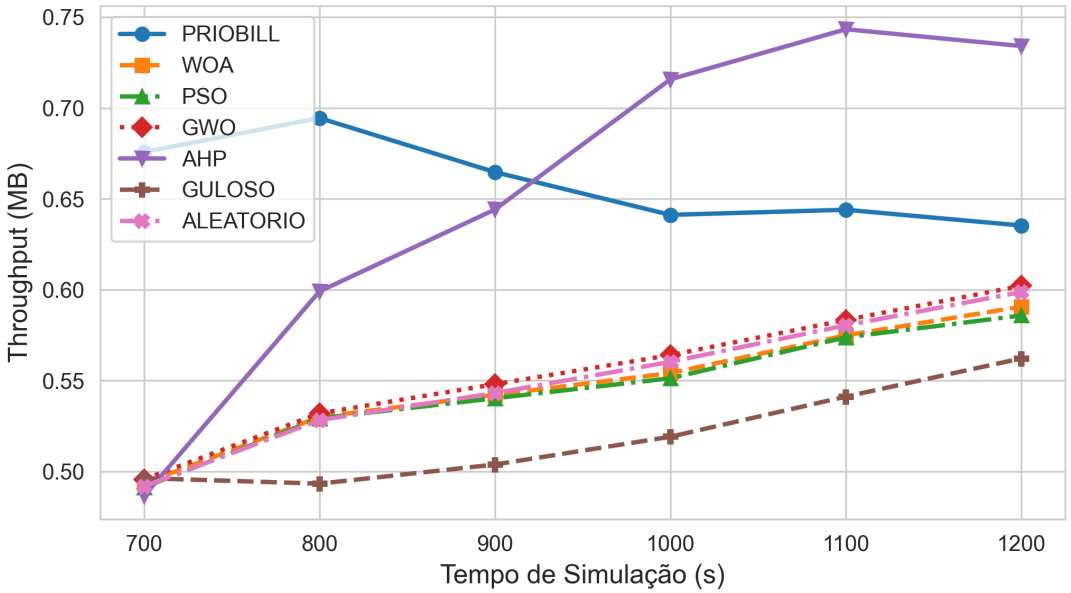
Com 1.100 segundos de simulação, o AHP manteve o maior *throughput*, com 0,7433. O PRIOBILL apresentou 0,6439, enquanto o GWO alcançou 0,5833. O ALEATÓRIO obteve 0,5803, seguido pelo WOA, com 0,5748, pelo PSO, com 0,5737, e pelo GULOSO, com 0,5411.

No último intervalo avaliado, aos 1.200 segundos de simulação, o AHP apresentou *throughput* de 0,7340. O PRIOBILL alcançou 0,6353, enquanto o GWO obteve 0,6021. O ALEATÓRIO apresentou 0,5987, seguido pelo WOA, com 0,5906, pelo PSO, com 0,5858, e pelo GULOSO, com 0,5621.

Na Figura 88 é apresentado o *throughput* das tarefas com pesos aleatórios no horário de pico. Aos 1.300 segundos de simulação, o AHP apresentou o maior *throughput*, atingindo 0,7352. O PRIOBILL alcançou 0,6311, enquanto o ALEATÓRIO obteve 0,6286. O GWO apresentou 0,6285, seguido pelo WOA, com 0,6187, pelo PSO, com 0,6125, e pelo GULOSO, com 0,5771.

Aos 1.400 segundos de simulação, o AHP manteve o maior *throughput*, com

Figura 87 – Throughput para tarefas com pesos aleatórios no horário da manhã.



Fonte: Gerado pelo autor.

0,7153. O ALEATÓRIO apresentou 0,6465, enquanto o GWO alcançou 0,6454. O WOA obteve 0,6340, seguido pelo PSO, com 0,6294, pelo PRIOBILL, com 0,6147, e pelo GULOSO, com 0,5807.

Com 1.500 segundos de simulação, o AHP apresentou *throughput* de 0,7174. O ALEATÓRIO alcançou 0,6689, enquanto o GWO obteve 0,6655. O WOA apresentou 0,6533, seguido pelo PSO, com 0,6461, pelo PRIOBILL, com 0,6176, e pelo GULOSO, com 0,5859.

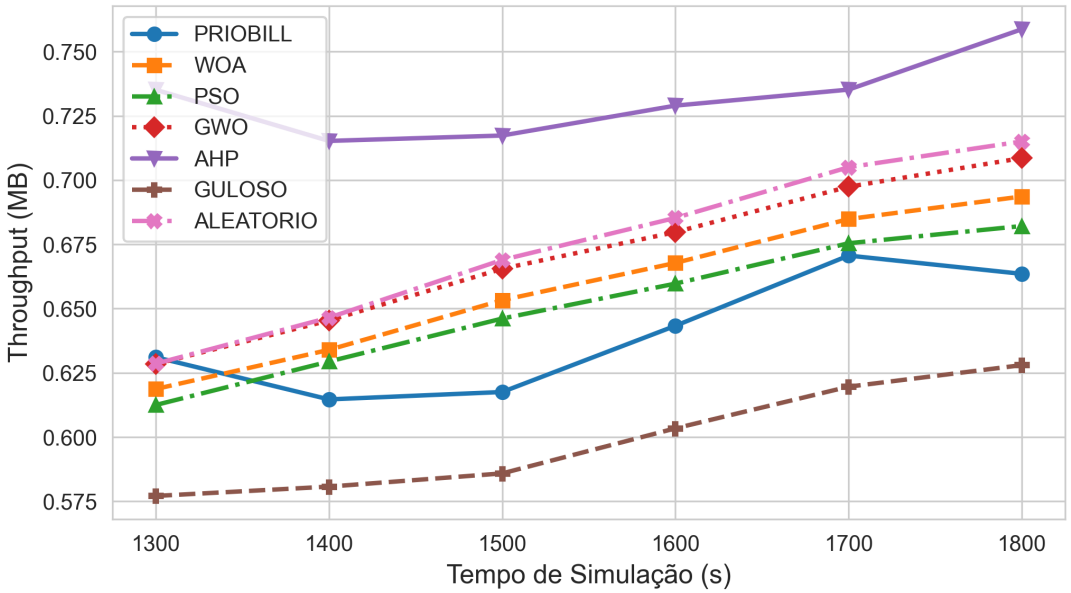
Aos 1.600 segundos de simulação, o AHP atingiu *throughput* de 0,7290. O ALEATÓRIO apresentou 0,6853, enquanto o GWO alcançou 0,6798. O WOA obteve 0,6678, seguido pelo PSO, com 0,6597, pelo PRIOBILL, com 0,6433, e pelo GULOSO, com 0,6034.

Com 1.700 segundos de simulação, o AHP apresentou *throughput* de 0,7352. O ALEATÓRIO alcançou 0,7050, enquanto o GWO obteve 0,6974. O WOA apresentou 0,6848, seguido pelo PSO, com 0,6754, pelo PRIOBILL, com 0,6706, e pelo GULOSO, com 0,6196.

No último intervalo avaliado, aos 1.800 segundos de simulação, o AHP atingiu *throughput* de 0,7587, mantendo o maior resultado entre os mecanismos analisados. O ALEATÓRIO apresentou 0,7151, enquanto o GWO alcançou 0,7086. O WOA obteve 0,6936, seguido pelo PSO, com 0,6821, pelo PRIOBILL, com 0,6635, e pelo GULOSO, com 0,6280.

Na avaliação do *throughput*, métrica responsável por mensurar a taxa de processa-

Figura 88 – Throughput para tarefas com pesos aleatórios no horário de pico.



