CANOPY: Alocação Orientada por Predição Energética de Aplicações no Contínuo Computacional

Giselly A. Reis[1], Gustavo B. Figueiredo[1], Bruno Santos[1], Cassio Prazeres[1], Luis Veiga[2], Maycon L. M. Peixoto[1]

[1]Instituto de Computação, Universidade Federal da Bahia (UFBA) – Brazil

{gisellyar, gustavobf, bruno.ps, prazeres, maycon.leone}@ufba.br

[2]INESC-ID Lisboa/Instituto Superior Técnico, Universidade de Lisboa

{luis.veiga@tecnico.ulisboa.pt}

Resumo. *A alocação de módulos energeticamente eficiente tornou-se um desafio crítico no contínuo computacional, onde aplicações modulares de IoT precisam operar sob recursos heterogêneos e cargas de trabalho dinâmicas. Embora abordagens baseadas em Aprendizado por Reforço tenham sido amplamente exploradas, elas frequentemente exigem extensos períodos de treinamento e, em geral, tratam o consumo de energia como uma métrica secundária ou reativa. Este artigo propõe o CANOPY, uma estratégia de* Alocação Ótima de Aplicações Consciente do Contínuo *guiada por predição de energia. O CANOPY emprega um modelo de aprendizado supervisionado baseado em um Regressor de Floresta Aleatória para estimar proativamente o consumo de energia da alocação de módulos de aplicações em dispositivos candidatos. Essas predições são integradas diretamente à lógica de alocação, possibilitando decisões energeticamente eficientes ao longo do contínuo computacional. Os resultados experimentais mostram que o CANOPY supera consistentemente estratégias de alocação do estado da arte, alcançando reduções no consumo de energia de até 12,42% em cenários com saturação de recursos.*

Abstract. *Energy-efficient module placement has become a critical challenge in the computing continuum, where modular IoT applications must operate under heterogeneous resources and dynamic workloads. Although Reinforcement Learning-based approaches have been widely explored, they often require extensive training periods and typically treat energy consumption as a secondary or reactive metric. This paper proposes CANOPY, a* Continuum-Aware Optimal Application Placement *strategy guided by energy prediction. CANOPY employs a supervised learning model based on a Random Forest Regressor to proactively estimate the energy consumption of application module placements on candidate devices. These predictions are directly integrated into the allocation logic, enabling energy-efficient decisions across the computing continuum. Experimental results show that CANOPY consistently outperforms state-of-the-art placement strategies, achieving energy consumption reductions of up to 12.42% in resource-saturated scenarios.*

1. Introdução

A ampla adoção de aplicações baseadas na Internet das Coisas (do Inglês Internet of Things (IoT)) e a demanda por processamento de baixa latência têm impulsionado a

consolidação do contínuo computacional, integrando camadas de computação em Nuvem, Névoa e Borda. Esse paradigma hierárquico deslocou os recursos computacionais para mais perto das fontes de dados, mitigando limitações centrais dos modelos de nuvem centralizada, como alta latência, congestionamento de enlaces e restrições de escalabilidade [Atwal et al. 2025, Reis et al. 2026]. À medida que as aplicações de IoT se tornaram cada vez mais modulares, heterogêneas e intensivas em recursos, a alocação eficiente de módulos ao longo do contínuo passou a ser um desafio central para alcançar desempenho, eficiência energética e sustentabilidade de longo prazo.

Para lidar com a natureza dinâmica desses ambientes, uma parcela significativa da literatura tem explorado técnicas de Aprendizado por Reforço (Reinforcement Learning (RL)) e Aprendizado por Reforço Profundo (Deep Reinforcement Learning (DRL)) para gerenciamento de recursos e alocação de serviços. Estudos anteriores demonstram o potencial de abordagens baseadas em RL para a tomada de decisão inteligente em infraestruturas de névoa e borda [Tran-Dang et al. 2022], enquanto estratégias de DRL têm sido propostas para otimizar a alocação de serviços e reduzir a latência fim a fim sob cargas de trabalho variáveis [Zhang et al. 2025]. Apesar de sua eficácia, essas abordagens frequentemente dependem de extensas fases de treinamento e podem apresentar uma generalização limitada quando expostas a condições de implantação ou perfis de carga de trabalho não observados anteriormente.

