ORION: Escalonamento de Tarefas baseado em Otimização Multiobjetivo para Nuvens Veiculares

Matheus R. Esteves[1], Bruno Kimura[1], Maycon Peixoto[2], Geraldo Rocha[3], Leandro Villas[4], Allan M. de Souza[4], Joahannes B. D. da Costa[1]

[1]Universidade Federal de São Paulo (UNIFESP), São José dos Campos, Brasil
[2]Universidade Federal da Bahia (UFBA), Salvador, Brasil
[3]Universidade Estadual do Sudoeste da Bahia (UESB), Vitória da Conquista, Brasil
[4]Universidade Estadual de Campinas (UNICAMP), Campinas, Brasil

{matheus.esteves31, bruno.kimura, joahannes.costa}@unifesp.br
maycon.leone@ufba.br, geraldo.rocha@uesb.edu.br, {lvillas, allanms}@unicamp.br

Abstract. *Vehicular Edge Computing (VEC) has emerged as a promising paradigm to address the growing demand for low-latency computation in vehicular applications, driven by the rapid increase of connected vehicles and the massive generation of data. However, the dynamic and resource-constrained nature of this environment poses significant challenges for efficient computational task scheduling. This work proposes ORION, a multi-objective optimization-based scheduler that leverages the NSGA-II algorithm to balance conflicting objectives: maximizing the number of tasks completed within deadlines, minimizing monetary cost, and reducing system latency. Results show that ORION outperforms existing task scheduling solutions in VEC, particularly under high resource-demand scenarios, maintaining high scheduling rates, lowering costs, and achieving acceptable latency.*

Resumo. *A Computação de Borda Veicular (VEC) surge como um paradigma promissor para atender à crescente demanda por processamento de baixa latência em aplicações veiculares, impulsionada pelo aumento contínuo de veículos conectados e pela geração massiva de dados. No entanto, a natureza dinâmica e as restrições de recursos desse ambiente impõem desafios significativos para o escalonamento eficiente de tarefas computacionais. Neste trabalho, propõe-se o ORION, um escalonador baseado em otimização multiobjetivo que utiliza o algoritmo NSGA-II para equilibrar objetivos conflitantes: maximizar o número de tarefas concluídas dentro dos prazos, minimizar o custo monetário e reduzir a latência do sistema. Os resultados demonstram que o ORION supera outras soluções de escalonamento de tarefas em VEC, especialmente em cenários de alta demanda por recursos, mantendo altas taxas de escalonamento, reduzindo custos e alcançando níveis aceitáveis de latência.*

1. Introdução

O número de veículos automotores no Brasil e no mundo segue em crescimento. Conforme dados do Sindicato Nacional da Indústria de Componentes para Veículos Automotores, a frota brasileira já ultrapassou 64 milhões de unidades em 2025. Além disso, há também uma crescente no número dos veículos conectados, que são aqueles que possuem capacidade de comunicação com outras entidades, tais como semáforos inteligentes,

veículos, infraestruturas e Internet [Baumann et al. 2024]. Globalmente, estima-se que o número de carros conectados chegue a mais de 367 milhões em 2027 [Weissberger 2023].

Esse crescimento traz desafios como congestionamentos, acidentes e um volume massivo de dados gerados por sensores e câmeras embarcadas nos veículos. Nesse cenário, muitas aplicações de segurança e eficiência de tráfego, como alertas de colisão, navegação cooperativa e entretenimento para os ocupantes, necessitam de processamento rápido e baixa latência [Felix et al. 2025]. O modelo tradicional de enviar todos os dados para uma nuvem remota nem sempre é adequado para essas aplicações, devido a restrições de latência e potencial sobrecarga na rede [Bai et al. 2025].

Nesse contexto, surge a Computação de Borda Veicular, ou *Vehicular Edge Computing (VEC)* em inglês, que aplica o paradigma de computação de borda para aproximar os recursos de computação e armazenamento do local de geração dos dados, neste caso, os veículos [Lieira et al. 2025]. A VEC cria uma infraestrutura flexível, chamada Nuvem Veicular (ou *Vehicular Cloud (VC)* em inglês), que usa: *(i)* recursos dos próprios veículos (como unidades de processamento, memória, armazenamento e comunicação), que se agrupam dinamicamente; e *(ii)* servidores de borda implantados em pontos fixos próximos às vias, como unidades de beira de estrada para comunicação veicular (*Roadside Units (RSUs)*) ou estações base (*Base Stations (BSs)*) de rede celular.

De maneira geral, uma VC é constituída por um conjunto de veículos e/ou infraestruturas de comunicação, onde seus recursos computacionais serão agregados e disponibilizados na rede para utilização dos demais usuários veiculares. Normalmente, essa utilização é realizada seguindo políticas de pague pelo que você usa, ou o *pay-as-you-go* tradicional em infraestruturas de Computação em Nuvem [da Costa et al. 2023]. Ou seja, se um usuário precisa de recursos computacionais para processar uma tarefa e não possui recursos para isso, essa tarefa será enviada na rede e a própria VEC se encarrega de decidir onde a mesma será processada. O usuário se preocupa em apenas realizar o pagamento pelo tempo de uso do sistema. A VEC se encarrega de decidir onde a tarefa será processada (veículos ou BS) e precificar o uso desses recursos.

Em um cenário dinâmico e complexo como o da VEC, uma das questões centrais é: *como decidir de maneira eficiente qual tarefa computacional deve ser processada, onde (VC ou BS) e quando?* Esse problema é conhecido como escalonamento de tarefas. A eficiência desse escalonamento impacta diretamente o tempo de resposta das aplicações, o custo operacional e a satisfação do usuário final.

Diante disso, este trabalho apresenta um mecanismo de escalonamento de tarefas baseado em otimização multiobjetivo para nuvens veiculares, chamado **ORION**. O ORION, é executado nos controladores VEC e utiliza otimização multiobjetivo para decidir o escalonamento, ou atribuição *tarefa* $\rightarrow$ *recurso computacional*, que otimize aspectos de eficiência de operação. A otimização adotada gera um conjunto de soluções que satisfazem os critérios estabelecidos, organizadas em uma Fronteira de Pareto [Caiano et al. 2025]. Dentre as opções, uma é escolhida e aplicada ao sistema. Por exemplo, o ORION busca minimizar conjuntamente o custo monetário associado ao uso dos recursos computacionais e a latência de processamento das tarefas. Ao considerar simultaneamente um conjunto de soluções que atendam às funções objetivo, reduz-se a necessidade de reescalonamento de recursos e, consequentemente, o custo total. Os resultados obtidos indicam que o ORION é capaz de escalonar um maior número de tarefas, além

de minimizar os custos de utilização dos recursos e reduzir a latência do sistema, quando comparado a outras abordagens da literatura.