Fonte: Gerado pelo autor.

mento e transferência das tarefas no sistema, o PRIOBILL apresentou desempenho competitivo em todos os cenários analisados. No cenário com tarefas menores, o PRIOBILL obteve o melhor desempenho durante os intervalos avaliados no horário da madrugada. No período da manhã, iniciou a simulação com vantagem em relação aos demais mecanismos e, posteriormente, manteve-se próximo ao mecanismo com melhor desempenho. Durante o horário de pico, o PRIOBILL permaneceu equilibrado em relação aos demais algoritmos, alcançando o melhor resultado no último intervalo de simulação.

Para as tarefas intermediárias, o PRIOBILL iniciou o cenário da madrugada com desempenho semelhante aos demais mecanismos, porém apresentou evolução ao longo da simulação, finalizando esse período com o melhor *throughput*. No horário da manhã, o mecanismo destacou-se significativamente, obtendo os maiores valores de *throughput* na maioria dos intervalos avaliados. Já no horário de pico, manteve desempenho competitivo nos primeiros intervalos e novamente alcançou o melhor resultado ao final da simulação.

No cenário composto por tarefas maiores, o PRIOBILL apresentou desempenho superior aos demais mecanismos no primeiro período de simulação. Embora todos os algoritmos tenham enfrentado maiores dificuldades de alocação devido à maior demanda de recursos das tarefas, o PRIOBILL conseguiu manter o melhor aproveitamento dos recursos disponíveis, resultando em maiores taxas de processamento.

Para o cenário com tarefas de pesos aleatórios, o PRIOBILL iniciou com o melhor desempenho no horário da madrugada. Entretanto, com o aumento da densidade de veículos e da complexidade do cenário nos períodos da manhã e de pico, apresentou uma redução relativa de desempenho, mantendo-se próximo aos mecanismos com melhores

resultados.

De maneira geral, o PRIOBILL apresentou o melhor comportamento considerando todos os cenários avaliados, demonstrando capacidade de adaptação às diferentes características das tarefas e condições da rede veicular. Mesmo nos casos em que não alcançou o maior *throughput*, permaneceu próximo aos melhores resultados obtidos pelos mecanismos comparados. Esse comportamento está relacionado à capacidade adaptativa do processo de seleção de nós de borda, implementado por meio do algoritmo metaheurístico LOA. As etapas de movimentação, caça, ataque, migração e equilíbrio permitem uma busca contínua por soluções eficientes, favorecendo a seleção de nós com maior capacidade de atendimento e mantendo o mecanismo competitivo diante das variações de demanda e disponibilidade de recursos.

4.10.6 Billing

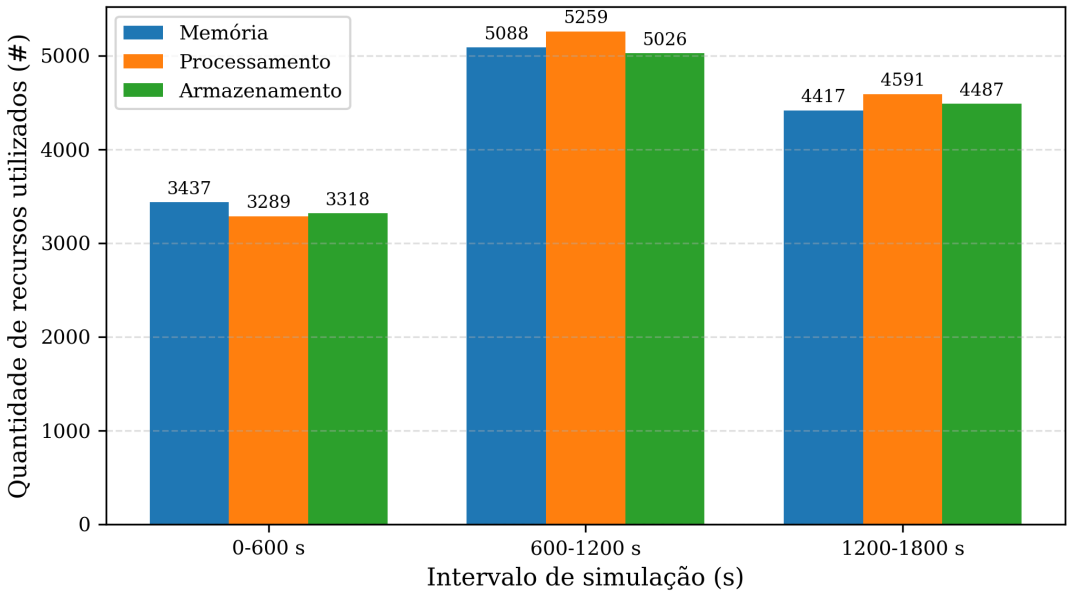
Nesta seção são apresentadas a contabilização dos recursos utilizados pelo PRIOBILL em cada cenário e a estimativa de *billing*.

Na Figura 89 é apresentada a quantidade de recursos utilizados no cenário de tarefas menores na simulação do PRIOBILL, para contabilização e aplicação do *billing*. No cenário da madrugada, até os 600 segundos de simulação, foram utilizados 3.437 MB de memória, 3.289 MB de processamento e 3.318 MB de armazenamento. No cenário da manhã, de 600 a 1200 segundos, foram utilizados 5.088 MB de memória, 5.259 MB de processamento e 5.026 MB de armazenamento. De 1200 a 1800 segundos de simulação, no horário de pico, a quantidade de recursos utilizados foi de 4.417 MB de memória, 4.591 MB de processamento e 4.487 MB de armazenamento.

Na Figura 90 apresenta-se a quantidade de recursos utilizados no cenário de tarefas intermediárias na simulação do PRIOBILL, para contabilização e aplicação do *billing*. No cenário da madrugada, até os 600 segundos de simulação, foram utilizados 2.655 MB de memória, 2.786 MB de processamento e 2.871 MB de armazenamento. No cenário da manhã, de 600 a 1200 segundos, foram utilizados 3.402 MB de memória, 3.283 MB de processamento e 3.362 MB de armazenamento. De 1200 a 1800 segundos de simulação, no horário de pico, a quantidade de recursos utilizados foi de 3.108 MB de memória, 3.119 MB de processamento e 3.095 MB de armazenamento.

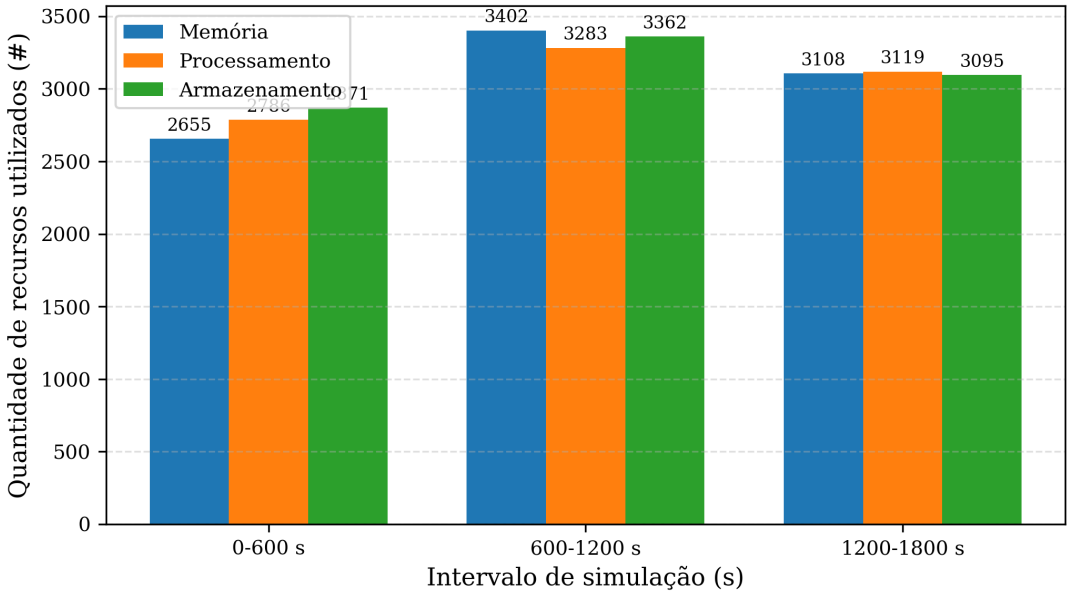
A quantidade de recursos utilizados no cenário de tarefas maiores é apresentada na Figura 91. No intervalo de até 600 segundos, o PRIOBILL acumulou a utilização de 980 MB de memória, 918 MB de processamento e 971 MB de armazenamento. De 600 a 1200 segundos, foram utilizados apenas 17 MB de memória, 18 MB de processamento e 22 MB de armazenamento. No intervalo de 1200 a 1800 segundos de simulação, foram utilizados 98 MB de memória, 97 MB de processamento e 110 MB de armazenamento.

