TEMIS: Provisionamento de Justiça na Utilização de Recursos Computacionais em Nuvens Veiculares

Joahannes B. D. da Costa[1], Allan M. de Souza[1], Denis Rosário[2], Leandro Villas[1]

[1]Universidade Estadual de Campinas (UNICAMP), Brasil
[2]Universidade Federal do Pará (UFPA), Brasil
{jbdc, allanms, lvillas}@unicamp.br, denis@ufpa.br

Abstract. *Vehicular Edge Computing (VEC) provides cloud computing services closer to vehicular users by combining computational resources from vehicles and edge nodes to form Vehicular Clouds (VCs). In this scenario, a task scheduler must decide which tasks will be executed on the available VCs, considering contextual aspects such as vehicular mobility and the requirements of these tasks. This is crucial to minimize both processing time and monetary costs associated with resource utilization. However, this direct optimization can lead to resource imbalance, degrading the overall efficiency of the system in terms of performance and fairness in workload distribution. In this context, this work introduces TEMIS, a task scheduling mechanism that takes contextual aspects into account in its decision-making process and applies a probabilistic selection function on VCs to balance processing load and increase fairness in resource utilization. Compared to state-of-the-art solutions, TEMIS demonstrates a higher level of fairness in resource utilization and can schedule a greater number of tasks while minimizing monetary costs and system latency.*

Resumo. *A Computação de Borda Veicular (VEC) fornece serviços de computação em nuvem mais próximo aos usuários veiculares, combinando recursos computacionais de veículos e de nós na borda da rede para a formação de Nuvens Veiculares (VCs). Nesse cenário, um escalonador de tarefas deve decidir quais tarefas serão executadas nas VCs disponíveis, considerando aspectos como mobilidade veicular e os requisitos das tarefas. Isso é importante para minimizar tanto o tempo de processamento quanto os custos monetários relacionados a utilização dos recursos. No entanto, essa otimização direta pode levar ao desbalanceamento no uso de recursos, degradando a eficiência do sistema em termos de performance e justiça na distribuição das cargas de trabalho. Nesse sentido, este trabalho apresenta o TEMIS, um escalonador de tarefas 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. Comparado às soluções da literatura, o TEMIS apresenta um nível mais elevado de justiça na utilização dos recursos e pode escalonar um maior número de tarefas, ao mesmo tempo em que minimiza custos monetários e a latência do sistema.*

1. Introdução

A Computação de Borda Veicular (*Vehicular Edge Computing (VEC)*) surgiu como um paradigma que fornece poder de computação e armazenamento próximo aos usuários veiculares [Xue et al. 2023]. Além disso, a VEC permite que aplicações com requisitos rigorosos de latência sejam atendidas em níveis mais baixos na hierarquia de rede, mitigando

problemas como atrasos de transmissão, baixa confiabilidade de conexão e alto uso de largura de banda. No entanto, para que isso seja possível, a VEC precisa agregar os recursos computacionais dos veículos e disponibilizá-los na rede [Ju et al. 2023].

Neste contexto, o processo de agregação de recursos computacionais provenientes de veículos e infraestruturas de comunicação, tais como Estações Base (*Base Stations (BSs)*) da rede 5G, é conhecido como formação de nuvens veiculares (*Vehicular Clouds (VCs)*) [da Costa et al. 2023b]. Esse processo geralmente considera o padrão de mobilidade dos veículos para agrupá-los na cobertura de uma BS e, com isso, pode manter uma disponibilidade estável de recursos naquela região [Zhang et al. 2020]. A utilização desses recursos computacionais agregados, por sua vez, é referida como escalonamento de tarefas [Keshari et al. 2022]. Esse processo é realizado em tempo real e é projetado para otimizar o uso dos recursos disponíveis, garantindo ao mesmo tempo que as tarefas sejam concluídas de maneira rápida e eficiente [da Costa et al. 2022].

Um dos principais desafios do processo de escalonamento de tarefas em VCs é a natureza dinâmica da rede veicular, pois os veículos se movem e alteram suas posições, tornando desafiador prever a disponibilidade de recursos em uma localização específica [da Costa et al. 2023a]. Para lidar com esse desafio, estratégias avançadas são utilizadas para determinar a melhor distribuição de tarefas na rede com base em fatores como custos de comunicação e previsão de mobilidade [Ju et al. 2023]. No entanto, muito menos atenção tem sido dada às questões de equidade no uso de recursos computacionais das VCs no escalonamento de tarefas [Chen et al. 2022]. Dessa forma, pode-se tratar essa questão como um problema de balanceamento de carga entre os nós que processam tarefas (veículos e BSs), ou seja, distribuindo uniformemente a carga de trabalho entre os recursos computacionais disponíveis [Hejja et al. 2022].

O balanceamento de carga é realizado tanto quando uma tarefa chega no sistema quanto após ela já ter sido enfileirada [McClure et al. 2022]. Isso garante que as tarefas sejam alocadas de forma eficiente para evitar subutilização ou superutilização dos recursos [Kashani and Mahdipour 2023]. Ao equilibrar a carga, o sistema pode alcançar uma melhor utilização de recursos e tempos de resposta. Além disso, o balanceamento de carga pode ajudar a mitigar congestionamentos e lidar com padrões de demanda variáveis, resultando em um processo de escalonamento de tarefas mais eficiente e eficaz. Portanto, outro desafio importante das VCs é atender às demandas dos usuários mantendo um balanceamento de carga justo no uso dos recursos computacionais disponíveis [Chen et al. 2022], considerando ainda a mobilidade dos nós no processo de escalonamento.

Nesse contexto, este trabalho apresenta um mecanismo de escalonamento de tarefas que busca provisionar jus**T**iça na utilização de r**E**cursos co**M**putacionais em nuvens ve**I**culare**S**, chamado **TEMIS**. O TEMIS opera nos controladores de VEC e utiliza a eficiência de Pareto para agendar tarefas em diferentes VCs. O mecanismo divide o conjunto de tarefas em partes distintas para melhorar a eficiência do sistema com gerenciamento paralelo, obtendo k conjuntos de Pareto diferentes e sendo capaz de tomar k decisões simultaneamente, onde k é o número de sub-problemas em execução em cada controlador VEC. O TEMIS tem como objetivo minimizar o tempo de processamento dentro das VCs, reduzindo assim a utilização de recursos e, consequentemente, os custos monetários. Além disso, ele considera aspectos contextuais em seu processo decisório, como a mobilidade de recursos em cada VC e os requisitos das tarefas. A eficiência do TEMIS foi avaliada em comparação com outras abordagens da literatura, e os resultados indicam sua capacidade de escalonar um maior número de tarefas, minimizar custos monetários e reduzir a latência geral do sistema. Por fim, o TEMIS emprega um melhor balanceamento de carga no processo de escalonamento, resultando em uma maior

equidade no uso compartilhado de recursos das VCs.