Investigações recentes têm destacado diversas limitações persistentes nas estratégias atuais de alocação baseadas em aprendizado. Wang et al. [Wang et al. 2024] relatam que muitos escalonadores baseados em aprendizado profundo têm dificuldade em equilibrar acurácia preditiva e responsividade em tempo real, particularmente em cenários altamente dinâmicos ou com restrições severas de recursos. De forma semelhante, Atwal e Singh [Atwal et al. 2025] enfatizam a necessidade de mecanismos de alocação mais robustos e escaláveis, capazes de sustentar demandas diversificadas de aplicações ao longo do contínuo computacional. Além disso, em uma parte substancial das soluções existentes, o consumo de energia é tratado como um objetivo secundário, em vez de um alvo de otimização explícito e previsível.

Motivado por esses desafios, este artigo apresenta uma estratégia de alocação orientada por métodos de Aprendizado de Máquina que incorporam explicitamente a predição do consumo de energia no processo de tomada de decisão. Diferentemente das abordagens convencionais baseadas em RL, introduzimos o *Continuum-aware ApplicatioN Optimal Placement guided by energY prediction (CANOPY)*, que emprega um modelo de aprendizado supervisionado para estimar o consumo de energia associado à alocação de módulos de aplicações em diferentes dispositivos. O CANOPY captura a relação entre características da aplicação, intensidade da carga de trabalho e capacidades dos dispositivos, possibilitando decisões de alocação energeticamente conscientes já na fase inicial de alocação.

A estratégia proposta é avaliada como um mecanismo de alocação sensível ao contexto, operando sob a disponibilidade heterogênea de recursos em um contínuo computacional multicamadas. Sua eficácia e robustez são analisadas por meio de uma extensa avaliação experimental, utilizando uma classe representativa de aplicações de processamento de imagens computacionalmente intensivo. As principais contribuições deste trabalho são resumidas a seguir:

- Um modelo preditivo de consumo de energia baseado em um Regressor de Floresta Aleatória, tratando a energia como uma variável previsível e de primeira classe em ambientes de contínuo computacional.
- O CANOPY, uma estratégia de alocação de módulos consciente do contínuo que integra a predição de energia diretamente à lógica de alocação nas camadas de nuvem, névoa e borda.
- Uma extensa avaliação experimental sob cargas de trabalho com saturação de recursos, demonstrando economias consistentes de energia em relação a estratégias de alocação do estado da arte.

2. Trabalhos Relacionados

Uma parcela significativa da literatura recente concentra-se em Aprendizado por Reforço (RL) e Aprendizado por Reforço Profundo (DRL) para lidar com a natureza dinâmica dos ambientes de borda. Tran-Dang et al. [Tran-Dang et al. 2022] apresentam uma revisão abrangente sobre como o RL pode mitigar a latência por meio do gerenciamento de filas de tarefas em nós de névoa. De forma semelhante, Zhang et al. [Zhang et al. 2025] empregam variantes de DRL, especificamente PPO, para otimizar a alocação de serviços e reduzir o consumo de energia. Embora esses trabalhos demonstrem o potencial do RL para a tomada de decisões sequenciais, eles frequentemente dependem de extensas fases de treinamento por tentativa e erro.

A complexidade das aplicações modernas frequentemente exige que elas sejam tratadas como um conjunto de módulos, em vez de tarefas monolíticas. Goudarzi et al. [Goudarzi et al. 2023] e Wenle Bai [Bai and Qian 2021] abordam a alocação de aplicações modulares utilizando modelos de Grafos Acíclicos Direcionados para satisfazer dependências entre tarefas em diferentes camadas computacionais. Este trabalho se baseia nessa perspectiva modular, acrescentando uma camada adicional de robustez ao avaliar a estratégia sob intensidades variáveis de carga de trabalho.