Este trabalho está organizado da seguinte forma. A Seção 2 discute os principais trabalhos relacionados. A Seção 3 apresenta a definição do problema e como o ORION funciona. A Seção 4 descreve o cenário de avaliação, a metodologia e discute os resultados obtidos. Por fim, a Seção 5 apresenta a conclusão e direções para trabalhos futuros.

2. Trabalhos Relacionados

Diversos trabalhos aplicaram modelos consolidados de tomada de decisão multicritério, otimização e teoria de filas ao problema de escalonamento em VEC. Por exemplo, [Pereira et al. 2021] propuseram o FORESAM, um mecanismo que utiliza o método de tomada de decisão multicritério *Analytic Hierarchy Process (AHP)* para decidir, de forma centralizada, se uma tarefa pode ser executada em uma VC. No entanto, trata-se de uma política que adota uma decisão do tipo gulosa (*greedy*) ao definir que se uma tarefa não se encaixa nos recursos da VC, ela é imediatamente descartada, o que pode impactar a eficiência global do sistema.

[da Costa et al. 2024] propuseram o TEMIS, um mecanismo de escalonamento que considera aspectos contextuais e aplica uma função de seleção probabilística nas VCs para equilibrar a carga de processamento e aumentar a equidade no uso dos recursos veiculares. O mecanismo utiliza uma abordagem de critérios duplos para encontrar o conjunto de Pareto, minimizando tempos de processamento e *deadlines*. Essa estratégia reduz o tempo de permanência das tarefas no sistema. Além disso, prioriza tarefas com requisitos de *deadline* mais restritos. Por fim, busca maximar as tarefas escalonadas aplicando uma heurística de aproximação para o problema de otimização *Bin Covering Problem (BCP)*.

Técnicas de otimização bioinspirada também têm sido exploradas. [Surayya et al. 2025] formularam um problema multiobjetivo para minimizar conjuntamente atraso e custo, resolvendo-o com um algoritmo baseado em *Particle Swarm Optimization (PSO)*. Esse método gera um conjunto de soluções (Fronteira de Pareto) e configurações, mas, como outros algoritmos evolucionários, enfrenta o desafio do tempo de convergência, que pode ser inadequado para decisões em tempo real. No entanto, embora o PSO seja eficaz na identificação de diversas configurações, pode ser necessário um refinamento adicional dos parâmetros para melhorar a precisão de suas soluções, o que dificulta sua aplicação em ambientes dinâmicos de tempo real.

Na mesma direção, [Lieira et al. 2025] apresentaram o LOPRIVE, um mecanismo bioinspirado de escalonamento de tarefas orientado por prioridades, baseado no *Lion Optimization Algorithm (LOA)*. Ao explorar o comportamento social do LOA, o mecanismo ajusta iterativamente as posições das tarefas, verificando sua "força" em relação aos recursos disponíveis, para maximizar tanto a eficiência do escalonamento quanto a priorização. Entretanto, embora o método priorize de forma eficaz tarefas críticas (de maior prioridade), ele depende de pesos de categoria estáticos e ainda não aborda variações nos perfis de usuários ou incentivos heterogêneos de compartilhamento de recursos.

Com a complexidade e dinamicidade do ambiente VEC, técnicas de aprendizado de máquina ganharam destaque por sua capacidade de aprender padrões e adaptar decisões. [Yang et al. 2024] introduziram uma abordagem que integra a previsão de demanda computacional com a tomada de decisão de escalonamento. Informações veicu-

lares são usadas para predizer demandas computacionais e permitir o pré-cache flexível de serviços de computação nas RSUs. Para a execução das tarefas, a abordagem emprega um algoritmo de aprendizado online baseado em *Multi Armed Bandit (MAB)*, que permite aos veículos aprenderem o desempenho dos nós de borda e tomarem decisões de escalonamento para múltiplas tarefas paralelas sem exigir informações completas sobre a topologia da rede, visando minimizar o atraso total das tarefas.

Com base na análise dos trabalhos relacionados, observa-se que muitas soluções tratam o problema de escalonamento de forma parcial, deixando de considerar simultaneamente fatores como dinamicidade da rede, heterogeneidade de recursos, diversidade de demandas computacionais e perfis de usuários. Além disso, diversas abordagens assumem parâmetros estáticos, decisões gulosas ou altos tempos de convergência, o que limita sua aplicabilidade em cenários VEC altamente dinâmicos.

3. Escalonamento de Tarefas baseado em Otimização Multiobjetivo

Esta seção descreve o ORION e está estruturada em três partes principais, sendo a visão geral do sistema, a definição formal do problema de escalonamento e a descrição do funcionamento da solução proposta.

3.1. Visão geral

No cenário VEC considerado, controladores de rede (controladores VEC) são responsáveis por coordenar R regiões pré-definidas da cidade, gerenciando tanto a formação de VC quanto o escalonamento de tarefas. A formação de VC envolve agrupar os recursos ociosos de veículos e infraestrutura, enquanto o escalonamento trata de utilizar tais recursos para processar as tarefas dos usuários. O ORION aborda somente o processo de escalonamento de tarefas, portanto, sem perda de generalidade, o processo de formação de VCs pode ser reduzido à simples associação entre veículo e infraestrutura durante a mobilidade veicular. Ou seja, enquanto o veículo estiver sob a cobertura de uma infraestrutura, seus recursos ociosos poderão ser utilizados pela VEC [da Costa et al. 2024].

Para que o escalonamento seja eficaz em um ambiente tão dinâmico como o veicular, três aspectos são considerados fundamentais: i) *Mobilidade Veicular*, pois a constante movimentação dos veículos causa mudanças frequentes na topologia da rede e na disponibilidade de recursos. Um bom mecanismo de escalonamento deve considerar esse aspecto para evitar perdas de tarefas devido a desconexões; ii) *Restrições de Prazo (Deadlines)*, onde muitas aplicações veiculares, especialmente as de segurança, exigem que os resultados das tarefas sejam entregues dentro de um tempo limite rigoroso. Tarefas que excedem seu prazo tornam-se inúteis; iii) *Custo Monetário*, onde seguindo o modelo de cobrança por uso comum em Computação em Nuvem, o custo do processamento é um fator relevante para o usuário final. Minimizar o custo, sem comprometer os prazos e atender o máximo de requisições possível, é um dos objetivos do escalonamento.