Este trabalho está organizado da seguinte forma. A Seção 2 discute os principais trabalhos relacionados. A Seção 3 apresenta o modelo do sistema, a definição do problema e como o TEMIS funciona. A Seção 4 descreve o cenário de avaliação, a metodologia dos experimentos e discute os resultados obtidos. Por fim, a Seção 5 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Estudos têm criado diversas estratégias para otimizar o escalonamento de tarefas em sistemas VEC. Além disso, algumas dessas estratégias buscam equilibrar a carga de escalonamento nas VCs a fim de tornar a utilização de recursos mais justa em termos de proporção de uso pelos participantes. Por exemplo, [Hattab et al. 2019] introduziu um algoritmo para o escalonamento de tarefas em VCs com recursos variados, que utiliza principalmente na política *First-Come, First-Served (FCFS)*. O algoritmo classifica as tarefas por meio da razão entre o tempo de conclusão e o tempo de espera, identifica aquelas com a menor razão e utiliza Programação Linear para resolvê-las. O objetivo é minimizar o tempo de conclusão das tarefas, não considerando variação nos recursos da VC e focando exclusivamente em uma única VC. Nesse caso, o balanceamento de carga é trivial, pois há apenas uma VC para a tomada de decisão.

Alguns outros trabalhos consideram Métodos de Decisão Multi-critério (*Multi-Criteria Decision Making (MCDM)*) para a seleção da VC que irá processar a tarefa. Tais métodos de MCDM tendem a equilibrar as opções selecionadas para inserir um balanceamento de carga durante o processo decisório. Alinhado a essa abordagem, [Mishra et al. 2020] introduziu duas políticas de escalonamento de tarefas baseadas no método *Analytic Hierarchy Process (AHP)*, denominadas SECA e AHP-EV. Os esquemas propostos consideram a carga de rede e a carga computacional durante a tomada de decisão, visando minimizar o atraso de cada tarefa. Essas políticas diferem na atribuição de pesos a cada critério, mas ambas têm resultados semelhantes. AHP-EV utiliza pesos predefinidos para poder computacional e rede, enquanto o SECA determina dinamicamente os pesos desses critérios.

Outros trabalhos utilizam métodos de otimização multiobjetivo para o processo de seleção de tarefas que serão escalonadas. [da Costa et al. 2022] apresentou um mecanismo de escalonamento de tarefas para VCs baseado na eficiência de Pareto, chamado CARONTE. O CARONTE emprega informações de mobilidade veicular para estimar os recursos disponíveis em cada VC, permitindo, assim, uma tomada de decisão mais precisa. Além disso, utiliza a eficiência de Pareto para selecionar o subconjunto ótimo de tarefas visando à minimização conjunta de *deadline* (tempo de espera pelo resultado) e tempo de processamento, o que reduz os custos monetários associados ao uso de recursos. No entanto, o CARONTE prioriza as VCs com maior número de recursos disponíveis sem considerar possíveis sobrecargas que essa seleção contínua pode causar.

Da mesma forma, [Luo et al. 2022] avaliou as implicações de atraso e custo associadas ao escalonamento de tarefas em VCs. O estudo estabeleceu um *framework* de escalonamento que acomoda comunicação e computação nas VCs, considerando tarefas com requisitos variados. Consequentemente, um problema multiobjetivo foi projetado para minimizar tanto o atraso quanto o custo. Um algoritmo de escalonamento baseado em Otimização por Enxame de Partículas foi sugerido para derivar soluções Pareto-ótimas. No entanto, devido à natureza bioinspirada da abordagem, o tempo de convergência pode impactar o desempenho geral da solução. Além disso, os autores não consideraram a mobilidade veicular como um requisito no processo de escalonamento.

[Ribeiro Jr et al. 2023] apresentou uma abordagem meta-heurística que modela o escalonamento de tarefas em VCs como uma coalizão. Em teoria dos jogos, uma coalizão refere-se a um grupo de jogadores que concordam em cooperar, combinando suas estratégias para aprimorar sua recompensa conjunta. Assim, os autores propõem uma coalizão para maximizar a utilização de recursos, equilibrando dinamicamente a carga entre as VCs. Inicialmente, o mecanismo estabelece uma estratégia baseada no valor de *Shapley* para determinar a sequência na qual as tarefas são escalonadas. Posteriormente, o mecanismo utiliza uma fila para escalonar as tarefas com base nos valores calculados. No entanto, a solução não leva em consideração as implicações da mobilidade veicular e as restrições de prazo (*deadline*) das tarefas na tomada de decisão.

Com base na análise do estado da arte, é possível observar os trabalhos geralmente não consideram informações de mobilidade nos processos de decisão de escalonamento. Além disso, é essencial considerar o balanceamento de carga durante o processo de decisão de escalonamento de tarefas para aumentar a equidade no uso de recursos e evitar sobrecargas em regiões específicas da cidade, o que poderia aumentar os custos de manutenção associados.

3. Justiça na Utilização de Recursos Veiculares

Esta seção descreve o TEMIS, que leva em consideração a eficiência de Pareto e uma função de seleção probabilística para maximizar o número de tarefas escalonadas, balanceamento de carga e equidade no uso de recursos computacionais das VCs.

3.1. Modelo de Sistema

A Figura 1 apresenta a arquitetura de sistema composta por veículos, BSs, controladores VEC e um Servidor Remoto (RS) na Internet. 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 BS pode se comunicar com o RS por meio de enlace de fibra óptica.

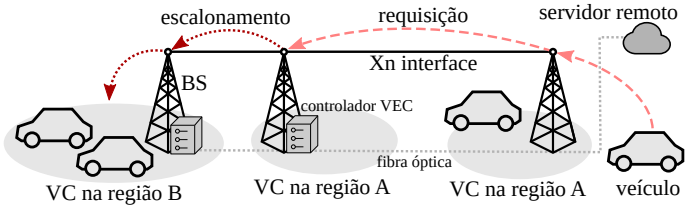


Figura 1. A arquitetura de sistema empregada pelo TEMIS, com seus principais componentes, tais como veículos, estações base, servidor remoto na nuvem de Internet, nuvens veiculares VCs) em diferentes regiões e controladores VEC.

Para aprimorar o gerenciamento das BSs, a cidade é dividida em R regiões, e cada região possui pelo menos uma BS. Além disso, consideramos um conjunto de $|R|$ controladores VEC, já que cada região é gerenciada por exatamente um controlador. Portanto, após a associação entre o veículo e a BS, a BS envia essa informação para o RS. O processo de associação considera a abordagem Max-SINR. As informações da BS são atualizadas a partir da mudança do número de veículos em sua cobertura.