A eficiência energética permanece como um gargalo crítico em sistemas distribuídos modernos. Wang et al. [Wang et al. 2024] identificam limitações persistentes de generalização e desempenho em tempo real, destacando a necessidade de soluções mais robustas e energeticamente conscientes. De forma semelhante, Atwal e Singh [Atwal et al. 2025] enfatizam a importância de aprimorar os mecanismos de alocação de recursos para garantir ambientes de contínuo computacional escaláveis e resilientes. Uma comparação estruturada da literatura é apresentada na Tabela 1.

Embora os métodos vistos na Tabela 1 sejam eficazes na descoberta de políticas por meio da interação com o ambiente, eles frequentemente requerem longos períodos de treinamento e carecem da interpretabilidade imediata oferecida por modelos de aprendizado supervisionado. Além disso, diversos estudos existentes concentram-se principalmente na otimização de latência e tráfego de rede [Ren et al. 2025], tratando o consumo de energia, em muitos casos, como uma restrição secundária, e não como um alvo preditivo primário. Este trabalho enfrenta essas limitações ao aproximar o gerenciamento arquitetural de alto nível da consciência energética dos dispositivos, oferecendo uma estrutura robusta para alocação inteligente e sensível ao contexto no contínuo computacional em evolução.

Tabela 1. Trabalhos relacionados

Autor(es)	Estratégia	Modular	Latência	Rede	Energia	Multinível
[Bai and Qian 2021]	DRL (A2C)	✓	✓	✗	✗	✗
[Zhang et al. 2025]	DRL (PPO)	✗	✓	✗	✓	✗
[Ansere et al. 2024]	Qe-DRL	✗	✓	✗	✓	✗
[Li et al. 2025]	Computação Colaborativa	✗	✓	✗	✓	✓
[Goudarzi et al. 2023]	DRL (IMPALA)	✓	✓	✓	✓	✓
[Yuan et al. 2022]	DL (LSTM)	✗	✓	✗	✗	✗
[Karthiga et al. 2025]	ML	✗	✓	✓	✗	✗
[Huang and Xie 2024]	RL Federado	✗	✓	✗	✗	✗
[Ren et al. 2025]	DL	✗	✓	✗	✓	✗
CANOPY	**ML (Floresta Aleatória)**	✓	✓	✓	✓	✓

3. CANOPY

A alocação de módulos no contínuo computacional exige o balanceamento entre latência, disponibilidade de recursos e eficiência energética em camadas heterogêneas. Enquanto dispositivos de borda oferecem baixa latência, são limitados em capacidade e eficiência energética, ao passo que nós de névoa e nuvem apresentam maior poder computacional e melhor eficiência por unidade de processamento, ao custo de maior latência. Nesse contexto, o **CANOPY** considera esse compromisso ao tratar o consumo de energia como uma *variável previsível*, integrando um modelo de predição baseado em regressão diretamente à lógica de alocação, permitindo decisões energeticamente eficientes e viáveis ao longo do contínuo computacional.