O cenário possui um conjunto de x veículos, indicados como $u_i \in U = \{u_1, u_2, \ldots, u_x\}$. Além disso, há um conjunto de p BSs implantadas na cidade, indicadas como $b_y \in B = \{b_1, b_2, \ldots, b_p\}$. Cada região $r \in R$ possui pelo menos uma BS e exatamente um controlador VEC. Além disso, existe um conjunto de nuvens veiculares $v_j \in V = \{v_1, v_2, ..., v_m\}$ disponíveis para processamento. Cada VC v_j possui uma capacidade total de *Central Processing Unit (CPU)* Ω_j, que é definida como o somatório das

capacidades individuais de processamento de todos os seus membros, englobando tanto os recursos provenientes dos veículos (ω_{u_i}) quanto das BSs (ω_{b_y}), em *Milhões de Instruções Por Segundo (MIPS)*, e uma capacidade total de armazenamento Φ_j, em *Megabyte (MB)*.

3.2. Definição do problema

O problema de escalonamento abordado pode ser formalizado da seguinte maneira. Considere um conjunto de tarefas $T = \{t_1, t_2, ..., t_n\}$, no qual cada tarefa $t_l \in T$ é representada pela tupla $\langle id_l, s_l, w_l, D_l \rangle$. Nessa tupla, id_l é um identificador único, s_l é o tamanho de dados de entrada, em MB, w_l representa a quantidade de instruções necessárias para seu processamento, em Milhões de Instruções (MI), e D_l denota o prazo máximo (*deadline*) para sua conclusão, em segundos.

Como os recursos de uma VC são compartilhados entre as tarefas nela alocadas, a capacidade efetiva de processamento Ψ_j para cada tarefa é dada pela capacidade total Ω_j dividida pelo número de tarefas alocadas naquela VC ($|T'_j|$), resultando em $\Psi_j = \frac{\Omega_j}{|T'_j|}, j \in V$. Sendo assim, o tempo de processamento d_{lj} de uma tarefa é obtido pela razão entre sua carga computacional w_l e a capacidade de processamento Ψ_j da máquina na qual será executada. Assim, $d_{lj} = \frac{w_l}{\Psi_j}, \forall t_l \in v_j$.

Como cada tarefa possui um prazo máximo, para uma tarefa ser considerada atendida com sucesso, seu tempo de processamento deve ser menor ou igual ao seu prazo, sendo $d_{lj} \leq D_l$. Além disso, quando uma tarefa é escalonada e começa a ser processada, há um custo monetário associado a essa alocação de recursos. Esse custo monetário C_{t_l} é modelado com base no tempo de processamento e nos ciclos de CPU utilizados, multiplicados por um preço unitário[1] pré-definido. Esse preço difere se a tarefa for executada em recursos de infraestrutura de comunicação (BS b_y) ou em recursos veiculares (u_i), refletindo custos operacionais distintos, conforme Equação (1):

$$C_{t_l} = \begin{cases} d_{lj} \times (w_l \times 10.34592), & \text{se } t_l \text{ utiliza recursos de } b_y, \\ d_{lj} \times (w_l \times 5.17296), & \text{se } t_l \text{ utiliza recursos de } u_i. \end{cases} \quad (1)$$

Em resumo, o objetivo do escalonador é selecionar um subconjunto de tarefas $T' \subseteq T$ para ser processado nas VCs disponíveis, maximizando o número de tarefas concluídas dentro do prazo e, ao mesmo tempo, minimizando o custo monetário total dessa utilização dos recursos. Trata-se, por natureza, de um problema de otimização multiobjetivo. Assim, as principais restrições consideradas são: (1) o prazo de execução de cada tarefa deve ser respeitado; (2) a soma dos tamanhos das tarefas alocadas em uma VC não pode exceder sua capacidade de armazenamento; e (3) a soma dos ciclos de CPU requeridos não pode ultrapassar a capacidade de processamento da VC.

3.3. Funcionamento do ORION

A Figura 1 apresenta o processo de escalonamento de tarefas no ORION. O processo é composto por três etapas principais. Primeiramente, múltiplas requisições chegam ao sistema. Tais requisições são tarefas de usuários que não possuem recursos computacionais suficientes para executá-las. As tarefas são enfileiradas no sistema (Etapa ①), podendo

[1]Os valores de preço são baseados em instâncias comerciais da Amazon EC2 na região *us-east-2* (*c8a.48xlarge* e *c8a.metal-24xl*), utilizadas como referência para os custos de b_y e u_i, respectivamente.

assumir dois estados, inicialmente *em espera* (tarefa ainda aguardando ser executada) ou *escalonada* (a tarefa foi direcionada para uma VC). A tarefa só muda do estado inicial (*em espera*) quando é selecionada por um mecanismo de escalonamento (Etapa ②) e é direcionada para processamento. O escalonamento pode ser realizado de diferentes maneiras, seguindo critérios distintos, como a priorização de tarefas mais curtas, de tarefas com *deadline* mais restrito ou daquelas que chegam primeiro à fila. Por fim, as tarefas são executadas nas VCs disponíveis no ambiente (Etapa ③). Todo o processo é monitorado pelos controladores VEC e uma tarefa só é removida da fila quando seu processamento é concluído ou seu *deadline* D_l é atingido.

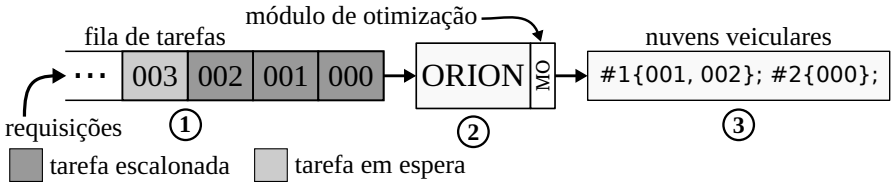


Figura 1. Pipeline de processamento de tarefas do sistema.

O ORION avalia as tarefas na fila e toma decisões de escalonamento com base em uma análise conjunta de diferentes critérios, como o peso (requisitos de recursos computacionais) e o *deadline* das tarefas. A ideia central é equilibrar a prioridade de tarefas com menor *deadline*, ou seja, aquelas que não podem aguardar longos períodos para obter o resultado do processamento, com a quantidade de recursos computacionais necessários para sua execução. Conforme discutido anteriormente, esse processo de seleção envolve múltiplos objetivos, de modo que algoritmos específicos para otimização multiobjetivo devem ser considerados. O ORION possui um módulo de otimização (MO) para esse fim.