No processo de agregação de recursos (formação de VCs), o controlador VEC precisa solicitar ao RS informações sobre as BSs e veículos para construir seu conhecimento regional [da Costa et al. 2023b] . Dessa forma, o sistema emprega um esquema de *Publish/Subscribe* para obter as informações relevantes sem introduzir tráfego indesejado

na rede. O conjunto de VCs é denotado por $v_j \in V = v_1, v_2, \ldots, v_m$, onde m é o número total de VCs. Considera-se que o número de VCs é o mesmo número de BSs, já que a posição das BSs define onde os VCs serão construídas. Em resumo, uma VC consiste em um conjunto de veículos e uma BSs capazes de compartilhar poder de processamento ω em Milhões de Instruções Por Segundo (*Milhões de Instruções Por Segundo (MIPS)*) e capacidade de armazenamento ϕ em Megabytes (MB). A quantidade total de poder de processamento Ω_j e capacidade de armazenamento Φ_j de cada VC v_j é a soma dos recursos compartilhados pelos veículos e BSs que compõem essas VCs.

Devido à variabilidade de recursos dos VCs ao longo do tempo, utilizaram-se dados precisos de mobilidade veicular para determinar o tempo de permanência de cada veículo dentro da cobertura das BSs. Mesmo que a previsão de mobilidade não seja o foco principal deste estudo, empregamos um método de previsão de mobilidade ótimo. Os dados futuros de mobilidade veicular são coletados a partir do conjunto de dados veiculares dentro de uma janela de tempo Z. Para simular erros de previsão, foi introduzido um ruído gaussiano branco a cada dado coletado [da Costa et al. 2022]. Finalmente, como agora os recursos disponíveis nos VCs podem ser estimados em cada unidade de tempo $z \in Z$, esses recursos podem ser representados por Ω_{jz} e Φ_{jz}.

3.2. Definição do Problema

No ambiente VEC, cada tarefa $t_l \in T = t_1, t_2, \ldots, t_n$ é representada por uma tupla id_l^t, s_l^t, w_l^t, D_l^t, onde id_l^t representa o número de identificação único, s_l^t denota o tamanho dos dados de entrada (em MB), w_l^t é o número de ciclos de *Central Processing Unit (CPU)* necessários para processar a tarefa e D_l^t é sua restrição de prazo (*deadline*). O tempo de processamento d_{lj}^t (tempo de execução de uma tarefa em uma configuração computacional específica) pode ser obtido com base no número necessário de ciclos de CPU w_l^t dividido pela frequência de ciclos de CPU do servidor Ω_j, sendo $d_{lj}^t = \frac{w_l^t}{\Omega_j}, \forall t_l \in v_j$.

Como os recursos computacionais das VCs são compartilhados entre diferentes tarefas, o Ω_j considerado para obter o tempo de processamento de uma tarefa específica deve ser atualizado de acordo com o grau de compartilhamento desse recurso dentro da VC, representado por Ψ_j. Assim, Ω_j é dividido pelo número de tarefas $|T_j'|$ que foram escalonadas na VC para produzir $\Psi_j = \frac{\Omega_j}{|T_j'|}, j \in V$.

Além disso, cada tarefa possui uma restrição de prazo D_l^t. Esse prazo representa um limite de tempo que a tarefa pode aguardar para ser processada. Se $d_{lj}^t \leq D_l^t$, a tarefa pode ser escalonada e executada na VC v_j. Além disso, quando uma tarefa é escalonada e começa a ser processada, há um custo associado a essa execução. O custo monetário de uma tarefa t_l pode ser modelado como:

$$\mathcal{C}_l = d_{lj}^t \times (w_l^t \times custoRecurso(t_l)). \tag{1}$$

Onde d_{lj}^t é o tempo de processamento t_l em $v_j \in V$ e w_l^t são seus ciclos de CPU necessários. $custoRecurso(t_l)$ indica o preço do recurso utilizado e é definido como 14.309 (se t_l usar os recursos da BS) ou 6.27 (se t_l usar os recursos dos veículos). Os preços são baseados em instâncias disponíveis na Amazon EC2[1] (Região Europeia, Frankfurt), como *g4ad* (BS) e *g3* (veículo).

Em resumo, quando uma tarefa chega ao sistema, ela é enfileirada e espera até ser escalonada. O controlador VEC deve selecionar a VC para processar essa tarefa. A

[1] `https://aws.amazon.com/pt/ec2/dedicated-hosts/pricing/`

decisão de seleção deve levar em consideração o poder de processamento da VC ao longo do tempo e os requisitos da tarefa. Assim, para considerar esses diferentes objetivos, formulou-se um problema de escalonamento de tarefas que busca principalmente maximizar o número de tarefas escalonadas, considerando restrições que impactam diretamente nos custos monetários da utilização dos recursos, confome Equação (2).

$$\text{maximizar} \sum_{l=1}^{n} t_l,\ l \in T, \tag{2}$$

$$\text{sujeito a } d_{lj}^t \leq D_l^t,\ l \in S',\ j \in V, \tag{3}$$

$$\sum_{l=1}^{n} s_l^t \leq \Phi_{jz},\ l \in S',\ j \in V,\ z \in Z, \tag{4}$$

$$\sum_{l=1}^{n} w_l^t \leq \Psi_{jz},\ l \in S',\ j \in V,\ z \in Z. \tag{5}$$

De maneira geral, a restrição (3) garante que o prazo da tarefa seja respeitado e ajuda a reduzir o custo monetário, evitando reescalonamentos no sistema. Além disso, as restrições (4) e (5) garantem que os limites de armazenamento e processamento das VCs durante os intervalos de tempo z necessários para o processamento sejam respeitados.

3.3. Principais Operações do TEMIS

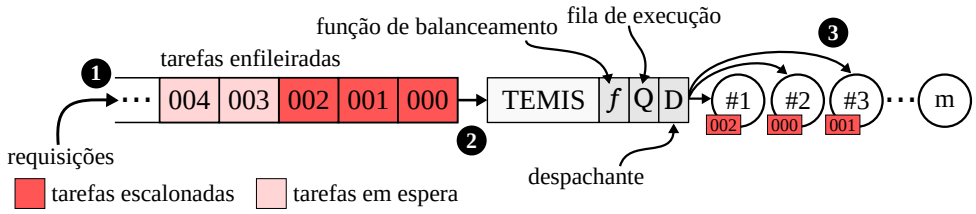


Figura 2. Pipeline de escalonamento de tarefas com uma função de balanceamento de carga integrada ao TEMIS.