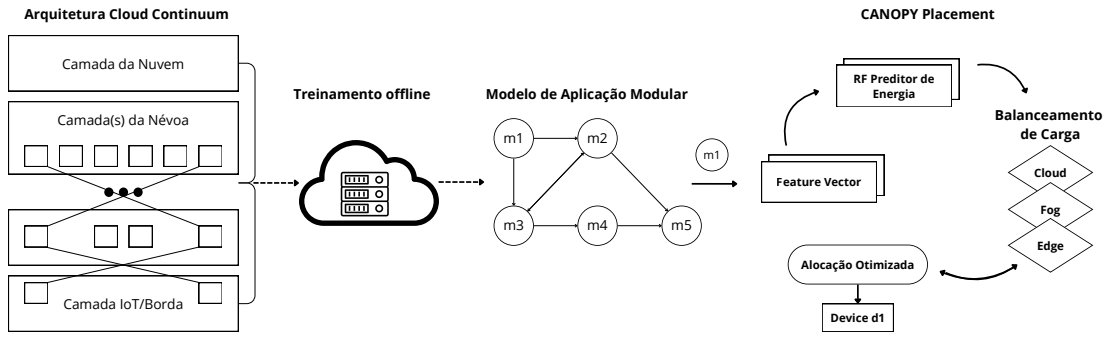


Figura 1. Visão geral da arquitetura do CANOPY, ilustrando a alocação de módulos orientada por energia ao longo do contínuo computacional, utilizando modelagem preditiva e mecanismo de *fallback* sensível à capacidade.

3.1. Formalização Matemática

Para caracterizar formalmente o problema de alocação de aplicações em um ambiente dinâmico no Contínuo Computacional, assume-se: M é o conjunto de módulos da aplicação e cada módulo $m \in M$ está associado a uma carga esperada de processamento em CPU, denotada por length_m, que captura a demanda computacional imposta pelo módulo. Essa carga é obtida a partir de parâmetros específicos da aplicação, a saber, as taxas de chegada de requisições e os requisitos de processamento associados a cada fluxo de dados, sendo calculada como:

$$\text{length}_m = \sum_{e \in \mathcal{E}_{in}(m)} \lambda_e \times L_e^{\text{CPU}}, \tag{1}$$

onde $\mathcal{E}_{in}(m)$ representa o conjunto de arestas de aplicação de entrada direcionadas ao módulo m, λ_e denota a taxa de chegada de requisições (requisições por segundo) na aresta e, e L_e^{CPU} corresponde ao número de ciclos de CPU necessários para processar uma única tupla.

Seja D o conjunto de dispositivos disponíveis no contínuo névoa–nuvem. Cada dispositivo $d \in D$ é caracterizado por sua capacidade de processamento CPU_d (em MIPS), consumo máximo de potência $P_{\text{max},d}$ e latência de uplink latency_d, que representam as capacidades estáticas da infraestrutura e são conhecidas antes da alocação. Além disso, seja C_d a carga máxima de CPU expressa em milhões de instruções por segundo (MIPS), que o dispositivo d pode sustentar. A alocação deve satisfazer a seguinte restrição de capacidade para todo $d \in D$:

$$\sum_{m \in M} w_{m,d} \, \text{length}_m \leq C_d, \quad \forall d \in D, \tag{2}$$

onde $w_{m,d}$ é uma variável de decisão binária que indica se o módulo m é alocado ao dispositivo d.

3.2. Arquitetura do CANOPY e Lógica de Alocação

A arquitetura e a lógica operacional do CANOPY baseiam-se em uma estratégia de alocação de módulos orientada por IA, projetada para minimizar proativamente o consumo de energia em ambientes de contínuo computacional. A ideia central do CANOPY é deslocar a eficiência energética de uma mera restrição operacional para um objetivo de otimização preditiva. Para isso, a estratégia integra diretamente um regressor de Floresta Aleatória treinado ao processo de alocação, permitindo que o sistema estime o impacto energético de alocações candidatas *antes* de sua execução. Essa capacidade preditiva permite que as decisões de alocação sejam orientadas por comportamento energético empírico, em vez de heurísticas estáticas.

O fluxo geral de execução do CANOPY é ilustrado no Algoritmo 1. O algoritmo inicia coletando o conjunto de aplicações $\mathcal{A}$ a serem implantadas e ordenando-as de acordo com um critério predefinido, como o tempo de chegada (linhas 1–2). Essa ordenação assegura comportamento determinístico e reflete cenários realistas de implantação em ambientes dinâmicos de névoa–nuvem. Para cada aplicação $a \in \mathcal{A}$, o conjunto correspondente de módulos M_a é extraído (linha 4), reforçando o modelo de aplicações modulares adotado neste trabalho.