Figura 89 – Quantidade de recursos utilizados na simulação com tarefas menores.



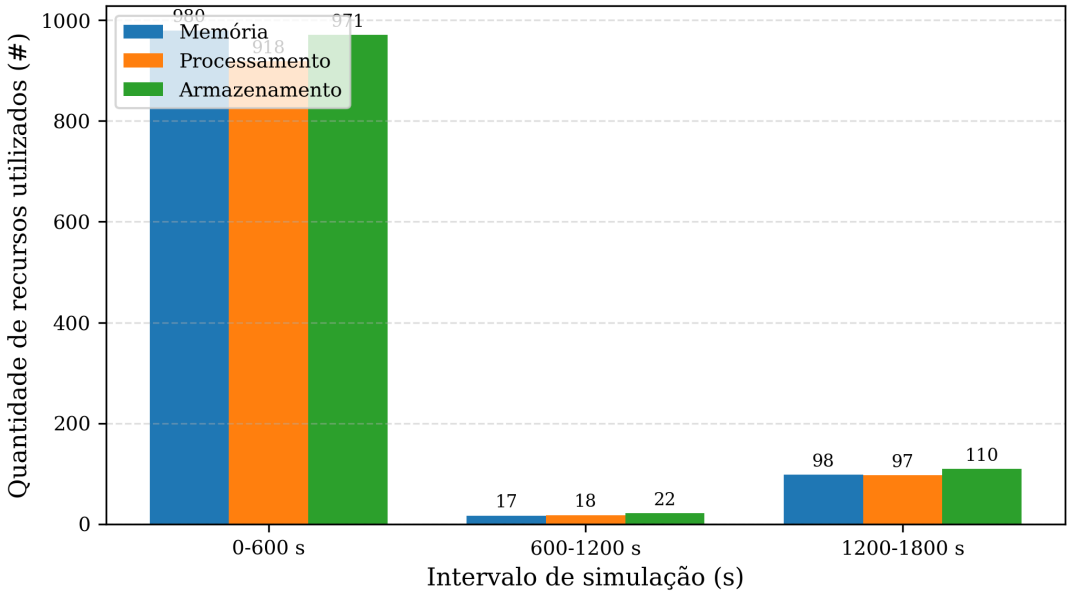
Fonte: Gerado pelo autor.

Figura 90 – Quantidade de recursos utilizados na simulação com tarefas intermediárias.



Fonte: Gerado pelo autor.

Figura 91 – Quantidade de recursos utilizados na simulação com tarefas maiores.

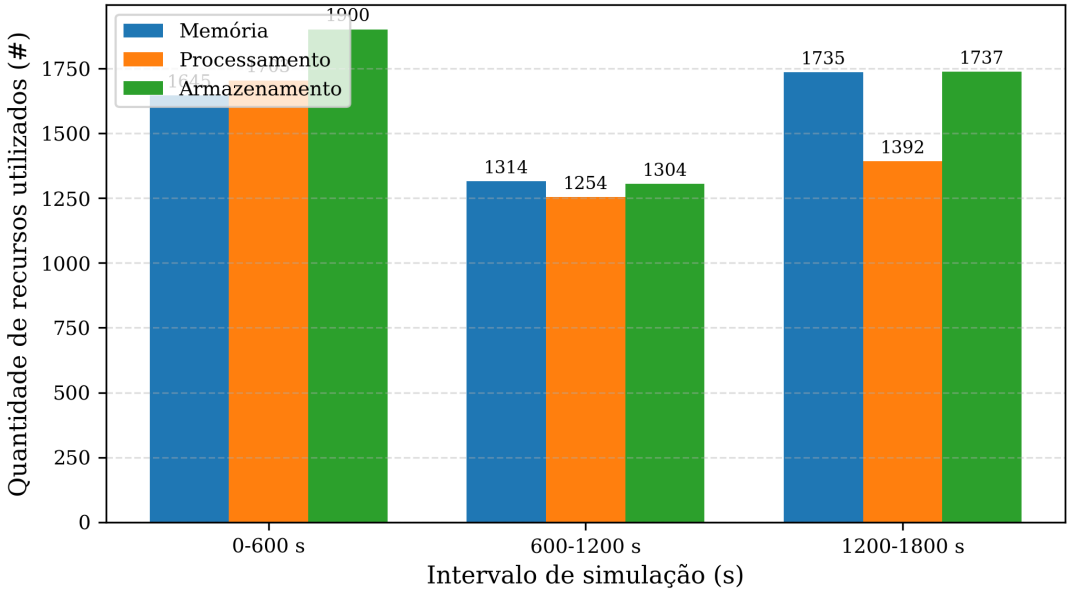


Fonte: Gerado pelo autor.

Na Figura 92 apresenta-se a quantidade de recursos utilizados no cenário de tarefas com pesos aleatórios na simulação do PRIOBILL, para contabilização e aplicação do *billing*. No cenário da madrugada, até os 600 segundos de simulação, foram utilizados 1.645 MB de memória, 1.703 MB de processamento e 1.900 MB de armazenamento. No cenário da manhã, de 600 a 1200 segundos, foram utilizados 1.314 MB de memória, 1.254 MB de processamento e 1.304 MB de armazenamento. De 1200 a 1800 segundos de simulação, no horário de pico, a quantidade de recursos utilizados foi de 1.735 MB de memória, 1.392 MB de processamento e 1.737 MB de armazenamento.

Na Tabela 6 apresentam-se os valores estimados de faturamento calculados pelo sistema de *billing* implementado no PRIOBILL. No cenário de tarefas menores, o faturamento calculado foi de R$ 0,0159 para memória, R$0,8148 para processamento e R$0,0020 para armazenamento, totalizando um faturamento de R$0,8327, com 12.942, 13.139 (6.790 tarefas) e 12.831 MBs de recursos utilizados, respectivamente. No cenário de tarefas intermediárias, o sistema calculou o *billing* de R$0,0112 para memória, R$0,2892 para processamento e R$0,0015 para armazenamento, totalizando R$0,3019 de *billing*, com 9.165, 9.188 (2.410 tarefas) e 9.328 MBs de recursos utilizados, respectivamente. Para tarefas maiores, a utilização de recursos foi de 1.095 MBs de memória, 1.033 MBs de processamento (131 tarefas) e 1.103 MBs de armazenamento, com um *billing* de R$0,0013, R$0,0157 e R$0,0002, respectivamente, e um total faturado de R$0,0172. No cenário de tarefas aleatórias, o consumo foi de 4.721 MBs de memória, 4.349 MBs de processamento (1.194 tarefas) e 4.941 MBs de armazenamento, com um *billing* calculado de R$0,0058, R$0,1433 e R$0,008, respectivamente, totalizando R$0,1570 de *billing*.

Figura 92 – Quantidade de recursos utilizados na simulação com tarefas com pesos aleatórios.



Fonte: Gerado pelo autor.

Tabela 6 – Recursos utilizados e faturamento estimado pelo sistema billing do PRIOBILL.