No entanto, para seleção do algoritmo de otimização a ser implementado, um conjunto de algoritmos foram testados para esse contexto em específico. Foram considerados três algoritmos de otimização multiobjetivo amplamente reconhecidos na literatura, a saber: Otimização por Enxame de Partículas (PSO), *Non-dominated Sorting Genetic Algorithm (NSGA)-II* e NSGA-III. Após análise preliminar do desempenho de cada algoritmo, resultados que serão apresentados e discutidos na Subseção 4.2.1, o NSGA-II foi selecionado como algoritmo para tomada de decisão do ORION.

O NSGA-II é um algoritmo evolutivo amplamente utilizado para resolver problemas de otimização multiobjetivo, especialmente quando existem objetivos conflitantes, como minimizar o custo monetário e maximizar o número de tarefas atendidas dentro do prazo. Seu funcionamento baseia-se na evolução de uma população de soluções candidatas ao longo de várias iterações. A cada geração, as soluções são avaliadas considerando todos os objetivos do problema e organizadas por meio de um processo de ordenação não-dominada, que identifica as soluções[2] pertencentes à Fronteira de Pareto, isto é, conjuntos de soluções Pareto-ótimas. Além disso, o algoritmo emprega a métrica de distância de aglomeração para manter diversidade entre as soluções, evitando que a busca se concentre em apenas uma região do espaço de soluções. Operadores genéticos, como cruzamento e mutação, são aplicados para gerar novas soluções, o que permite explorar diferentes possibilidades de alocação de tarefas até se obter um conjunto de escalonamentos eficientes.

[2]Uma solução candidata representa um escalonamento completo, isto é, uma atribuição das tarefas do conjunto T às VCs disponíveis, respeitando as restrições de prazo e capacidade.

Para integrar o ORION ao ambiente VEC, foi desenvolvido um ciclo de operação do sistema descrito no Algoritmo 1, responsável por gerenciar a execução das tarefas. O algoritmo opera em intervalos de tempo $k \in K$, nos quais novas tarefas são continuamente adicionadas à fila de espera (Linhas 1 e 2). Sempre que houver tarefas pendentes, o mecanismo ORION é acionado para realizar o escalonamento, produzindo o conjunto de tarefas escalonadas $\mathcal{S}$ com base nos recursos disponíveis no conjunto de VCs V (Linhas 3 e 4). Em seguida, cada tarefa escalonada é monitorada durante sua execução: caso seja concluída, seu custo é calculado conforme a Equação (1), a tarefa é removida da fila de espera e o resultado é retornado ao usuário (Linhas 6 a 9). Caso contrário, a estimativa de progresso da tarefa é atualizada para refletir seu estado atual (Linha 11). Esse fluxo evidencia o papel do escalonador como componente central no gerenciamento dinâmico das tarefas no ambiente VEC.

Algoritmo 1: Operação da VEC com o ORION integrado.

```
1  para cada intervalo de tempo k ∈ K faça
2      Adiciona novas tarefas na fila de espera
3      se a fila de tarefas não estiver vazia então
4          S ← ORION(T, V)                    ▷ Aciona mecanismo de escalonamento
5          para cada s ∈ S faça
6              se s concluiu execução então
7                  Calcular custo com a Equação (1)
8                  Remove tarefa da fila de espera
9                  Retorna resultado ao usuário
10             senão
11                 Atualizar estimativa de progresso de s
```

O Algoritmo 2 descreve como o ORION adapta o algoritmo NSGA-II para o escalonamento de tarefas no ambiente VEC. O algoritmo recebe como entrada o conjunto de tarefas T e o conjunto de VCs disponíveis V. Inicialmente, são extraídos os prazos D_l de cada tarefa (Linha 1) e calculados os tempos de processamento estimados PT para cada combinação entre tarefas e VCs (Linha 2). Em seguida, é inicializada uma população P com `POP` soluções viáveis, garantindo que as restrições de CPU e armazenamento das VCs sejam respeitadas por meio de um operador de reparo (Linha 3). A partir dessa população, o algoritmo executa um processo evolutivo por `GENS` gerações (Linha 4). Em cada geração, uma população filha P_{child} é construída (Linha 5) a partir da seleção de pais via torneio binário (Linha 7). Os pais selecionados passam por *crossover* com probabilidade `CXPROB` (Linha 8) e mutação com probabilidade `MUTPROB` (Linha 9), gerando novas soluções. Essas soluções são reparadas para manter a viabilidade (Linha 10) e adicionadas à população filha (Linha 11). Após combinar as populações pai e filha (Linha 12), as soluções são classificadas em Fronteiras de Pareto segundo o critério de não-dominância (Linha 13). Para preservar diversidade, calcula-se a *crowding distance* (Linha 14) e selecionam-se os `POP` melhores indivíduos para a próxima geração (Linha 15). Ao final, identifica-se a primeira fronteira F_1 (Linha 16) e escolhe-se a solução s^* com menor custo monetário total (Linha 17). O conjunto final de tarefas escalonadas T' é então obtido (Linha 18) e retornado como resultado do ORION (Linha 19). Os parâmetros evolutivos (operadores genéticos) considerados foram: $\texttt{POP} = 30$, $\texttt{GENS} = 10$, $\texttt{CXPROB} = 0.9$, e $\texttt{MUTPROB} = 0.15$.

Algoritmo 2: Pseudocódigo do ORION

Entrada: conjunto de tarefas T e conjunto de VCs V
Saída: subconjunto de tarefas escalonadas T'

Extrair prazos $D \leftarrow \{D_l \mid t_l \in T\}$
Calcular $PT \leftarrow \{\text{tempo}(t_l, v_j) \mid t_l \in T, v_j \in V\}$
Inicializar população P com `POP` soluções viáveis
para $gen = 1$ *até* `GENS` **faça**
    $P_{\text{child}} \leftarrow \varnothing$
    enquanto $|P_{child}| <$ `POP` **faça**
        Selecionar pais p_1, p_2 de P via torneio binário
        Aplicar *crossover* a p_1, p_2 com probabilidade `CXPROB`
        Aplicar mutação aos filhos com probabilidade `MUTPROB`
        Reparar filhos para garantir viabilidade
        $P_{\text{child}} \leftarrow P_{\text{child}} \cup \{\text{filhos}\}$
    $P \leftarrow P \cup P_{\text{child}}$
    Classificar P em Fronteiras de Pareto (não-dominância)
    Calcular *crowding distance* para cada solução
    Selecionar os `POP` melhores indivíduos para a próxima geração
Identificar a primeira frente de Pareto F_1
Selecionar $s^* \in F_1$ com menor custo monetário total
$T' \leftarrow$ tarefas alocadas conforme s^*
retorna T'

4. Avaliação de Desempenho

Esta seção descreve a metodologia e as métricas utilizadas para avaliar a eficiência do ORION em comparação com outras soluções de escalonamento em VEC. Além disso, apresenta e discute os resultados obtidos.