Antes que qualquer decisão de alocação seja tomada, o CANOPY estima explicitamente a demanda computacional de cada módulo. Conforme mostrado no Algoritmo 1 (linhas 5–7), a carga esperada de processamento em CPU length_m é calculada para cada módulo $m \in M_a$ utilizando a Eq. (1). Essa etapa captura o efeito combinado das taxas de chegada de requisições e dos requisitos de processamento por tupla, fornecendo uma caracterização realista da intensidade da carga de trabalho. Uma vez que todas as car-

Algorithm 1 Inicialização da Simulação

1: $\mathcal{A} \leftarrow$ Conjunto de aplicações a serem alocadas
2: **sort**($\mathcal{A}$) de acordo com uma ordenação predefinida (e.g., tempo de chegada)
3: **for** cada aplicação $a \in \mathcal{A}$ **do**
4: $M_a \leftarrow$ Módulos da aplicação a
5: **for** cada módulo $m \in M_a$ **do**
6: length$_m \leftarrow$ EstimateExpectedCPULoad(m) ▷ Estimativa da demanda de CPU com base na Eq. (1)
7: **end for**
8: InitialModuleAllocation(a, M_a) (Algoritmo 2)
9: **end for**

gas dos módulos tenham sido estimadas, o algoritmo invoca o procedimento principal de alocação, detalhado no Algoritmo 2 (linha 8).

Algorithm 2 Alocação Inicial de Módulos

1: $M \leftarrow$ Módulos da aplicação a
2: **while** existirem módulos a serem alocados em M **do**
3: $D \leftarrow$ Conjunto de dispositivos disponíveis de névoa–nuvem
4: $m \leftarrow$ Selecionar o próximo módulo para alocação a partir de M
5: $minEnergy \leftarrow \infty$
6: $bestDevice \leftarrow null$
7: **for** cada dispositivo d em D **do**
8: $F_{m,d} \leftarrow \{C_d, \text{length}_m, L^{\text{CPU}}_{e,m}, \lambda_{e,m}, \text{Instances}_m\}$ ▷ Construção do vetor de características
9: $predictedEnergy \leftarrow RF\_Model.predict(F_{m,d})$ ▷ Execução da inferência
10: **if** $predictedEnergy < minEnergy$ **then**
11: $minEnergy \leftarrow predictedEnergy$
12: $bestDevice \leftarrow d$
13: **end if**
14: **end for**
15: $placeModuleOnDevice(m, bestDevice)$ (Algoritmo 3)
16: Marcar o módulo m como alocado
17: Remover m de M
18: **end while**

O Algoritmo 2 descreve o mecanismo central de tomada de decisão do CANOPY. Para cada módulo aguardando alocação, o algoritmo avalia todos os dispositivos disponíveis no contínuo névoa–nuvem. O processo de alocação segue uma estratégia gulosa, inicializada definindo o consumo mínimo de energia previsto como infinito (linhas 4–6), assegurando que os dispositivos candidatos sejam comparados exclusivamente com base em sua pegada energética estimada.

Para cada dispositivo candidato $d \in D$, o CANOPY constrói um vetor de características $F_{m,d}$, conforme mostrado no Algoritmo 2 (linha 8). Esse vetor agrega informações estáticas e dinâmicas, incluindo a capacidade do dispositivo C_d, a carga estimada do módulo length$_m$, o custo de CPU por tupla $L^{\text{CPU}}_{e,m}$, a taxa de chegada de requisições $\lambda_{e,m}$ e o número de instâncias ativas Instances$_m$. Essas características são consistentes com aquelas utilizadas durante o treinamento offline do regressor de Floresta Aleatória, assegurando coerência entre as fases de aprendizado e inferência.

O vetor de características é então submetido ao modelo de Floresta Aleatória treinado, que realiza uma inferência em tempo real para estimar o consumo total de energia

associado à execução do módulo m no dispositivo d (Algoritmo 2, linha 9). Ao comparar os valores de energia previstos entre todos os dispositivos candidatos (linhas 10–14), o algoritmo identifica o dispositivo que minimiza a pegada energética esperada para o módulo em questão. Uma vez selecionado esse dispositivo energeticamente ótimo, a decisão de alocação é encaminhada à rotina de viabilidade e execução definida no Algoritmo 3 (linha 15).