A Figura 2 apresenta um esquema do processo de escalonamento de tarefas no ambiente VEC considerado neste trabalho. Em resumo, as tarefas são enfileiradas no controlador VEC assim que chegam ao sistema (Rótulo ❶, Figura 2). Nessa fase, a tarefa pode assumir dois estados principais, sendo *(i)* em espera ou *(ii)* escalonada. A tarefa assume o estado escalonada e retorna a um estado de espera se a VC não conseguir concluir seu processamento e o *deadline* restante da tarefa for $D_l^t > 0$. A partir desse momento, o *deadline* deve ser observado constantemente, pois a tarefa está aguardando a decisão do escalonador. Com base em seus critérios pré-definidos, o escalonador decide em qual VC a tarefa será processada. Esse escalonador pode empregar uma estratégia de balanceamento de carga para aumentar os níveis de equidade na utilização dos recursos computacionais da VEC. O despachante distribui as tarefas para as VCs (Rótulo ❷, Figura 2) de acordo com as decisões tomadas pelo escalonador. As tarefas em processamento são adicionadas à uma fila de controle de execução Q e só são removidas quando o processamento é finalizado. A proporção de distribuição de tarefas nas VCs é armazenada pelo controlador VEC para fins de balanceamento de carga em decisões futuras. Por fim, as VCs recebem as tarefas e iniciam o processamento (Rótulo ❸, Figura 2).

Para a tomada de decisão de escalonamento, o TEMIS busca o conjunto de Pareto usando uma abordagem de critérios duplos, a fim de minimizar simultaneamente os tempos de processamento e os *deadlines* das tarefas. Essa minimização conjunta é importante para reduzir o tempo que uma tarefa fica processando no sistema e garante que tarefas com requisitos restritos de *deadline* sejam priorizadas. Um vetor distinto é criado para cada critério (tempo de processamento e *deadline*), disposto em um plano bidimensional para encontrar o conjunto de Pareto. Nesse contexto, um conjunto de Pareto em 2D pode ser obtido com complexidade de tempo polinomial $\mathcal{O}(n \log n)$ [da Costa et al. 2022]. Além disso, como o objetivo é maximizar o número de tarefas escalonadas, o problema foi simplificado para uma instância do *Bin Covering Problem (BCP)*.

Além disso, o TEMIS divide a fila de tarefas e o conjunto de VCs em k partes para tornar o processo de escalonamento mais eficiente. Agora, o TEMIS calcula um conjunto de Pareto em cada parte k, tendo o universo de VCs disponíveis menor para a aplicação do BCP. Dessa forma, o TEMIS toma k decisões simultaneamente. Com a decisão de escalonamento tomada, é essencial realizar um balanceamento de carga entre os VCs disponíveis para aumentar a equidade no uso de recursos. Assim, uma função de seleção probabilística é utilizada para seleção de VCs no processo de escalonamento. Após a seleção de uma VC, sua probabilidade de seleção diminui. Assim, quando o TEMIS seleciona as VCs na próxima rodada, é realizada uma ordenação do vetor de probabilidades, e a VC com a maior probabilidade atual é selecionado. Com isso, após m rodadas, todos as VCs são selecionados pelo menos uma vez durante a execução do TEMIS, onde m é o número total de VCs.

Em resumo, cada VC tem uma probabilidade P_j^0, onde $j = 1, 2, \ldots, m$, de ser selecionada durante o processo de escalonamento. No entanto, a probabilidade de seleção é reduzida pela metade (0.5) se a VC for selecionada na rodada atual. Caso contrário, a probabilidade anterior é mantida, conforme Equação (6).

$$P_j^z = \begin{cases} 0.5 \times P_j^{z-1}, & \text{se o item } j \text{ foi selecionado} \\ P_j^{z-1}, & \text{caso contrário} \end{cases} \tag{6}$$

Para evitar o cenário em que, após um alto número de rodadas, a probabilidade de uma VC ser reduzida a zero e nunca mais ser selecionada, se faz necessário considerar uma probabilidade mínima não nula, como mostrado na Equação (7).

$$P_j^z = \begin{cases} 0.0001, & \text{se } P_j^z \leq 0 \\ P_j^z, & \text{caso contrário} \end{cases} \tag{7}$$

A Algoritmo 1 descreve as principais operações do TEMIS em um controlador VEC. O controlador recebe o conjunto de VCs V, o conjunto de tarefas T, e o número de k subproblemas que serão processados paralelamente, fornecendo o conjunto $\mathcal{S}$ de tarefas escalonadas como saída. Para cada intervalo de tempo de execução, a fila de espera de tarefas é verificada e caso não esteja vazia, as tarefas enfileiradas devem ser escalonadas (Linhas 1 e 2). Nessa etapa, a entidade DESPACHANTE segmenta T e V em k partes e inicia a execução do processo de ESCALONAMENTO em paralelo para cada k definido (Linhas 3 a 5). O resultado atual obtido, que pode ser tanto uma lista com as tarefas escalonadas ou um conjunto vazio (caso nenhuma tarefa possa ser escalonada na VC atual), é adicionado ao conjunto $\mathcal{S}_k$ (Linha 5). Por fim, o conjunto solução atual $\mathcal{S}_k$ é incorporado ao conjunto final $\mathcal{S}$, que automaticamente inclui as tarefas na fila de execução Q (Linhas 6 e 7). Uma entidade de monitoramento verifica a cada intervalo de tempo se a tarefa concluiu ou não seu processamento em Q.

Algoritmo 1: Pseudocódigo do TEMIS

Entrada: conjunto de tarefas T, conjunto de VCs V e número de subproblemas k
Saída: conjunto de tarefas escalonadas $\mathcal{S}$

```
1  para cada intervalo de tempo faça
2      se a fila de espera não estiver vazia então
           /* A entidade Despachante atua neste momento */
3          Divide o conjunto de tarefas T em k partes
4          Divide o conjunto de VCs V em k partes
5          S_k ← ESCALONAMENTO(T_k, V_k) para cada subproblema k      ▷ Em paralelo
6          S.add(S_k)
7          Q.put(S)
```

Adicionalmente, o Algoritmo 2 detalha a função de escalonamento. Inicialmente, o vetor de probabilidades dos VCs é organizado em ordem não crescente para selecionar o VC com a maior probabilidade de seleção (Linha 2). Dois vetores são criados com base em T, o primeiro R para o tempo de processamento estimado, e o segundo D para os prazos (Linhas 4 e 5). O TEMIS chama o procedimento CONJUNTOPARETO com configuração para minimização conjunta dos vetores R e D (Linha 6). O procedimento retorna um conjunto $\mathcal{P}$ contendo os id das tarefas no conjunto de Pareto. Depois disso, o TEMIS executa o BCP para selecionar o subconjunto $S' \in \mathcal{P}$ que melhor se ajusta a V atual (Linha 7). Além disso, o número total de recursos necessários para este conjunto retornado é calculado (Linha 8). Para cada tarefa no subconjunto S', é verificado se o VC terá recursos disponíveis até o seu prazo D_l^t (Linha 10). Se não tiver, essa tarefa é removida de S' (Linha 11). Se a VC possuir recursos, seu tempo de processamento atual é calculado (Linha 14). Se o tempo de processamento for maior que o *deadline*, a tarefa é removida de S' e será reagendada em uma próxima rodada. Caso contrário, o subconjunto S' é adicionado à lista de tarefas escalonadas S_k (Linha 18). Para cada v verificado e utilizado por ao menos uma tarefa, a probabilidade de seleção deve ser atualizada (Linha 20). Por fim, as tarefas escalonadas do segmento k e o vetor de probabilidade atualizado V_k^P são retornados para a função principal (Linha 21).