Métrica	**Pequenas**	**Intermediárias**	**Grandes**	**Aleatórias**
Memória utilizada (MB)	12.942	9.165	1.095	4.721
Processamento utilizado (MB)	13.139	9.188	1.033	4.349
Armazenamento utilizado (MB)	12.831	9.328	1.103	4.941
Tarefas processadas*	6.790	2.410	131	1.194
Billing da memória (R$)	0,0159	0,0112	0,0013	0,0058
Billing do processamento* (R$)	0,8148	0,2892	0,0157	0,1433
Billing do armazenamento (R$)	0,0020	0,0015	0,0002	0,0080
Billing total (R$)	**0,8327**	**0,3019**	**0,0172**	**0,1570**

***O billing do processamento é baseado na quantidade de tarefas processadas (1x CPU por tarefa processada).**

4.10.7 Considerações parciais

Com base nos resultados apresentados, observa-se que o PRIOBILL foi bastante eficiente no processamento de tarefas e no uso de recursos computacionais, conseguindo maximizar as alocações e minimizar o desperdício dos recursos disponíveis. Quanto ao número de tarefas processadas, o PRIOBILL apresentou desempenho superior aos demais métodos em diferentes cenários, alcançando vantagens de até 30,98% em relação ao segundo melhor método e mantendo desempenho praticamente equivalente ao GULOSO no maior pico de tráfego, com diferença de apenas 0,05% ao final da simulação. Em relação às tarefas não processadas, o PRIOBILL apresentou redução de até 18,98% em comparação com o segundo melhor método, mantendo desempenho consistente em diferentes níveis de

carga, tipos de tarefa e condições de tráfego. Quanto à utilização de recursos computacionais, o PRIOBILL apresentou maior capacidade de utilização em diferentes cenários, alcançando 13.707 recursos utilizados e uma diferença de 31,78% em relação aos métodos subsequentes no cenário de tarefas intermediárias durante a manhã.

Quanto à latência média, o PRIOBILL apresentou o menor tempo de seleção em todos os intervalos avaliados nos cenários de tarefas menores durante a manhã e de tarefas intermediárias no horário de pico, alcançando redução de até 0,88% em relação ao segundo melhor método. Além disso, apresentou um bom desempenho na avaliação da vazão *throughput*. Na análise do throughput, o PRIOBILL apresentou o melhor desempenho em diferentes cenários, destacando-se nas tarefas intermediárias durante a manhã, com vantagem de até 30,98% em relação ao segundo melhor mecanismo, e nas tarefas maiores, com vantagem de 43,21% em relação ao segundo colocado.

Ao buscar um equilíbrio entre as métricas de vazão, latência e atendimento, o PRIOBILL classificou-se entre os melhores algoritmos testados. A dinâmica de coleta de dados aplicada na proposta do PRIOBILL também permitiu analisar separadamente os custos dos recursos considerados nas simulações e, assim, realizar a contabilização dos custos do mecanismo. O custo operacional foi estimado utilizando um modelo linear de precificação em nuvem, baseado nas dimensões de faturamento do AWS Fargate. O custo total foi calculado considerando os recursos de processamento, memória e armazenamento faturados separadamente. Esses valores foram utilizados para estimar o custo operacional associado aos recursos computacionais alocados durante as simulações.

5 CONCLUSÃO

Os Sistemas de Transportes Inteligentes tornam-se uma realidade cada vez mais presente no cotidiano das pessoas, possibilitada pela evolução expressiva do mundo tecnológico. As redes veiculares em cidades inteligentes contribuem significativamente para essa evolução tecnológica. No entanto, existem desafios importantes a serem investigados e propostas de solução. Entre os diversos desafios, destaca-se a demanda por um gerenciamento da escassez de recursos disponibilizados pelos usuários em Computação de Borda Veicular.

Este trabalho foi direcionado à descoberta de soluções eficientes para gerenciar o processo de alocação de tarefas computacionais em Computação de Borda Veicular em cenários urbanos. Nesse sentido, foi apresentado um *framework* para gerenciamento de alocação de tarefas em Computação de Borda Veicular com prioridade por colaboração e sistema *billing*. O *framework*, chamado PRIOBILL, implementa, na comunicação veículo-para-infraestrutura, um sistema de gerenciamento de tarefas baseado na colaboração dos usuários, considerando a disponibilização de recursos e o processamento de tarefas. O *framework* também possui um método de seleção de nó de borda baseado no algoritmo meta-heurístico bioinspirado na natureza, denominado algoritmo de otimização do leão. Além disso, um sistema *billing* foi implementado no *framework* com base nas tarefas atendidas pelos usuários.

Uma simulação em um cenário urbano realista, com variações de tarefas e tráfego de veículos, foi utilizada para validar a eficiência do *framework* PRIOBILL. O mecanismo proposto mostrou-se eficiente ao maximizar o atendimento de tarefas na maioria dos cenários implementados, principalmente nos cenários com maior escassez de recursos, alcançando até 30,98% no número de tarefas processadas em relação ao segundo melhor método. Também buscou minimizar as tarefas não processadas, reduzindo em até 18,98% o número de tarefas não processadas em relação ao segundo melhor método. O PRIOBILL também demonstrou eficiência na utilização de recursos, alcançando uma diferença de até 31,78% em relação aos métodos subsequentes, e apresentou um bom *throughput*, com vantagens de até 30,98% sobre o segundo melhor mecanismo. Além disso, manteve uma média de latência adequada em comparação com a variação apresentada pelos demais algoritmos da literatura. O *framework* também implementou, de modo dinâmico, um sistema *billing*, contabilizando a colaboração dos usuários de forma a permitir a compensação como incentivo para o compartilhamento de recursos.

Além disso, o *framework* PRIOBILL demonstrou ser facilmente adaptável a diferentes cenários de simulação, apresentando pouca variação em seu desempenho, ao

contrário dos mecanismos da literatura, que oscilaram significativamente dependendo do cenário. Os métodos baseados em algoritmos meta-heurísticos permitem essa adaptação simplificada. Nesse sentido, podem ser utilizados em diversos cenários, considerando diferentes métricas e podendo ser aplicados em tomadas de decisão de outras áreas.

5.1 PRINCIPAIS CONTRIBUIÇÕES

Com esta pesquisa foi possível contribuir com a literatura. Dessa forma, nesta seção são listadas as contribuições do *framework* PRIOBILL:

- Proposta de um *framework* dinâmico para alocação de tarefas em Computação de Borda Veicular utilizando comunicação veículo-para-infraestrutura;
- Desenvolvimento do *framework* PRIOBILL na linguagem de programação Python, com o objetivo de ser facilmente customizável;
- Implantação de um simulador de mobilidade urbana para criação de um ambiente realista de simulação com serviços de computação de borda móvel;
- Criação de tarefas com tamanhos diferentes para avaliar a capacidade de adaptação do PRIOBILL;
- Implementação de um mecanismo dinâmico de colaboração como forma de incentivo ao compartilhamento de recursos nos serviços de computação de borda veicular;
- Maximização do atendimento de tarefas nos mais diferentes cenários de mobilidade e tamanhos de tarefas;
- Maximização do aproveitamento dos recursos disponibilizados nos serviços de computação de borda veicular;
- Maximização do *throughput* e equilíbrio da latência média aplicada no processo de seleção do nó de borda do PRIOBILL;
- Implementação de um sistema *billing* como forma de compensação e incentivo aos usuários colaboradores.

5.2 PRODUÇÃO CIENTÍFICA

Nesta seção são apresentadas as produções científicas publicadas durante o período do doutorado. Destacam-se os trabalhos publicados em periódicos e congressos como primeiro autor. Também são apresentadas as colaborações em trabalhos relacionados a esta pesquisa e publicados em periódicos e congressos. Os trabalhos aqui apresentados estão relacionados às pesquisas desenvolvidas neste projeto.

5.2.1 Publicação em periódico como autor principal

Lieira et al. (2023b). An Evaluation of Bio-Inspired Resource Allocation Methods for Vehicular Edge Computing. IEEE COMMUNICATIONS MAGAZINE, v. 1, p. 1-7, 2023.