Algorithm 3 Alocar Módulo em Dispositivo

Require: $m, bestDevice$
1: **if** length$_m \leq C_{bestDevice}$ **then** ▷ Restrição de capacidade Eq. (2)
2: Alocar o módulo m ao $bestDevice$
3: Atualizar a capacidade restante de CPU do $bestDevice$
4: **else**
5: $parent \leftarrow$ getParent($bestDevice$)
6: **if** $parent \neq -1$ **then**
7: placeModuleOnDevice($m, parent$) ▷ Recursão para o dispositivo pai
8: **else**
9: Alocar o módulo m à Nuvem ▷ Fallback final
10: Atualizar os recursos da Nuvem
11: **end if**
12: **end if**

O Algoritmo 3 assegura a viabilidade de recursos, ao mesmo tempo em que preserva a natureza energeticamente consciente da decisão de alocação. Ao receber o par candidato $(m, bestDevice)$, o algoritmo primeiramente verifica se a capacidade de CPU remanescente do dispositivo selecionado é suficiente para acomodar a carga do módulo, conforme definido pela restrição de capacidade na Eq. (2) (linhas 1–3). Caso a restrição seja satisfeita, o módulo é alocado e o estado do dispositivo é atualizado de acordo.

Quando o dispositivo energeticamente ótimo não dispõe de capacidade suficiente, o CANOPY ativa um mecanismo hierárquico de *fallback*, descrito explicitamente no Algoritmo 3 (linhas 4–7). O algoritmo avalia recursivamente os dispositivos pais na hierarquia névoa–nuvem, movendo-se progressivamente da borda para a névoa e, se necessário, para a camada de nuvem. Caso nenhum dispositivo em nível de névoa possa hospedar o módulo, a nuvem é selecionada como *fallback* final (linhas 8–10), garantindo a completude da alocação mesmo sob contenção extrema de recursos.

4. Análise Experimental

Todos os algoritmos são avaliados sob cargas de trabalho de aplicações, topologia do sistema e configurações de dispositivos idênticas, de modo a permitir uma comparação controlada. A avaliação concentra-se em três métricas de desempenho extraídas do modelo do sistema: consumo total de energia, latência fim a fim e tráfego de rede. Os experimentos foram projetados para avaliar o desempenho da abordagem proposta em comparação com três estratégias de referência representativas: CB-E [Peixoto et al. 2022], MGA [Rajagopal et al. 2023] e Edgeward, um algoritmo amplamente adotado na literatura de computação em névoa [Gupta et al. 2017]. Essas estratégias representam diferentes filosofias de alocação, possibilitando uma comparação abrangente em termos de eficiência energética, tratamento de latência e utilização de recursos.

Todos os experimentos foram conduzidos utilizando a topologia hierárquica de névoa–nuvem ilustrada na Figura 2. A topologia consiste em uma camada de nuvem no nível mais alto, seguida por um gateway com latência de 100 ms, duas camadas de névoa (Nível de Névoa 0 com latência de 50 ms e Nível de Névoa 1 com latência de 25 ms) e uma camada de IoT na borda da rede. Essa topologia tem sido amplamente adotada em estudos anteriores [Peixoto et al. 2022, Oliveira et al. 2024, Reis et al. 2026], permitindo comparação direta com resultados previamente reportados.