4. Avaliação de Desempenho

Esta seção descreve a metodologia e as métricas utilizadas para avaliar a eficiência do TEMIS em comparação com outras abordagens do estado da arte. Além disso, apresenta e discute os resultados obtidos.

4.1. Metodologia

Os experimentos foram conduzidos com o *Simulator of Urban Mobility (SUMO)*, versão 1.18.0. Os algoritmos foram implementados na linguagem Python, versão 3.10.12, e conectados ao SUMO via interface TraCI. Utilizou-se o cenário de mobilidade *Luxembourg SUMO Trace* (LuST)[2], que reproduz o tráfego veicular de 24 horas na cidade de Luxemburgo. Foi considerada 1 hora de mobilidade veicular, período entre 15:00 e 16:00, com até 1000 veículos simultâneos. O tempo de simulação foi de 900 segundos. As simulações foram executadas 33 vezes para se obter um intervalo de confiança de 95%.

Aplicações *Bag-of-Tasks (BoT)* foram consideradas, uma vez que podem ser executadas fora da ordem de chegada e não possuem dependência uma das outras. Os *deadlines* das tarefas foram variam em $0.5, 0.8, 1$ e 3 segundos para generalizar diferentes classes de aplicações. A formação das VCs foi a cada 5 segundos. Diferentes taxas de

[2] https://github.com/lcodeca/LuSTScenario/

Algoritmo 2: ESCALONAMENTO(T_k, V_k)

Entrada: conjunto de tarefas T_k, conjunto de VCs V_k
Saída: conjunto de tarefas escalonadas S_k no segmento k, probabilidades de seleção de V_k^P

```
1  S_k ← ∅
2  V_k ← ordem decrescente de probabilidade de seleção
3  para cada v ∈ V_k faça
4      R ← tempo de processamento das tarefas para v
5      D ← deadline das tarefas T_k
6      P ← ConjuntoPareto({R, D}, objetivo=[min, min])
7      S' ← BinCoveringProblem(P, v)
8      totalRecursos ← soma todos os recursos em S'
9      para cada t' ∈ S' faça
10         se totalRecursos < v até D_l^t então
11             S'.remove(t')
12             totalRecursos ← totalRecursos − t'
13         senão
14             d_lj^t ← como mostrado na Seção 3.2
15             se d_lj^t > D_l^t então
16                 S'.remove(t')
17             senão
18                 S_k ← S'
19     se v possui ao menos uma tarefa escalonada então
20         V_k^P ← Atualiza a probabilidade de v através da Equação (6)
21 retorna S_k, V_k^P
```

chegada de tarefas λ foram consideradas, sendo $1, 5$ e 10 tarefas/segundo e seguem uma distribuição de Poisson. Os raios de comunicação dos veículos e das BSs foram de 250 e 2000 metros, respectivamente.

Além disso, o tamanho atribuído às tarefas foi $s_l^t = [1, 10]$ (MB), e os ciclos de CPU necessários variaram em $w_l^t = [1, 40]$ Milhões de Instruções (MI). O número de CPU por veículo é 1, o que, sem perda de generalidade, representa 1 MIPS. A capacidade de armazenamento de cada veículo foi simplificada para 1 MB. Foram utilizadas 14 BSs, e cada uma pode compartilhar 20 MIPS de poder de processamento e 20 MB de capacidade de armazenamento. Foram considerados 5 controladores VEC, e cada um pode gerenciar até 4 BSs vizinhas. O valor de k foi de 5, o que significa que cada controlador VEC pode realizar até 5 operações simultaneamente. Informações reais de posicionamento de infraestruturas de telefonia celular foram consideradas nesta avaliação[3].

Para avaliação de desempenho, o TEMIS foi comparado com três abordagens da literatura, a saber: RANDOM, que combina uma política baseada no FCFS [Hattab et al. 2019] com uma política randomizada para selecionar VCs [Beraldi et al. 2020]; CARONTE [da Costa et al. 2022], que utiliza a eficiência de Pareto no processo de escalonamento de tarefas e seleciona VCs com mais recursos a cada rodada; e AHP-EV [Mishra et al. 2020], que utiliza a abordagem multi-critério AHP em seu processo de tomada de decisão para proporcionar um balanceio entre a nuvem e as tarefas observadas em cada rodada.

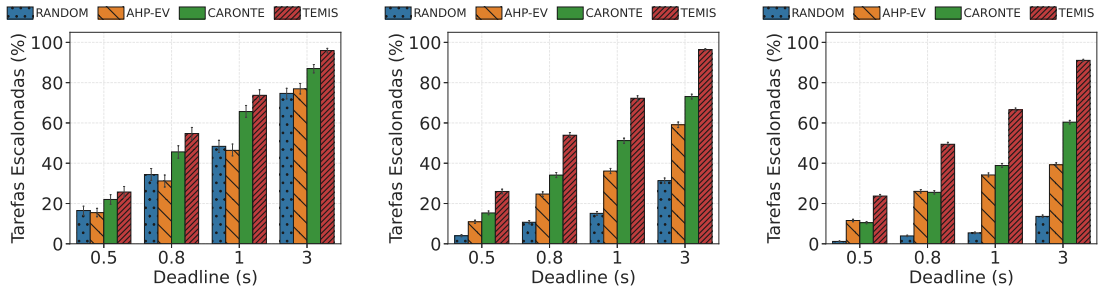
As métricas de comparação utilizadas foram: i) *Tarefas escalonadas*, que representa a porcentagem de tarefas concluídas com sucesso; ii) *Latência do sistema*, que se re-

[3] https://github.com/joahannes/eNodeB-LuST-Locations

fere ao tempo de processamento da tarefa na VC mais o tempo de espera na fila; iii) *Custo monetário*, que se refere ao preço de uso dos recursos computacionais, conforme discutido na Seção 3; e iv) *Justiça*, que representa o índice de justiça de Jain [Jain et al. 1984]. Esta métrica é amplamente utilizada para medir quão justa é a utilização de recursos em um sistema computacional.