5.2.2 Publicações em congressos como autor principal

- Lieira et al. (2025). LOPRIVE: LOA-Based Priority-Driven Task Allocation in the Vehicular Edge. In: 2025 IEEE Symposium on Computers and Communications (ISCC), 2025, Bologna. 2025 IEEE Symposium on Computers and Communications (ISCC), 2025. p. 1.
- Lieira et al. (2025). MPAT: Mecanismo para Priorização de Alocação de Tarefas em VEC por Função de Tarefas Inspirado em Algoritmo Bioinspirado. In: CISTI'2025 - 20th Iberian Conference on Information Systems and Technologies, 2025, Lisboa. Iberian Conference on Information Systems and Technologies, 2025.
- Lieira et al. (2023a). Optimization of the Task Allocation Process in VEC with the GWO Bioinspired Algorithm. In: 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), 2023, Aveiro. 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), 2023. p. 1.
- Lieira et al. (2022b). 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), 2022, Marina del Rey. 2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2022. p. 283.
- Lieira et al. (2022a). 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), 2022, Madrid. 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), 2022. p. 1.

5.2.3 Colaborações em publicações em periódicos

- Quessada et al. (2026). Adaptive Bat-Inspired Optimization for Infrastructure-Agnostic Task Allocation in Vehicular Fogs. Ieee Open Journal Of Intelligent Transportation Systems, v. 1, p. 1-1, 2026.
- Ehsan et al. (2026). Adaptive Priority-Based Edge-Centric Resource Management for the Internet of Vehicles. Future Generation Computer Systems, v. 175, p. 108094, 2025.

5.2.4 Colaborações em publicações em congressos

- Gottsfritz et al. (2023). MALTREAT - A Metaheuristic Algorithm for Allocation of Resources in Vehicular AD-HOC Networks. In: 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSSIoT), 2023, Pafos. 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), 2023. p. 609.
- 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), 2022, Marina del Rey. 2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2022. p. 298.
- Quessada et al. (2022a). ARCANE: Algoritmo Meta-heurístico para Alocação de Tarefas em Nuvens Veiculares. In: Workshop de Computação Urbana, 2022, Brasil. Anais do VI Workshop de Computação Urbana (CoUrb 2022). p. 140.

5.3 TRABALHOS FUTUROS

Com o desenvolvimento do *framework* proposto neste trabalho, são apresentadas novas ideias para avançar as pesquisas relacionadas à alocação de tarefas computacionais em computação de borda veicular, as quais serão listadas a seguir:

- Utilizar aprendizado de máquina com algoritmos meta-heurísticos para buscar otimizar a eficiência da alocação de tarefas e da utilização de recursos computacionais;
- Propor novas formas de compensação aos usuários durante os processos de alocação para incentivar a colaboração com a computação de borda veicular;
- Implantar no *framework* análises e soluções para lidar com outros fatores, como tempo de resposta, interferências, oscilações de rede, entre outros;
- Implementar no sistema *billing* os módulos de segurança das transações, integrações de pagamentos, autenticação, auditorias e políticas econômicas;
- Desenvolver outras topologias de rede, considerar outras infraestruturas, outros dispositivos e outras métricas da literatura para analisar a eficiência do mecanismo desenvolvido;

5.4 UTILIZAÇÃO DE INTELIGÊNCIA ARTIFICIAL

Neste trabalho foi utilizada a Inteligência Artificial, por meio do programa ChatGPT, para a geração e adaptação de imagens com maior qualidade, contribuindo para o melhor

entendimento dos cenários propostos neste projeto. Também foi utilizada a ferramenta para a correção ortográfica e gramatical do texto deste trabalho.

- Imagens: https://chatgpt.com/c/6a198675-00b8-83e9-8a1c-d5967a715ace
- Correção do texto: https://chatgpt.com/share/6a4e853a-aae4-83e9-9cb7-ea18bb1e652f

REFERÊNCIAS

ALAM, M.; FERREIRA, J.; FONSECA, J. (Ed.). *Introduction to Intelligent Transportation Systems*. Cham: Springer International Publishing, 2016. 1–17 p. ISBN 978-3-319-28183-4. Disponível em: https://doi.org/10.1007/978-3-319-28183-4_1.

ALFARRAJ, O. et al. Neighbor predictive adaptive handoff algorithm for improving mobility management in vanets. *Computer Networks*, v. 151, p. 224–231, 2019. ISSN 1389-1286.

ALI, I.; LI, F. An efficient conditional privacy-preserving authentication scheme for vehicle-to-infrastructure communication in vanets. *Vehicular Communications*, Elsevier, v. 22, p. 100228, 2020.

ANDRIULO, F. C. et al. Edge computing and cloud computing for internet of things: A review. *Informatics*, v. 11, n. 4, 2024. ISSN 2227-9709. Disponível em: https://www.mdpi.com/2227-9709/11/4/71.

AWS. Aws fargate pricing. In: *AWS Amazon*. AWS European Sovereign Cloud, Accessado: 16/05/2026. p. 1. Disponível em: https://aws.eu/fargate/pricing/.

AZIZI, S. et al. Deadline-aware and energy-efficient iot task scheduling in fog computing systems: A semi-greedy approach. *Journal of Network and Computer Applications*, v. 201, p. 103333, 2022. ISSN 1084-8045.

BELLAVISTA, P. et al. A survey on fog computing for the internet of things. *Pervasive and Mobile Computing*, v. 52, p. 71–99, 2019. ISSN 1574-1192. Disponível em: https://www.sciencedirect.com/science/article/pii/S1574119218301111.

BITTENCOURT, J. C. N. et al. The road to intelligent cities. *Smart Cities*, MDPI, v. 8, n. 3, p. 77, 2025.

BOUKERCHE, A.; SOTO, V. Computation offloading and retrieval for vehicular edge computing: Algorithms, models, and classification. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 53, n. 4, aug 2020. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3392064.

BURHAN, M. et al. A comprehensive survey on the cooperation of fog computing paradigm-based iot applications: Layered architecture, real-time security issues, and solutions. *IEEE Access*, v. 11, p. 73303–73329, 2023.

CHEN, Q. et al. Digital twin assisted multi-task offloading for vehicular edge computing under sagin with blockchain. *IET Communications*, v. 19, n. 1, p. e70002, 2025. Disponível em: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cmu2.70002.

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

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

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.

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

CUI, J. et al. Reliable and efficient content sharing for 5g-enabled vehicular networks. *IEEE Transactions on ITS*, v. 23, n. 2, p. 1247–1259, 2022.

DAI, P. et al. A probabilistic approach for cooperative computation offloading in mec-assisted vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, v. 23, n. 2, p. 899–911, 2022.

EHSAN, M. et al. Adaptive priority-based edge-centric resource management for the internet of vehicles. *Future Generation Computer Systems*, v. 175, p. 108094, 2026. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X25003887.

EICHLER, S. Performance evaluation of the ieee 802.11 p wave communication standard. In: IEEE. *2007 IEEE 66th Vehicular Technology Conference*. [S.l.], 2007. p. 2199–2203.

EKE, C. I. et al. A survey of user profiling: State-of-the-art, challenges, and solutions. *IEEE Access*, v. 7, p. 144907–144924, 2019.

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

FU, S.; DING, C.; JIANG, P. Computational offloading of service workflow in mobile edge computing. *Information*, v. 13, n. 7, 2022. ISSN 2078-2489. Disponível em: https://www.mdpi.com/2078-2489/13/7/348.

GARG, T.; KAUR, G. A systematic review on intelligent transport systems. *Journal of Computational and Cognitive Engineering*, v. 2, n. 3, p. 175–188, Aug. 2023. Disponível em: https://ojs.bonviewpress.com/index.php/JCCE/article/view/245.