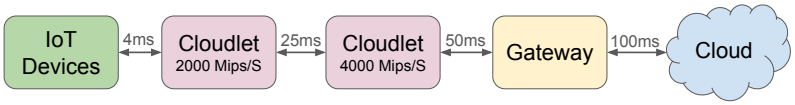


Figura 2. Topologia hierárquica de névoa–nuvem utilizada nos experimentos

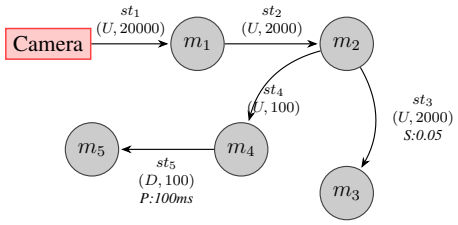
Essa topologia estabelece um ambiente experimental hierárquico e heterogêneo, capaz de capturar os principais trade-offs entre proximidade ao usuário, capacidade computacional e latência de comunicação característicos do contínuo computacional. Os múltiplos níveis de névoa, combinados com um gateway intermediário e uma camada de nuvem, permite avaliar o comportamento das estratégias de alocação sob diferentes regimes de carga e disponibilidade de recursos. Esse cenário fornece a base necessária para analisar como aplicações intensivas em processamento se propagam ao longo da hierarquia e como decisões de alocação impactam desempenho e consumo energético, motivando a escolha da aplicação VSOT, descrita a seguir.

4.1. Aplicação

A avaliação experimental baseia-se em uma aplicação representativa e computacionalmente intensiva: Video Surveillance Object Tracking (VSOT). Essa aplicação foi selecionada devido às suas elevadas demandas de processamento e requisitos rigorosos de latência, tornando-a adequada para avaliar a robustez de estratégias de alocação sob condições de saturação de recursos. Para refletir estresse operacional realista, a carga de trabalho foi escalonada de acordo com a intensidade computacional dos módulos da aplicação VSOT.

A aplicação VSOT envolve tarefas complexas de visão computacional executadas sobre fluxos distribuídos de câmeras, impondo uma carga significativa sobre os recursos de borda e de névoa. Para induzir cenários controlados de sobrecarga e avaliar explicitamente a lógica de realocação descrita no Algoritmo 3, o número de instâncias concorrentes de VSOT foi variado de 1 a 4. Esta faixa de instâncias é representativa para a topologia avaliada pois, devido ao número controlado de nós de névoa, o escalonamento para além deste limite satura completamente os recursos locais. Experimentos preliminares demonstraram que, após 4 instâncias, a alocação torna-se puramente dependente da nuvem para todas as estratégias, eliminando a sensibilidade necessária para validar os mecanismos de decisão na borda e na névoa, que são o foco central desta proposta.

Seguindo o modelo originalmente introduzido por Gupta et al. [Gupta et al. 2017], a aplicação VSOT é implementada no iFogSim como um conjunto de cinco módulos interconectados: Motion Detector, Object Detector, Object Tracker, PTZ Control e User Interface. Em conjunto, esses módulos são responsáveis por detectar movimento, rastrear



(a)

Índice	Componente
m_1	Detector de Movimento
m_2	Detector de Objetos
m_3	Interface do Usuário
m_4	Rastreador de Objetos
m_5	Controle PTZ

(b)

Arco	Ação
st_1	Fluxo Vídeo Bruto
st_2	Fluxo Vídeo com Movimento
st_3	Objeto Detectado
st_4	Localização do Objeto
st_5	Parâmetros PTZ

(c)

Figura 3. Aplicação VSOT: (a) topologia, (b) componentes e (c) transições de estado.

objetos e controlar dinamicamente os parâmetros da câmera para otimizar a cobertura de vigilância.

A Figura 3(a) ilustra a topologia de comunicação da aplicação VSOT, incluindo as interações entre módulos, as direções dos fluxos de dados e as transições de estado. As arestas direcionadas representam os fluxos de dados trocados entre os módulos, anotados com suas respectivas direções (Up ou Down) e tamanhos de dados. Algumas transições incorporam semânticas adicionais de execução, como o parâmetro de seletividade em st_3, enquanto outras operam periodicamente, como no caso de st_5, com um intervalo fixo de 100 ms. As Figuras 3(b) e 3(c) resumem os módulos da aplicação e as correspondentes transições de estado, respectivamente.