4.2. Resultados

A Figura 3 apresenta os resultados de tarefas escalonadas. Assim, conforme pode ser visto, o desempenho de todas as abordagens melhora à medida que o *deadline* é estendido. Isso significa que os mecanismos têm mais tempo para a tomada de decisão e podem realizar mais tentativas sem sucesso até que esse *deadline* seja atingido. No entanto, o TEMIS consegue escalonar mais tarefas em todos os cenários observados. Além disso, o TEMIS pode escalonar mais de 90% das tarefas nos cenários mais complexos (*deadline* de 3 segundos e $\lambda = 10$). O uso de informações de mobilidade ajuda em decisões mais precisas, garantindo que a tarefa será concluída em seu processamento na VC selecionada. O CARONTE é o segundo mecanismo que consegue escalonar mais tarefas. No entanto, o fato de ele considerar apenas um conjunto de Pareto pode dificultar o processo decisório, pois conjuntos possivelmente melhores podem ser ignorados. O AHP-EV atinge níveis mais baixos de escalonamento em comparação com os dois mecanismos anteriores, devido à sua estratégia de decisão que seleciona apenas um par (tarefa, VC) a cada rodada. Essa abordagem visa integrar os critérios, que são os requisitos da tarefa, e selecionar a melhor tarefa para a VC atual. O RANDOM tem o pior desempenho em todos os cenários. Isso se deve à sua estratégia de decisão, que considera apenas o tamanho da tarefa empregado pelo FCFS, ignorando aspectos fundamentais como *deadline* e tempo de processamento.

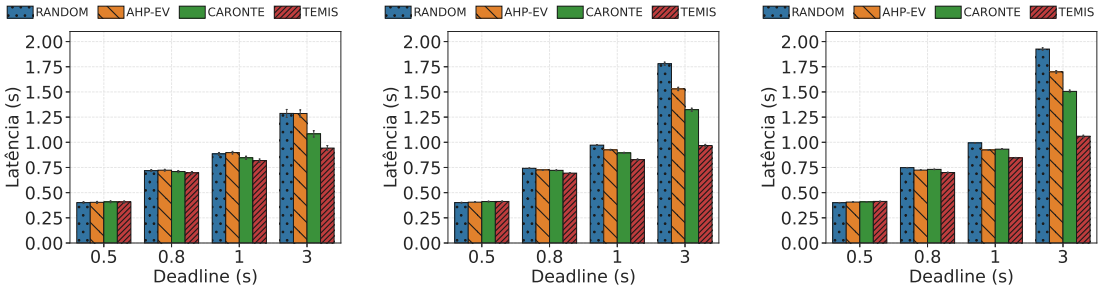


(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

Figura 3. Tarefas escalonadas com diferentes taxas de chegada de tarefas.

Figura 4 mostra os resultados referentes à latência do sistema. Nesta métrica, uma latência menor significa que as rodadas de escalonamento são mais eficientes. Em todos os cenários de avaliação, pode-se observar que o TEMIS reduz a latência do sistema. O melhor resultado é principalmente alcançado devido à seleção de VCs empregada pelo TEMIS, que minimiza conjuntamente tanto o tempo de processamento da tarefa quanto a seu *deadline*. Além disso, o uso de k tomadas de decisão ao mesmo tempo permite explorar um universo menor de tarefas, o que ajuda a reduzir a latência. Dessa forma, o TEMIS consegue lidar com um número significativo de tarefas em um tempo mais curto do que outras abordagens. A informação de mobilidade utilizada na tomada de decisão ajuda a estimar os recursos futuros, garantindo taxas de erro mais baixas em seu processo de escalonamento. O RANDOM tem uma latência mais alta em todas as avaliações devido à sua tomada de decisão que prioriza apenas o tamanho da tarefa. O CARONTE consegue reduzir a latência priorizando sempre a VC com maior capacidade, o que torna

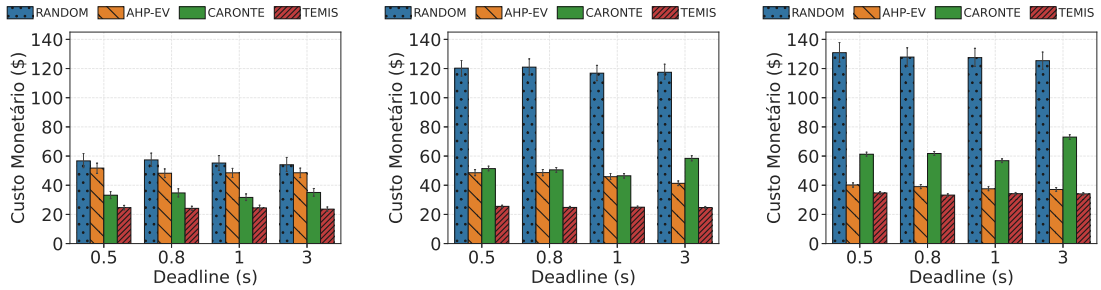
o processamento mais rápido. Já o AHP-EV possui maiores níveis de latência por conta da tomada de decisão individual para cada tarefa enfileirada. Em resumo, TEMIS melhorou o gerenciamento de recursos incorporando aspectos contextuais das tarefas e VCs em seu processo de tomada de decisão.



(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

Figura 4. Latência do sistema com diferentes taxas de chegada de tarefas.

Figura 5 apresenta o custo do uso dos recursos. Como os preços dos recursos diferem, conforme discutido na Seção 3.2, as abordagens priorizam os recursos das VCs para minimizar o custo monetário final. Pode-se observar que o TEMIS minimiza o custo monetário em todas as avaliações realizadas. O melhor desempenho do TEMIS se deve à seleção de tarefas considerando os recursos futuros disponíveis nas VCs. A eficiência de Pareto permite escalonar o conjunto ótimo de tarefas na mesma VC com o mínimo de tempo de processamento e *deadlines*. Além disso, separar as tarefas enfileiradas em k partes torna as verificações mais eficientes, e o TEMIS comete menos erros durante a decisão de escalonamento por conta dessa diminuição do universo de observação. O RANDOM apresenta pior desempenho nessa métrica porque a seleção de VCs é totalmente aleatória, o que não garante que a tarefa seja concluída na VC selecionada.