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

GONG, Y. et al. Resource allocation for integrated sensing and communication in digital twin enabled internet of vehicles. *IEEE Transactions on Vehicular Technology*, v. 72, n. 4, p. 4510–4524, 2023.

GOTTSFRITZ, E. N. et al. Maltreat - a metaheuristic algorithm for allocation of resources in vehicular ad-hoc networks. In: *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. [S.l.: s.n.], 2023. p. 609–616.

GOYAL, S. et al. An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm. *Sensors*, v. 21, n. 5, 2021. ISSN 1424-8220.

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.

HOSSEINI, E.; NICKRAY, M.; GHANBARI, S. Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process. *Computer Networks*, v. 206, p. 108752, 2022. ISSN 1389-1286.

HOUSSEIN, E. H. et al. Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends. *Swarm and Evolutionary Computation*, v. 62, p. 100841, 2021. ISSN 2210-6502. Disponível em: https://www.sciencedirect.com/science/article/pii/S221065022100002X.

HUSSEIN, N. H. et al. A comprehensive survey on vehicular networking: Communications, applications, challenges, and upcoming research directions. *IEEE Access*, v. 10, p. 86127–86180, 2022.

INSIGHTS, F. B. Vehicle to vehicle communication market size, share industry analysis, by vehicle type (hatchback/sedan, suvs, lcv, and hcv), by technology (dedicated short-range communication (dsrc) and cellular communication), by application (traffic safety, traffic efficiency, infotainment and payments, and other applications), by sales channel (oem and aftermarket), and regional forecast, 2025-2032. In: *Transportation Logistics/Vehicle to Vehicle Communication Market*. Fortune Business Insights, 2024. p. 200. Disponível em: https://www.fortunebusinessinsights.com/vehicle-to-vehicle-communication-market-112867.

JIANG, D.; DELGROSSI, L. Ieee 802.11p: Towards an international standard for wireless access in vehicular environments. In: *VTC Spring 2008 - IEEE Vehicular Technology Conference*. [S.l.: s.n.], 2008. p. 2036–2040.

JU, Y. et al. Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, v. 24, n. 5, p. 5555–5569, 2023.

KESHARI, N.; SINGH, D.; MAURYA, A. K. A survey on vehicular fog computing: Current state-of-the-art and future directions. *Vehicular Communications*, v. 38, p. 100512, 2022. ISSN 2214-2096. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214209622000596.

KHAN, W. Z. et al. Edge computing: A survey. *Future Generation Computer Systems*, v. 97, p. 219–235, 2019. ISSN 0167-739X. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167739X18319903.

KOMNINOS, N. *The age of intelligent cities: smart environments and innovation-for-all strategies*. [S.l.]: Routledge, 2014.

KUMAR, A. S.; ZHAO, L.; FERNANDO, X. Task offloading and resource allocation in vehicular networks: A lyapunov-based deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, v. 72, n. 10, p. 13360–13373, 2023.

KUMAR, S. et al. A game-theoretic approach for increasing resource utilization in edge computing enabled internet of things. *IEEE Access*, v. 10, p. 57974–57989, 2022.

LIEIRA, D. et al. Mpat: Mecanismo para priorização de alocação de tarefas em vec por função de tarefas inspirado em algoritmo bioinspirado. In: *CISTI'2025 - 20th Iberian Conference on Information Systems and Technologies*. [S.l.]: CISTI, 2025.

LIEIRA, D. D. et al. Meta-heuristic mechanism based on whale optimization algorithm for tasks allocation in edge computing. In: *Proc. of the Iberian Conference on Information Systems and Technologies*. [S.l.: s.n.], 2022. p. 1–6.

LIEIRA, D. D. et al. Mechanism for optimizing resource allocation in vanets based on the pso bio-inspired algorithm. In: *Proc. of the Int. Conference on Distributed Computing in Sensor Systems*. [S.l.: s.n.], 2022. p. 283–290.

LIEIRA, D. D. et al. LOPRIVE: LOA-Based Priority-Driven Task Allocation in the Vehicular Edge . In: *2025 IEEE Symposium on Computers and Communications (ISCC)*. Los Alamitos, CA, USA: IEEE Computer Society, 2025. p. 1–6. Disponível em: https://doi.ieeecomputersociety.org/10.1109/ISCC65549.2025.11325929.

LIEIRA, D. D. et al. Optimization of the task allocation process in vec with the gwo bioinspired algorithm. In: *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.: s.n.], 2023. p. 1–7.

LIEIRA, D. D. et al. An evaluation of bio-inspired resource allocation methods for vehicular edge computing. *IEEE Communications Magazine*, p. 1–7, 2023.

LIEIRA, D. D. et al. An evaluation of bio-inspired resource allocation methods for vehicular edge computing. *IEEE Communications Magazine*, v. 62, n. 5, p. 120–126, 2024.

LIU, B.; LIU, C.; PENG, M. Computation offloading and resource allocation in unmanned aerial vehicle networks. *IEEE Transactions on Vehicular Technology*, v. 72, n. 4, p. 4981–4995, 2023.

LIU, L. et al. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, v. 26, n. 3, jun 2021. ISSN 1572-8153. Disponível em: https://doi.org/10.1007/s11036-020-01624-1.

MENEGUETTE, R. et al. Vehicular edge computing: Architecture, resource management, security, and challenges. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 55, n. 1, nov 2021. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/3485129.

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

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

NGUYEN, H. D. et al. A run-time dynamic computation offloading strategy in vehicular edge computing. In: *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*. [S.l.: s.n.], 2021. p. 1–7.

NING, Z. et al. Deep reinforcement learning for vehicular edge computing: An intelligent offloading system. *ACM Trans. Intell. Syst. Technol.*, Association for Computing

Machinery, New York, NY, USA, v. 10, n. 6, oct 2019. ISSN 2157-6904. Disponível em: https://doi.org/10.1145/3317572.

NOOR-A-RAHIM, M. et al. A survey on resource allocation in vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, v. 23, n. 2, p. 701–721, 2022.

OWOADE, S. et al. Cost-reduction models in cloud-native applications: Strategies for billing optimization in aws environments. *Journal of Frontiers in Multidisciplinary Research (JFMR),*, v. 4, n. 1, 2023.

PATTUROSE, J. G. B. et al. Remote and centralized billing system using cloud computing. In: OWOC, M. L. et al. (Ed.). *Computational Intelligence in Data Science*. Cham: Springer Nature Switzerland, 2024. p. 241–249. ISBN 978-3-031-69986-3.

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

QUESSADA, M. et al. Arcane: Algoritmo meta-heurístico para alocação de tarefas em nuvens veiculares. In: *Anais do VI Workshop de Computação Urbana*. Porto Alegre, RS, Brasil: SBC, 2022. p. 140–153. ISSN 2595-2706. Disponível em: https://sol.sbc.org.br/index.php/courb/article/view/21450.

QUESSADA, M. S. et al. Towards bat bio-inspired decision-making for task allocation in vehicular fogs. In: *Proc. of the International Conference on DCOSS*. [S.l.: s.n.], 2022. p. 298–305.

QUESSADA, M. S. et al. Adaptive bat-inspired optimization for infrastructure-agnostic task allocation in vehicular fogs. *IEEE Open Journal of Intelligent Transportation Systems*, v. 7, p. 923–936, 2026.

RIBEIRO, A. et al. Harmonic: Shapley values in market games for resource allocation in vehicular clouds. *Ad Hoc Networks*, v. 149, p. 103224, 2023. ISSN 1570-8705. Disponível em: https://www.sciencedirect.com/science/article/pii/S1570870523001440.

SANDESH, R.; SANTHOSH, K. Revolutionizing vehicle communication: A comprehensive exploration of technologies for enhanced road safety and autonomous vehicles. *Alexandria Engineering Journal*, v. 129, p. 976–997, 2025. ISSN 1110-0168. Disponível em: https://www.sciencedirect.com/science/article/pii/S1110016825008804.