4.1. Metodologia

Para avaliar o desempenho dos mecanismos de escalonamento de tarefas, foi construído um ambiente de simulação que reproduz as características dinâmicas e os desafios de um cenário real de ambiente VEC. O ambiente integra o *Simulator of Urban Mobility (SUMO)*, na versão 1.18.0, com os algoritmos que foram implementados na linguagem Python 3.10. Os algoritmos foram conectados ao SUMO via interface TraCI, permitindo um ciclo de coleta de dados e tomada de decisão em tempo real. Para uma maior granularidade na simulação, o passo de tempo (parâmetro `--step-length` no SUMO)[3] configurado foi de 0.1 segundos. As simulações foram executadas em uma máquina com processador Intel® Core™ i7-12700H 20×(3.50, 4.70) GHz e Linux Ubuntu 22.04 LTS.

O cenário de mobilidade considerado é baseado em um recorte do *trace* realístico da cidade de Colônia, Alemanha[4], com área total de aproximadamente 114 km^2. Durante a simulação, com duração de 600 segundos (sendo os 100 segundos iniciais destinados ao *warm-up*), o número de veículos ativos varia ao longo do tempo, atingindo um pico de 350 veículos. Essa dinâmica cria um cenário desafiador para o gerenciamento de recursos no ambiente veicular, pois a demanda por recursos é mantida superior à oferta disponível. Cada experimento foi replicado utilizando cinco diferentes sementes de aleatoriedade.

[3] `https://sumo.dlr.de/docs/sumo.html#time`
[4] `https://sumo.dlr.de/docs/Data/Scenarios/TAPASCologne.html`

Foram implantadas 16 infraestruturas de comunicação no cenário, que nesse caso são BSs de rede 5G, com alcance de comunicação de 2000 metros, cada uma contando com 15 MIPS de capacidade de processamento. A cidade foi dividida em 4 regiões, cada uma sob a gerência de um controlador VEC. Os veículos, com alcance de comunicação de 250 metros, contribuem voluntariamente com 1 MIPS e 1 MB de armazenamento cada, formando VCs dinâmicas. Essas atribuições de MIPS para BS e veículo é apenas para haver uma diferenciação entre esses elementos, sem perda de generalidade.

A carga de trabalho é composta por tarefas do tipo *Bag-of-Tasks (BoT)*, que chegam ao sistema seguindo uma distribuição de Poisson [da Costa et al. 2024]. Para testar a robustez dos algoritmos sob diferentes condições de carga, foram avaliadas três taxas de chegada de tarefas no sistema ($\lambda = 5, 15, \text{e } 30$) tarefas por segundo. Cada tarefa é caracterizada por um tamanho de entrada entre 1 e 10 MB, uma demanda computacional entre 1 e 30 MI, e um deadline máximo. Foram considerados quatro níveis de prazos (*deadlines*), sendo $D_l = 0.5, 1, 5$ e 7 segundos, permitindo analisar o desempenho desde aplicações mais sensíveis à latência até aquelas mais tolerantes.

O ORION foi comparado a outros mecanismos de escalonamento em VEC propostos na literatura, a saber: *First-Come, First-Served (FCFS)*, linha de base de simplicidade que atende às tarefas estritamente na ordem de chegada na fila. A abordagem é baseada em [Hattab et al. 2019]; *MAB*, que emprega uma estratégia baseada em aprendizado por reforço para equilibrar a exploração de novos recursos com a exploração das combinações que performam melhor naquele escalonamento. A abordagem é baseada em [Yang et al. 2024]; *LOA*, que implementa um processo decisório bioinspirado baseado no algoritmo de otimização de leões, e que pondera os requisitos das tarefas e a capacidade das VCs. A abordagem é baseada em [Lieira et al. 2025]; e *TEMIS*, uma abordagem que combina heurística de aproximação do BCP e otimização de Pareto para minimização conjunta de custo monetário e tempos de processamento [da Costa et al. 2024].

A avaliação de desempenho foi realizada com base em quatro métricas principais, que fornecem uma visão holística da eficiência e viabilidade de cada solução, sendo elas: i) *Tarefas Escalonadas (%)*: porcentagem de tarefas executadas com sucesso dentro do prazo; ii) *Custo Monetário ($)*: custo agregado do uso de recursos de processamento em veículos e BSs; iii) *Latência do Sistema (s)*: latência total, incluindo tempo de espera na fila e execução, refletindo a Qualidade de Serviço (QoS) percebida; iv) *Tempo de Uso de CPU (ms)*: tempo de CPU consumido pelo algoritmo de escalonamento, indicando sua viabilidade em cenários de tempo real.

4.2. Resultados

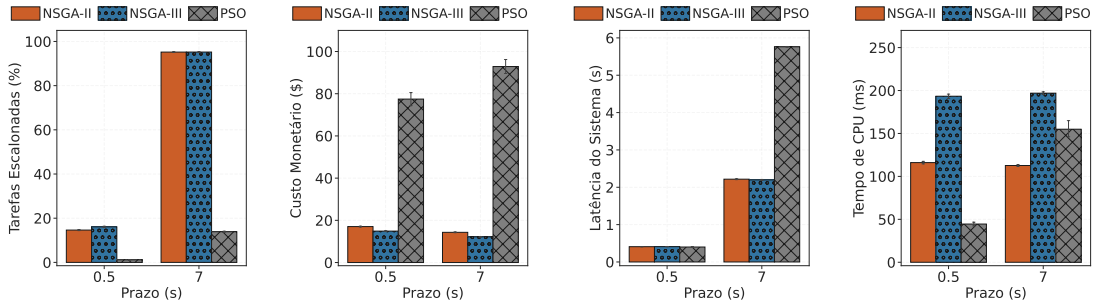
Esta seção apresenta os resultados das simulações em duas partes: a primeira discute a escolha do algoritmo de otimização multiobjetivo do ORION, e a segunda compara o ORION com abordagens de escalonamento da literatura.

4.2.1. Optimização multiobjetivo

Para selecionar o algoritmo de otimização multiobjetivo a ser integrado ao mecanismo ORION, foram avaliados os três algoritmos indicados na Subseção 3.3. A análise comparativa foi conduzida com base nas quatro métricas definidas na seção anterior, consi-

derando o cenário de alta demanda ($\lambda = 30$ tarefas/segundo) e dois extremos de prazos máximos (*deadlines*), fixados em $D_l = 0.5$ e $D_l = 7$ segundos.