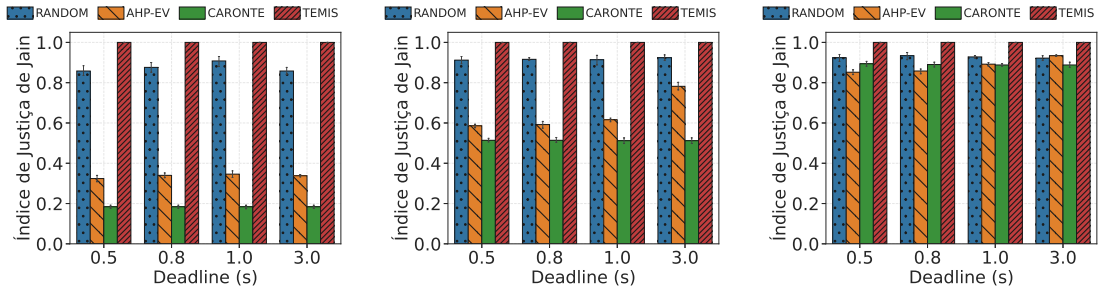


(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

Figura 5. Custo monetário com diferentes taxas de chegada de tarefas.

Finalmente, Figura 6 apresenta o índice de justiça de Jain obtido pelas abordagens ao selecionar VCs para o processo de escalonamento. O TEMIS obtém o melhor índice de justiça em todos os cenários considerados. A função de seleção de VCs baseada em probabilidade permite que todas as VCs sejam escolhidas durante o processo de escalonamento, aumentando assim a capacidade da rede de atender às demandas dos usuários. A abordagem RANDOM obteve o segundo melhor desempenho nessa métrica, pois seleciona aleatoriamente uma VC a cada rodada, aumentando significativamente a chance de todas as VCs serem escolhidas ao menos uma vez. No entanto, é crucial visualizar essas métricas de forma holística. Não é suficiente para uma abordagem de escalonamento ter

um alto índice de justiça na utilização dos recursos se ela não puder satisfazer nem mesmo 15% das demandas dos usuários em cenários mais complexos, como o RANDOM na Figura 3(c). O AHP-EV possui menores níveis de justiça também por selecionar a VCs de maior capacidade para uma tarefa por vez. O CARONTE tem o pior índice de equidade, pois prioriza VCs com mais recursos em cada rodada de escalonamento sem se importar em como os recursos serão utilizados ou em um balanceamento dos recursos disponíveis.



(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

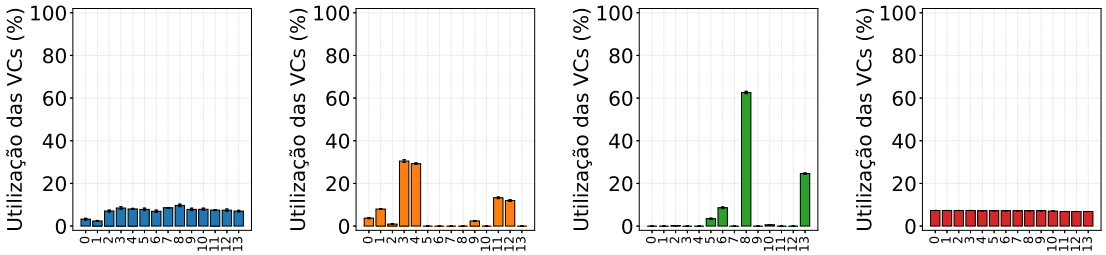
Figura 6. Índice de Justiça de Jain com diferentes taxas de chegada de tarefas.

Adicionalmente, é importante visualizar como uma solução utiliza os recursos das VCs durante o processo de escalonamento em diferentes cenários de variabilidade de recursos. Sendo assim, as Figura 7, 8 e 9 apresentam o nível balanceamento na utilização das VCs durante o processo de escalonamento para cada uma das abordagens consideradas. Nessa avaliação, o *deadline* foi fixado em 0.5 segundos (cenário mais complexo) e variou-se a taxa de chegada λ das tarefas no sistema. No cenário de menor complexidade, quando $\lambda = 1$, as abordagens conseguem ter um espaço de exploração menor por conta do menor número de tarefas enfileiradas no sistema. No entanto, o *deadline* da tarefa também deve ser levado em consideração na tomada de decisão, o que a torna desafiadora quando considera-se tarefas com requisitos restritos de latência.

A Figura 7 apresenta a distribuição da seleção de VCs de cada abordagem durante o escalonamento em tempo real (*deadline* igual a 0.5 segundos). O TEMIS é a abordagem que melhor balanceia a escolha das VCs durante o escalonamento, justamente por conta de sua função de seleção baseada na probabilidade de escolha. O RANDOM também consegue empregar um balanceamento adequado no uso dos recursos, fazendo com que as VCs participem uniformemente do processo de escalonamento por conta da sua seleção aleatória. Já o CARONTE e o AHP-EV possuem baixos níveis de balanceamento, em razão do processo de seleção das VCs, onde o CARONTE seleciona sempre a VC de maior capacidade na rodada e o AHP-EV seleciona a VC de maior capacidade para uma única tarefa a fim de criar um pareamento ótimo entre tarefa e VC.

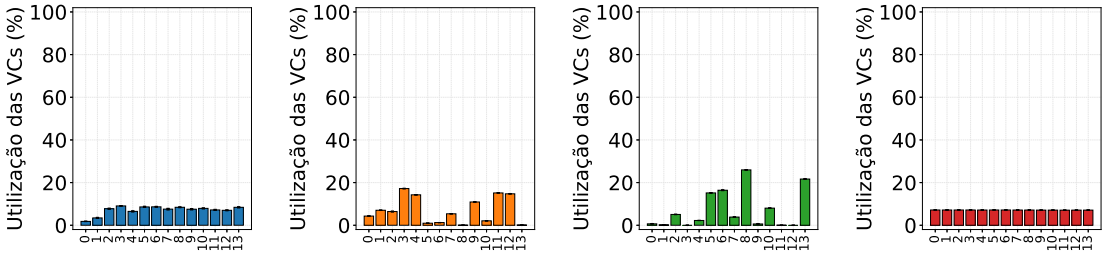
Por outro lado, um comportamento interessante é observado quando a taxa de chegada de tarefas no sistema aumenta. As abordagens tendem a manter um maior nível de balanceamento no uso dos recursos quando o número de tarefas no sistema aumenta devido ao aumento de tentativas de escalonamento que acontecem nesse cenário mais desafiador. Ou seja, o cenário fica mais complexo e mais VCs precisam ser observadas nas rodadas de escalonamento. Dessa forma, esse nível de balanceamento e utilização dos recursos é totalmente ilusório se observado de forma individual, havendo então a necessidade da análise conjunta com as métricas de escalonamento discutidas anteriormente. Nas Figuras 8 e 9 pode-se observar que o AHP-EV passa a considerar mais nuvens durante o processo de escalonamento, bem como ocorre com o CARONTE, pois o universo de exploração precisa ser expandido com mais tentativas de escalonamento. Por fim, o

TEMIS se mostra superior em todas as avaliações, pois mantém altos níveis de balanceamento tanto nos cenários menos complexos quanto nos mais complexos.