SHA-MOHAMMAD, S. A.; ABDEL-WAHAB, H. M.; POPESCU, D. C. A study of ieee 802.11 standard for use in vehicle to infrastructure communication. In: *2013 IEEE Globecom Workshops (GC Wkshps)*. [S.l.: s.n.], 2013. p. 1303–1307.

SHAO, S. et al. Multi task dynamic edge–end computing collaboration for urban internet of vehicles. *Computer Networks*, v. 227, p. 109690, 2023. ISSN 1389-1286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1389128623001354.

SHOELEH, F. et al. User profiling on universal data insights tool on ibm cloud pak for security. In: *2021 18th International Conference on Privacy, Security and Trust (PST)*. [S.l.: s.n.], 2021. p. 1–10.

SING, R. et al. A whale optimization algorithm based resource allocation scheme for cloud-fog based iot applications. In: *Electronics 2022*. [S.l.: s.n.], 2022. v. 11, p. 3207.

STANDER, P. E. Cooperative hunting in lions: the role of the individual. *Behavioral ecology and sociobiology*, Springer, v. 29, n. 6, p. 445–454, 1992.

STRIPE. Infrastructure-as-a-service (iaas) pricing explained. In: . Stripe Billing, 2025. Disponível em: https://stripe.com/en-br/resources/more/infrastructure-as-a-service-iaas-pricing-explained.

SUNYAEV, A. Cloud computing. In: _____. *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*. Cham: Springer Nature Switzerland, 2024. p. 165–209. ISBN 978-3-031-61014-1. Disponível em: https://doi.org/10.1007/978-3-031-61014-1_6.

TIZHOOSH, H. R. Opposition-based learning: A new scheme for machine intelligence. In: *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06) - Volume 01*. USA: IEEE Computer Society, 2005. (CIMCA '05), p. 695–701. ISBN 076952504001.

TOLOSANA-CALASANZ, R. et al. Capacity management for streaming applications over cloud infrastructures with micro billing models. In: *Proceedings of the 9th International Conference on Utility and Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2016. (UCC '16), p. 251–256. ISBN 9781450346160. Disponível em: https://doi.org/10.1145/2996890.3007868.

TOMAR, V.; BANSAL, M.; SINGH, P. Metaheuristic algorithms for optimization: A brief review. *Engineering Proceedings*, v. 59, n. 1, 2023. ISSN 2673-4591. Disponível em: https://www.mdpi.com/2673-4591/59/1/238.

WAHEED, A. et al. A comprehensive review of computing paradigms, enabling computation offloading and task execution in vehicular networks. *IEEE Access*, v. 10, p. 3580–3600, 2022.

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

WANG, J. et al. Distributed clustering-based cooperative vehicular edge computing for real-time offloading requests. *IEEE Transactions on Vehicular Technology*, v. 71, n. 1, p. 653–669, 2022.

WEI, W. et al. Multi-objective optimization for resource allocation in vehicular cloud computing networks. *IEEE Transactions on Intelligent Transportation Systems*, v. 23, n. 12, p. 25536–25545, 2022.

XIANG, Z. et al. Robust and cost-effective resource allocation for complex iot applications in edge-cloud collaboration. *Mobile Networks and Applications*, v. 27, n. 4, p. 1506–1519, 2022.

XIAO, Y.; ZHU, C. Vehicular fog computing: Vision and challenges. In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. [S.l.: s.n.], 2017. p. 6–9.

XIE, Y. et al. A random online algorithm for reselling reserved iaas instances in amazon's cloud marketplace. *IEEE Transactions on Network Science and Engineering*, v. 9, n. 3, p. 1235–1244, 2022.

YAZDANI, M.; JOLAI, F. Lion optimization algorithm (loa): A nature-inspired metaheuristic algorithm. *Journal of Computational Design and Engineering*, v. 3, n. 1, p. 24–36, 2016. ISSN 2288-4300. Disponível em: https://www.sciencedirect.com/science/article/pii/S2288430015000524.

YEDDULAPALLI, H. S. et al. *VECA: Reliable and Confidential Resource Clustering for Volunteer Edge-Cloud Computing*. 2024. Disponível em: https://arxiv.org/abs/2409.03057.

DADOS CURRICULARES

IDENTIFICAÇÃO	
	Douglas Dias Lieira Data nasc. 23/03/1987
Nacionalidade	Brasileiro
Nome em citações bibliográficas	Lieira, Douglas D.
Currículo Lattes	https://lattes.cnpq.br/3154645144661505
ORCID	https://orcid.org/0000-0001-9622-1913
FORMAÇÃO ACADÊMICA	
2022/2026	Ciência da Computação - Doutorado Universidade Estadual Paulista - UNESP
2020/2021	Ciência da Computação - Mestrado Universidade Estadual Paulista - UNESP
2008/2011	Informática para Negócios - Graduação Faculdade de Tecnologia de SJRP - FATEC
PRODUÇÃO BIBLIOGRÁFICA	
QUESSADA, MATHEUS S.; LIEIRA, DOUGLAS D.; YAN, PENGHAN; GRANDE, ROBSON E. DE; MENEGUETTE, RODOLFO I. Adaptive Bat-Inspired Optimization for Infrastructure-Agnostic Task Allocation in Vehicular Fogs. *In: IEEE Open Journal of Intelligent Transportation Systems.* v. 1, p. 1-1, 2026.	
EHSAN, MOHAIMIN ; LIEIRA, DOUGLAS D. ; MENEGUETTE, RODOLFO I. ; DE GRANDE, ROBSON E. Adaptive Priority-Based Edge-Centric Resource Management for the Internet of Vehicles. Future Generation Computer Systems. v. 175, p. 108094, 2025.	
LIEIRA, DOUGLAS D.; QUESSADA, MATHEUS S. ; SAMPAIO, SANDRA ; LOUREIRO, ANTONIO A. F. ; MENEGUETTE, RODOLFO I. An Evaluation of Bio-Inspired Resource Allocation Methods for Vehicular Edge Computing. IEEE COMMUNICATIONS MAGAZINE, v. 1, p. 1-7, 2023.	

LIEIRA, D. D.; QUESSADA, M. S. ; CRISTIANI, A. L. ; MENEGUETTE, R. I. Algorithm for 5G Resource Management Optimization in Edge Computing. IEEE Latin America Transactions, v. 19, p. 1772-1780, 2021.

PEREIRA, RICKSON S. ; LIEIRA, DOUGLAS D. ; SILVA, M. A. C. ; PIMENTA, ADINOVAM H. M. ; DA COSTA, JOAHANNES B. D. ; ROSÁRIO, DENIS ; VILLAS, LEANDRO ; MENEGUETTE, RODOLFO I. . RELIABLE: Resource Allocation Mechanism for 5G Network using Mobile Edge Computing. SENSORS, v. 20, p. 5449, 2020.

QUESSADA, MATHEUS S ; PEREIRA, RICKSON S ; REVEJES, WILLIAM ; SARTORI, BRUNO ; GOTTSFRITZ, EUCLYDES N ; LIEIRA, DOUGLAS D ; DA SILVA, MARCO AC ; ROCHA FILHO, GERALDO P ; MENEGUETTE, RODOLFO I . ITSMEI: An intelligent transport system for monitoring traffic and event information. International Journal of Distributed Sensor Networks, v. 16, p. 155014772096375, 2020.

LIEIRA, DOUGLAS D.; QUESSADA, MATHEUS S. ; GRANDE, ROBSON E. DE ; MENEGUETTE, RODOLFO I. . LOPRIVE: LOA-Based Priority-Driven Task Allocation in the Vehicular Edge. *In:* 2025 IEEE Symposium on Computers and Communications (ISCC), 2025, Bologna. **Anais** 2025 IEEE Symposium on Computers and Communications (ISCC), 2025. p. 1.