Inicialmente, analisou-se a porcentagem de tarefas escalonadas com sucesso (Figura 2(a)). Observa-se que, para o prazo mais restritivo ($D_l = 0.5$), o desempenho global é reduzido para todos os algoritmos, porém o NSGA-II e o NSGA-III apresentam taxas de escalonamento significativamente superiores ao PSO, evidenciando maior robustez em cenários críticos. Quando o prazo é ampliado para $D_l = 7$, tanto o NSGA-II quanto o NSGA-III alcançam percentuais elevados de tarefas escalonadas, próximos a 93%, enquanto o PSO mantém um desempenho consideravelmente inferior. Esses resultados indicam que os algoritmos baseados em NSGA são mais eficazes para lidar com alta carga e requisitos temporais mais flexíveis.



(a) Tarefas escalonadas. (b) Custo monetário. (c) Latência do sistema. (d) Tempo de CPU.

Figura 2. Desempenho dos algoritmos de otimização multiobjetivo com $\lambda = 30$, considerando dois extremos de *deadlines* $D_l = 0.5$ e $D_l = 7.0$, respectivamente.

Em relação ao custo monetário total do processamento, ilustrado na Figura 2(b), nota-se que o NSGA-II e o NSGA-III apresentam custos muito próximos entre si em ambos os prazos analisados, sendo consistentemente inferiores aos custos observados para o PSO. A análise da latência do sistema, apresentada na Figura 2(c), revela que o NSGA-II e o NSGA-III mantêm níveis de latência baixos e muito semelhantes entre si. Especialmente quando o prazo é maior ($D_l = 7$), a latência do PSO cresce de forma significativa, enquanto o NSGA-II e o NSGA-III preservam um comportamento estável, indicando maior eficiência na gestão das filas e do tempo de execução das tarefas.

Por fim, a métrica de tempo de uso de CPU pelo algoritmo de escalonamento, apresentada na Figura 2(d), é fundamental para avaliar sua viabilidade de implantação em ambientes de tempo real. Os resultados indicam que o NSGA-II apresenta um custo computacional intermediário, sendo mais eficiente que o NSGA-III em ambos os cenários analisados. O PSO, por sua vez, destaca-se pelo menor tempo de processamento, embora esse ganho ocorra em detrimento do desempenho nas demais métricas.

Nesse contexto, o NSGA-II apresenta um equilíbrio entre eficiência computacional e qualidade das soluções, enquanto o NSGA-III, apesar de eficaz em termos de escalonamento e latência, exige um esforço computacional significativamente superior. Adicionalmente, o NSGA-II é amplamente utilizado em problemas que envolvem até três objetivos conflitantes, porém pode apresentar limitações em situações com um número superior de objetivos, contexto no qual o NSGA-III demonstra maior robustez.

4.2.2. Escalonamento de tarefas

Com o algoritmo NSGA-II integrado ao ORION, realizou-se a comparação de seu desempenho com outras abordagens da literatura. A avaliação, fundamentada nas mesmas quatro métricas e cenários apresentados na Subseção 4.1, busca identificar a solução que oferece o melhor equilíbrio entre eficiência, custo, qualidade de serviço e viabilidade computacional em um ambiente dinâmico de VEC.

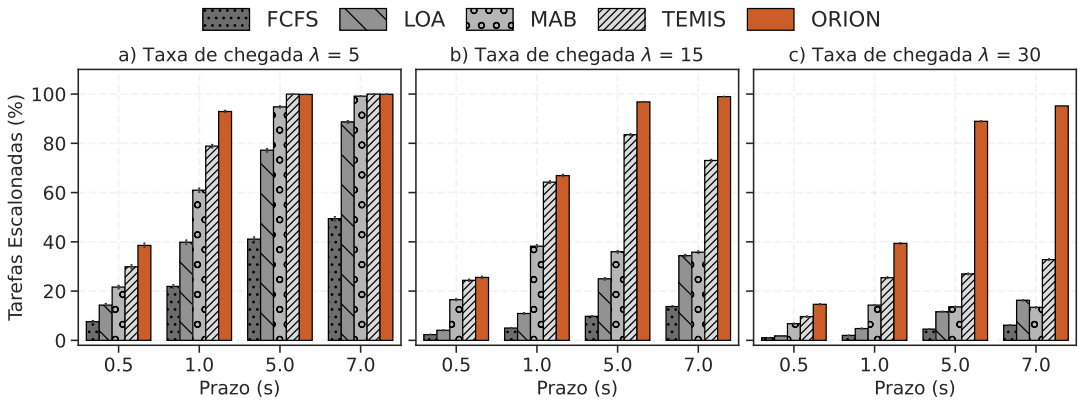


Figura 3. Resultados de tarefas escalonadas com sucesso.

A Figura 3 apresenta resultados de porcentagem de tarefas escalonadas. O ORION se destaca como a solução mais eficaz entre as abordagens avaliadas, mantendo consistentemente as maiores taxas de atendimento em diferentes cenários de carga e prazo. Mesmo em condições críticas de alta intensidade de chegada ($\lambda = 30$) e *deadlines* restritivos, o ORION preserva desempenho superior, demonstrando robustez e alta capacidade de adaptação à dinamicidade da rede. Em contraste, soluções como FCFS e LOA apresentam quedas expressivas na taxa de sucesso conforme os prazos se tornam mais curtos e a carga aumenta, evidenciando limitações em situações de congestionamento elevado. O MAB e o TEMIS, embora incorporem estratégias adaptativas, obtêm resultados intermediários, superando métodos simples em alguns casos, mas sem alcançar o nível de consistência do ORION, sobretudo sob maior demanda. Observa-se ainda que, mesmo em prazos mais longos, apenas o ORION mantém percentuais elevados de escalonamento na carga mais alta, reforçando sua escalabilidade.

No que diz respeito ao custo monetário total (Figura 4), observa-se que o ORION se apresenta como a solução mais econômica em todos os cenários avaliados, destacando-se mesmo sob elevadas taxas de chegada ($\lambda = 15$ e $\lambda = 30$). Enquanto o FCFS e LOA exibem custos significativamente maiores, sobretudo conforme os prazos aumentam, o ORION mantém valores baixos e estáveis, evidenciando sua eficiência na seleção de recursos com menor impacto financeiro. O MAB e o TEMIS, embora mais adaptativos, apresentam custos intermediários, com o TEMIS controlando melhor os gastos em cargas moderadas, mas ainda sem alcançar o desempenho consistente do ORION em cenários críticos. Nota-se também que o MAB tende a oscilar em função do caráter exploratório, podendo incorrer em alocações temporariamente sub-ótimas durante o aprendizado. Assim, os resultados reforçam que o ORION não apenas melhora o atendimento das tarefas, mas também oferece uma vantagem expressiva em termos de viabilidade econômica, fator essencial para aplicações em larga escala em ambientes dinâmicos de VEC.