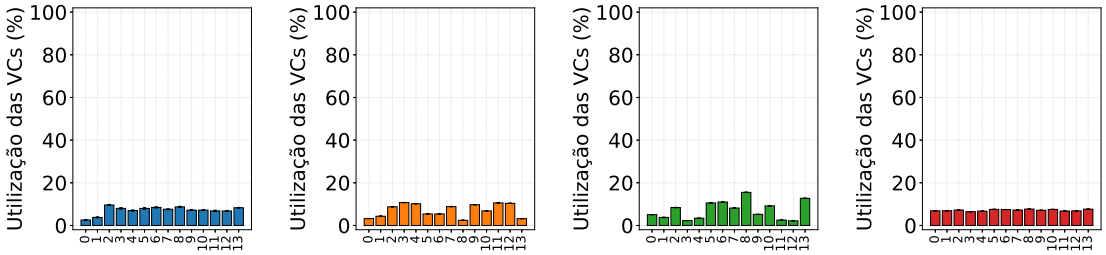
(a) RANDOM (b) AHP-EV (c) CARONTE (d) TEMIS

Figura 7. Nível de balanceamento com $\lambda = 1$.



(a) RANDOM (b) AHP-EV (c) CARONTE (d) TEMIS

Figura 8. Nível de balanceamento com $\lambda = 5$.



(a) RANDOM (b) AHP-EV (c) CARONTE (d) TEMIS

Figura 9. Nível de balanceamento com $\lambda = 10$.

5. Conclusão

Neste trabalho, foi proposto um mecanismo de escalonamento de tarefas para ambientes VEC, utilizando o princípio da eficiência de Pareto para selecionar o melhor conjunto de tarefas a ser escalonado nas VCs disponíveis. Combinou-se a eficiência de Pareto com o BCP para encontrar a associação mais adequada entre as tarefas e os recursos dos VCs. Além disso, foram considerada informações de mobilidade veicular para estimar os recursos em cada VC e tomar decisões com base nessa estimativa de recursos ao longo do tempo. A abordagem proposta, nomeada TEMIS, também garante altos níveis de justiça na utilização de recursos veiculares, aplicando uma função de probabilidade para o balanceamento de carga nos VCs selecionados. Comparado às soluções do estado-da-arte, o TEMIS possui um nível mais elevado de justiça na distribuição das cargas de trabalho e utilização de recursos, e pode escalonar um maior número de tarefas, ao mesmo tempo que minimiza custos monetários e a latência do sistema.

Agradecimentos

Este projeto foi apoiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) por meio do processo de Nº #2018/16703-4. Também recebeu apoio do Ministério da Ciência, Tecnologia e Inovações, financiado pela Lei nº 8.248 de 23 de outubro de 1991. Foi desenvolvido sob a égide do Programa Prioritário de Informática (PPI-Softex), gerido pela Softex, e foca em agentes inteligentes para plataformas móveis utilizando tecnologia de Arquitetura Cognitiva (processo nº 01245.013778/2020-21).

Referências

Beraldi, R., Canali, C., Lancellotti, R., and Proietti Mattia, G. (2020). Randomized Load Balancing under Loosely Correlated State Information in Fog Computing. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 123–127. ACM.

Chen, C., Li, H., Li, H., Fu, R., Liu, Y., and Wan, S. (2022). Efficiency and Fairness Oriented Dynamic Task Offloading in Internet of Vehicles. *IEEE Transactions on Green Communications and Networking*, 6(3):1481–1493.

da Costa, J. B., de Souza, A. M., Meneguette, R. I., Cerqueira, E., Rosário, D., and Villas, L. A. (2022). Escalonamento de tarefas ciente de contexto para computaçao de borda veicular. In *Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 15–28. SBC.

da Costa, J. B., Lobato, W., de Souza, A. M., Cerqueira, E., Rosário, D., Sommer, C., and Villas, L. A. (2023a). Mobility-aware vehicular cloud formation mechanism for vehicular edge computing environments. *Ad Hoc Networks*, 151:103300.

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

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

Hejja, K., Berri, S., and Labiod, H. (2022). Network slicing with load-balancing for task offloading in vehicular edge computing. *Vehicular Communications*, 34:100419.

Jain, R. K., Chiu, D.-M. W., Hawe, W. R., et al. (1984). A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 21.

Ju, Y., Chen, Y., Cao, Z., Liu, L., Pei, Q., Xiao, M., Ota, K., Dong, M., and Leung, V. C. M. (2023). Joint Secure Offloading and Resource Allocation for Vehicular Edge Computing Network: A Multi-Agent Deep Reinforcement Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):5555–5569.

Kashani, M. H. and Mahdipour, E. (2023). Load Balancing Algorithms in Fog Computing: A Systematic Review. *IEEE Transactions on Services Computing*, 16(2):1505–1521.

Keshari, N., Singh, D., and Maurya, A. K. (2022). A survey on Vehicular Fog Computing: Current state-of-the-art and future directions. *Vehicular Communications*, 38:100512.

Luo, Q., Li, C., Luan, T. H., and Shi, W. (2022). Minimizing the Delay and Cost of Computation Offloading for Vehicular Edge Computing. *IEEE Transactions on Services Computing*, 15(5):2897–2909.

McClure, S., Ousterhout, A., Shenker, S., and Ratnasamy, S. (2022). Efficient scheduling policies for Microsecond-Scale tasks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1–18.

Mishra, S., Sahoo, M. N., Bakshi, S., and Rodrigues, J. J. P. C. (2020). Dynamic Resource Allocation in Fog-Cloud Hybrid Systems Using Multicriteria AHP Techniques. *IEEE Internet of Things Journal*, 7(9):8993–9000.

Ribeiro Jr, A., da Costa, J. B., Rocha Filho, G. P., Villas, L. A., Guidoni, D. L., Sampaio, S., and Meneguette, R. I. (2023). Harmonic: Shapley values in market games for resource allocation in vehicular clouds. *Ad Hoc Networks*, page 103224.

Xue, J., Wang, Q., Zhang, H., An, N., and An, C. (2023). Idle-parked vehicles assisted collaborative resource allocation in VEC based on Stackelberg game. *Ad Hoc Networks*, 142:103069.

Zhang, J., Guo, H., Liu, J., and Zhang, Y. (2020). Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution. *IEEE Transactions on Vehicular Technology*, 69(2):2092–2104.