GOTTSFRITZ, EUCLYDES N. ; QUESSADA, MATHEUS S. ; LIEIRA, DOUGLAS D. ; ROCHA FILHO, GERALDO P. ; MENEGUETTE, RODOLFO I. . MARVEL: Um Algoritmo Meta-heurístico para Alocação de Recursos em Redes Veiculares ad-hoc. *In:* Workshop de Gerência e Operação de Redes e Serviços, 2023, Brasil. **Anais** do XXVIII Workshop de Gerência e Operação de Redes e Serviços (WGRS 2023). p. 15.

LIEIRA, DOUGLAS DIAS; QUESSADA, MATHEUS SANCHES ; NAKAMURA, LUIS HIDEO VASCONCELOS ; SAMPAIO, SANDRA ; DE GRANDE, ROBSON E. ; MENEGUETTE, RODOLFO IPOLITO . Optimization of the Task Allocation Process in VEC with the GWO Bioinspired Algorithm. *In:* 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), 2023, Aveiro. **Anais** 2023 18th Iberian Conference on Information Systems and Technologies (CISTI), 2023. p. 1.

GOTTSFRITZ, EUCLYDES N. ; QUESSADA, MATHEUS S. ; LIEIRA, DOUGLAS D. ; GUIDONI, DANIEL L. ; FILHO, GERALDO P. ROCHA ; MENEGUETTE, RODOLFO I. . MALTREAT - A Metaheuristic Algorithm for Allocation of Resources in Vehicular AD-HOC Networks. *In:* 2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSSIoT), 2023, Pafos. **Anais** 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT), 2023. p. 609.

LIEIRA, D. D.; QUESSADA, M. S. ; DE GRANDE, R. ; MENEGUETTE, R. I. . MPAT: Mecanismo para Priorização de Alocação de Tarefas em VEC por Função de Tarefas Inspirado em Algoritmo Bioinspirado. *In:* CISTI'2025 - 20th Iberian Conference on Information Systems and Technologies, 2025, Lisboa. **Anais** Iberian Conference on Information Systems and Technologies, 2025.

QUESSADA, MATHEUS S. ; LIEIRA, DOUGLAS D. ; DE GRANDE, ROBSON E. ; MENEGUETTE, RODOLFO I. . Towards Bat Bio-inspired Decision-making for Task Allocation in Vehicular Fogs. *In:* 2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2022, Marina del Rey. **Anais** 18th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2022. p. 298.

QUESSADA, MATHEUS S. ; LIEIRA, DOUGLAS D. ; COSTA, JOAHANNES B. D. DA ; ROCHA FILHO, GERALDO P. ; GRANDE, ROBSON E. DE ; MENEGUETTE, RODOLFO I. . ARCANE: Algoritmo Meta-heurístico para Alocação de Tarefas em Nuvens Veiculares. *In:* Workshop de Computação Urbana, 2022, Brasil. **Anais** 18th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2022. p. 283.

LIEIRA, DOUGLAS D.; QUESSADA, MATHEUS S. ; CRISTIANI, ANDRE L. ; DE GRANDE, ROBSON E. ; MENEGUETTE, RODOLFO I. . 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), 2022, Marina del Rey. **Anais** Iberian Conference on Information Systems and Technologies, 2025.

LIEIRA, DOUGLAS DIAS; GOTTSFRITZ, EUCLYDES NASORRI ; QUESSADA, MATHEUS SANCHES ; CRISTIANI, ANDRE LUIS ; FILHO, GERALDO P. ROCHA ; MENEGUETTE, RODOLFO IPOLITO . 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), 2022, Madrid. **Anais** 17th Iberian Conference on Information Systems and Technologies (CISTI), 2022. p. 1.

LIEIRA, DOUGLAS DIAS; QUESSADA, MATHEUS SANCHES ; CRISTIANI, ANDRE LUIS ; IMMICH, ROGER ; MENEGUETTE, RODOLFO IPOLITO . TRIAD: Whale Optimization Algorithm for 5G-IoT Resource Allocation Decision in Edge Computing. *In:* 2021 16th Iberian Conference on Information Systems and Technologies (CISTI), 2021, Chaves. **Anais** 16th Iberian Conference on Information Systems and Technologies (CISTI), 2021. p. 1.

LIEIRA, DOUGLAS D.; QUESSADA, MATHEUS S. ; DA COSTA, JOAHANNES B. D. ; CERQUEIRA, EDUARDO ; ROSARIO, DENIS ; MENEGUETTE, RODOLFO I. . TOVEC: Task Optimization Mechanism for Vehicular Clouds using Metaheuristic Technique. *In:* International Wireless Communications and Mobile Computing (IWCMC), 2021, Harbin City. **Anais** International Wireless Communications and Mobile Computing (IWCMC), 2021. p. 358.

LIEIRA, DOUGLAS D.; QUESSADA, MATHEUS S. ; CRISTIANI, ANDRE L. ; MENEGUETTE, RODOLFO I. . Resource Allocation Technique for Edge Computing Using Grey Wolf Optimization Algorithm. *In:* 2020 IEEE LatinAmerican Conference on Communications (LATINCOM), 2020, Santo Domingo. **Anais** IEEE LatinAmerican Conference on Communications (LATINCOM), 2020. p. 1.

LIEIRA, D. D.; MENEGUETTE, R. I. ; VALENTINI, E. P. ; NAKAMURA, L. H. V. . Uma Nova Arquitetura Descentralizada para Mobilidade de Fluxo em Redes Veiculares. *In:* XXIII Workshop de Gerência e Operação de Redes e Serviços, 2018, Campos do Jordão. **Anais** XXIII Workshop de Gerência e Operação de Redes e Serviços, 2018.

PARTICIPAÇÃO EM BANCAS E ORIENTAÇÕES

LIEIRA, D. D.; VALENTINI, E. P.; TEIXEIRA, M. A.. Participação em banca de Classio Gabriel da Silva. **Simulador de Bomba de Infusão.** Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) - Instituto Federal de São Paulo. Catanduva. 2022.

LIEIRA, D. D.; MASCHI, L. F. C.; DA SILVA, MARCO AC. Participação em banca de Matheus Sanches Quessada.**Sistema de Gestão Esportiva Aplicado ao Futebol.** Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) - Instituto Federal de São Paulo. Catanduva. 2019.

LIEIRA, D. D.; MASCHI, L. F. C.; MURARI., R. C. B. D. L.. Participação em banca de Rickson Pereira Simioni. **Game Learning: Um Jogo Educacional Para Crianças. Desenvolvimento de Um Jogo 2D Para Dispositivos Móveis.** Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) - Instituto Federal de São Paulo. Catanduva. 2018.

LIEIRA, D. D.; MASCHI, L. F. C.; MURARI., R. C. B. D. L.. **Participação em banca de Danielle da Silva Fernandes.Game Learning: Um Jogo Educacional Para Crianças. Desenvolvimento de Um Jogo 2D Para Dispositivos Móveis.** 2018. Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) - Instituto Federal de São Paulo. Catanduva. 2018.

LIEIRA, D. D.; MASCHI, L. F. C.; MENEGUETTE, R. I.. Participação em banca de André Luis Cristiani. **Aplicando Mineração de Textos Para Análise de Tweets Referentes Às Eleições Presidenciais de 2018.** 2018. Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) - Instituto Federal de São Paulo. Catanduva. 2018.

PARTICIPAÇÃO EM EVENTOS CIENTÍFICOS

CISTI, 20th., 2025, (Lisoa). MPAT: Mecanismo para Priorização de Alocação de Tarefas em VEC por Função de Tarefas Inspirado em Algoritmo Bioinspirado. 2025 (Apresentação de Trabalho/Conferência)

BROCK UNIVERSITY, 2023, (St. Catharines - ON - Canadá). Mini-curso: Resource allocation mechanisms in vehicular fog: Concepts, Challenges, and validation. 2023. (Palestra)