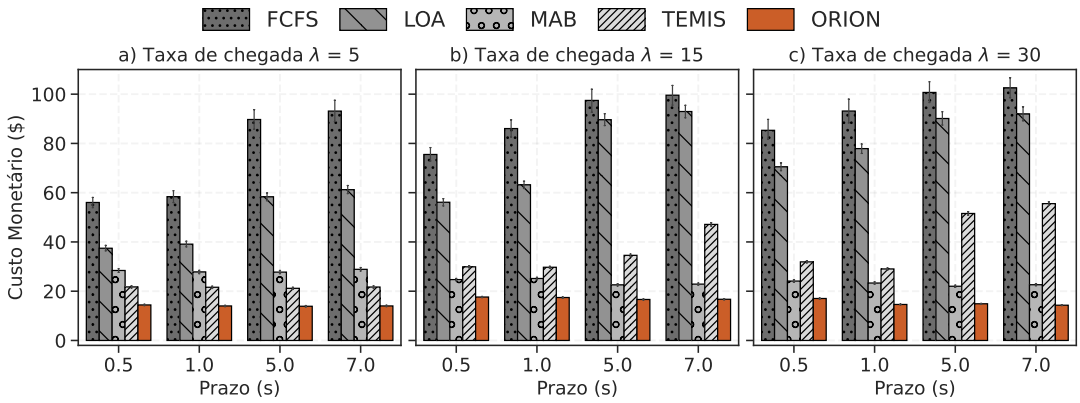


Figura 4. Resultados de custo monetário.

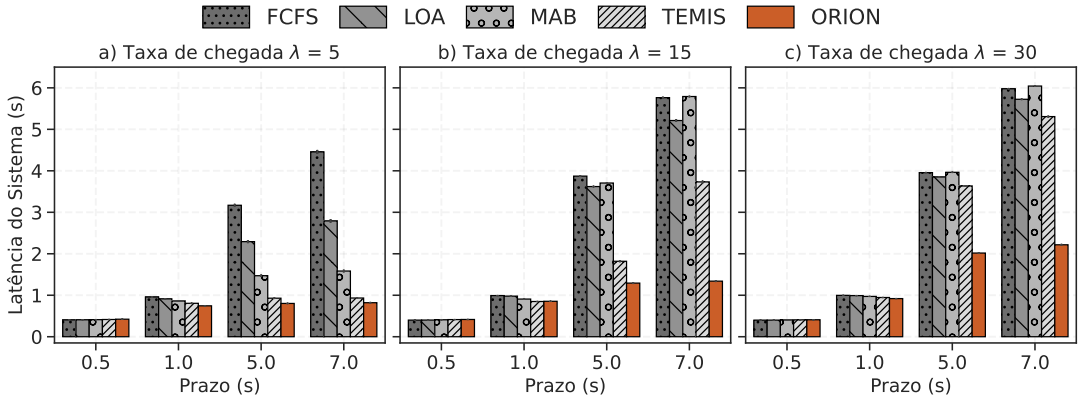


Figura 5. Resultados de latência do sistema.

Quanto à latência total do sistema (Figura 5), observa-se que o ORION mantém desempenho superior, apresentando consistentemente as menores latências médias em praticamente todos os cenários avaliados, especialmente sob condições de maior carga ($\lambda = 15$ e $\lambda = 30$). Enquanto métodos tradicionais como FCFS e LOA exibem crescimento acentuado da latência à medida que os prazos aumentam, o ORION controla de forma mais eficiente o tempo de resposta, evitando atrasos excessivos mesmo em situações críticas. O TEMIS também obtém resultados competitivos, com latências relativamente baixas, porém ainda superiores às do ORION, principalmente em prazos mais longos, onde as diferenças se tornam mais evidentes. Mesmo o TEMIS tendo como objetivo a minimização de latência, seu processo de funcionamento não explora o espaço de soluções viáveis de forma otimizada. Já o MAB apresenta comportamento intermediário, mas tende a oscilar em função do processo exploratório, o que pode resultar em decisões sub-ótimas e aumento de atraso. Dessa forma, os resultados reforçam que o ORION é a abordagem mais eficaz para minimizar latência, garantindo maior responsividade e confiabilidade em ambientes dinâmicos de VEC.

Por fim, a métrica de tempo de uso de CPU pelo algoritmo de escalonamento (Figura 6) evidencia que o ORION demanda o maior custo computacional entre as abordagens avaliadas, com crescimento expressivo conforme a taxa de chegada aumenta, especialmente em $\lambda = 15$ e $\lambda = 30$. Enquanto soluções mais simples como FCFS e

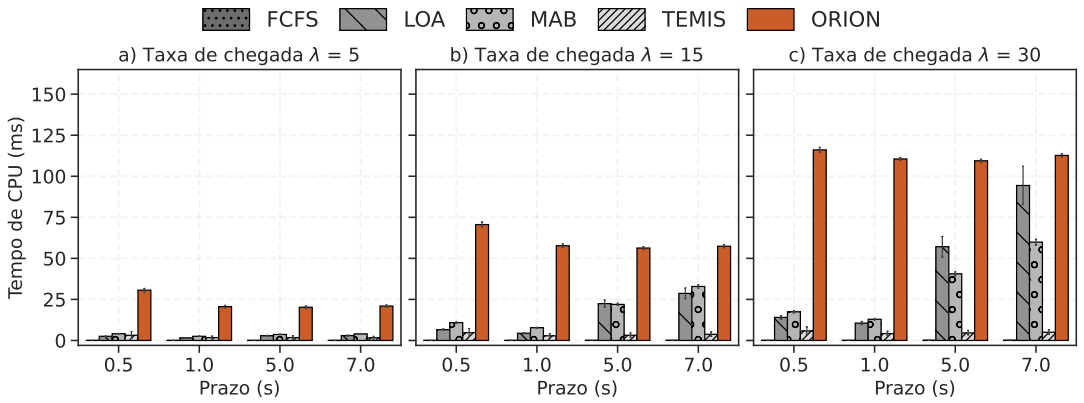


Figura 6. Resultados de tempo total de uso de CPU.

TEMIS mantêm baixos tempos de processamento devido à reduzida complexidade decisória, o ORION apresenta um consumo significativamente superior, refletindo o esforço necessário para realizar otimizações multiobjetivo mais elaboradas. O MAB e o LOA permanecem como soluções intermediárias, sendo que o LOA apresenta uma maior variação devido ao seu processo de exploração de soluções. Ainda assim, apesar do custo adicional, o desempenho do ORION nas demais métricas (como maior taxa de tarefas escalonadas, menor latência e menor custo monetário) sugere que esse aumento no tempo de CPU representa um *trade-off* justificável para aplicações que demandam maior qualidade de serviço e decisões mais robustas em ambientes dinâmicos de VEC.

O ORION demonstrou eficiência em relação aos algoritmos comparados, permitindo balancear dinamicamente metas conflitantes como cumprimento de prazos, minimização de custos e redução de latência. Enquanto abordagens tradicionais como FCFS e LOA utilizam políticas estáticas e não consideram explicitamente a criticidade das tarefas nem a eficiência econômica, o ORION avalia cada requisição conforme o estado do sistema, selecionando alocações mais robustas. O MAB e o TEMIS, embora adaptativos, apresentam desempenho intermediário: o primeiro pode gerar decisões subótimas durante o aprendizado, e o segundo, apesar do baixo custo computacional, não alcança o mesmo equilíbrio sob alta carga por não explorar o espaço de soluções viáveis de forma otimizada. Ainda que demande maior tempo de CPU, esse custo reflete a complexidade da busca evolucionária baseada no NSGA-II, responsável por soluções mais balanceadas. Assim, valida-se a eficácia do uso do NSGA-II como núcleo de decisão multiobjetiva do ORION, essencial para sustentar operações em ambientes de VEC com restrições diversas e alta variabilidade de carga.

5. Conclusão

Este trabalho propôs o ORION, um escalonador baseado em otimização multiobjetivo que emprega o algoritmo NSGA-II para balancear objetivos conflitantes, tais como a maximização do número de tarefas atendidas dentro do prazo e a minimização do custo monetário. A implementação e a validação do ORION foram realizadas em um ambiente de simulação integrado ao simulador SUMO, utilizando um cenário realístico da cidade de Colônia, Alemanha, com mobilidade veicular e cargas de trabalho variáveis. Os resultados obtidos demonstraram que o ORION apresentou um desempenho superior ao de abordagens de escalonamento da literatura, especialmente em cenários de alta carga e prazos

restritivos. O mecanismo proposto destacou-se pela capacidade de manter altas taxas de tarefas escalonadas dentro do prazo, ao mesmo tempo em que reduziu custos monetários e latência do sistema. A escolha do NSGA-II como núcleo de otimização mostrou-se acertada, oferecendo um equilíbrio adequado entre eficácia no escalonamento/alocação e viabilidade computacional. Trabalhos futuros incluem a investigação de mecanismos adaptativos para ajustar dinamicamente os parâmetros do NSGA-II e a incorporação de outras métricas relevantes, como balanceamento de carga e justiça, visando uma avaliação mais completa e o aprimoramento da robustez do ORION.

Disponibilidade de Artefatos

Em conformidade aos princípios da Ciência Aberta, o código-fonte e os dados utilizados neste trabalho estão disponíveis em: `https://github.com/joahannes/ORION`.

Agradecimentos

Este projeto foi apoiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) por meio do processo #2023/00673-7. Este projeto também foi apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-SOFTEX, coordenado pela Softex e publicado Arquitetura Cognitiva (Fase 3), DOU 01245.003479/2024-10.

Referências

Bai, L., Cao, J., Zhang, M., and Li, B. (2025). Collaborative Edge Intelligence for Autonomous Vehicles: Opportunities and Challenges. *IEEE Network*.

Baumann, D., Sommer, M., Dettinger, F., Rösch, T., Weyrich, M., and Sax, E. (2024). Connected Vehicle: Ontology, Taxonomy and Use Cases. In *2024 IEEE International Systems Conference (SysCon)*, pages 1–6. IEEE.

Caiano, S. R., Marfort, L., and Tang, J. (2025). Otimização Multiobjetivo Eficiente de Sistemas AEB: Um Estudo Comparativo entre Sweeping e NSGA-II. *Blucher Engineering Proceedings*, 12(1):489–493.

da Costa, J. B., de Souza, A. M., Meneguette, R. I., Cerqueira, E., Rosário, D., Sommer, C., and Villas, L. (2023). Mobility and Deadline-Aware Task Scheduling Mechanism for Vehicular Edge Computing. *IEEE Transactions on Intelligent Transportation Systems*, 24(10):11345–11359.

da Costa, J. B., de Souza, A. M., Rosário, D., and Villas, L. (2024). TEMIS: Provisionamento de Justiça na Utilização de Recursos Computacionais em Nuvens Veiculares. In *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pages 15–28. SBC.

Felix, A. A., da Costa, J. B., and Oliveira, H. M. d. S. (2025). Assessing vehicle collision prevention based on machine learning and v2x communication. In *2025 IEEE 102nd Vehicular Technology Conference (VTC2025-Fall)*, pages 1–5. IEEE.

Hattab, G., Ucar, S., Higuchi, T., Altintas, O., Dressler, F., and Cabric, D. (2019). Optimized Assignment of Computational Tasks in Vehicular Micro Clouds. In *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*, pages 1–6.

Lieira, D. D., Quessada, M. S., De Grande, R. E., and Meneguette, R. I. (2025). LOPRIVE: LOA-Based Priority-Driven Task Allocation in the Vehicular Edge. In *2025 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE.

Pereira, R., Boukerche, A., da Silva, M. A., Nakamura, L. H., Freitas, H., Rocha Filho, G. P., and Meneguette, R. I. (2021). FORESAM-FOG Paradigm-Based Resource Allocation Mechanism for Vehicular Clouds. *Sensors*, 21(15):5028.

Surayya, A., Hussain, M. M., Reddy, V. D., Abdul, A., and Gazi, F. (2025). Evolutionary Algorithms for Edge Server Placement in Vehicular Edge Computing. *IEEE Access*.

Weissberger, A. (2023). Juniper research: 5g connectivity opportunity for the connected car market. Acessado em 13 jan. 2026.

Yang, J., Yang, K., Dai, X., Xiao, Z., Jiang, H., Zeng, F., and Li, B. (2024). Service-aware computation offloading for parallel tasks in vec networks. *IEEE Internet of Things Journal*.