Do ponto de vista da execução, o Motion Detector (m_1) processa fluxos de vídeo brutos capturados pela câmera e encaminha eventos de movimento detectados ao Object Detector (m_2). O Object Detector realiza a identificação de objetos e inicializa o rastreamento quando necessário, calculando as coordenadas iniciais do objeto. Essas coordenadas são então processadas pelo Object Tracker (m_4), que atualiza continuamente as posições dos objetos e envia sinais de controle ao módulo PTZ Control (m_5). Por fim, informações relevantes de rastreamento são entregues à User Interface (m_3), fornecendo feedback visual em tempo real ao usuário. Esse fluxo de execução gera padrões diversos de carga de trabalho e perfis de utilização de recursos ao longo do contínuo computacional, os quais são explorados para construir o conjunto de dados utilizado no treinamento e na avaliação do modelo preditivo de energia.

4.2. Dataset

Com base nos rastros de execução e nos padrões de utilização de recursos gerados pelas cargas de trabalho experimentais, foi construído um conjunto de dados personalizado para treinar e avaliar o modelo de predição de energia do CANOPY. O conjunto de dados foi derivado de simulações conduzidas no ambiente iFogSim2 e inclui três perfis representativos de aplicações: o EEG Tractor Beam Game (EEGTBG) para interação humano–computador de baixa latência, o Video Surveillance Object Tracking (VSOT) para cargas de trabalho de visão computacional e um Serviço de Tradução de Áudio (ATS) para tarefas de processamento de linguagem natural. O conjunto de dados bruto compreende 4.194 observações distribuídas em 22 atributos arquiteturais e operacionais. A partir desse espaço de alta dimensionalidade, foi aplicada seleção de características para identificar as variáveis com maior poder explicativo em relação ao consumo de energia. O modelo final utiliza um conjunto híbrido de características:

- **Características numéricas:** capacidade de hardware (C_d), complexidade da tarefa ($length_m$), ciclos de CPU por tupla (L_e^{CPU}) e intensidade de chegada representada pela taxa de requisições (λ_e) combinada com o número de instâncias ativas.
- **Características categóricas:** tipo de aplicação, identificador do módulo de software e identificador do dispositivo de destino, capturando a variabilidade de execução entre diferentes combinações de software e hardware.

4.3. Modelagem de IA

Utilizando o conjunto de dados construído, foi implementado um regressor de Floresta Aleatória (Random Forest (RF)) para modelar o consumo de energia, adotando uma abordagem de aprendizado em conjunto que mitiga a alta variância comumente associada a árvores de decisão individuais, por meio da agregação de predições de múltiplos estimadores não correlacionados. A escolha do modelo justifica-se pela sua capacidade intrínseca de capturar interações não-lineares e complexas entre variáveis heterogêneas, que muitas vezes são negligenciadas por modelos de regressão linear simples. Embora modelos de Aprendizado por Reforço (RL) sejam populares, optou-se pelo RF devido à sua menor complexidade de treinamento e maior interpretabilidade para o problema de predição de energia.

Antes do treinamento, os dados passaram por um pipeline de pré-processamento. O conjunto de dados foi particionado em 80% para treinamento e 20% para validação, utilizando uma semente aleatória fixa de 42 para garantir reprodutibilidade. Variáveis categóricas foram codificadas utilizando um One-Hot Encoder para evitar a introdução de relações ordinais implícitas, enquanto as características numéricas foram padronizadas utilizando um *StandardScaler*. Essa normalização impede que características com magnitudes maiores influenciem de forma desproporcional os critérios de divisão durante a partição recursiva. A arquitetura final do modelo consiste em 100 árvores, otimizadas para execução paralela e inferência de baixa latência durante decisões de alocação em tempo de execução.

A eficácia preditiva do regressor foi avaliada utilizando métricas estatísticas padrão. O modelo alcançou um Coeficiente de Determinação (R^2) de 0,9523 e um Erro Absoluto Médio (MAE) de 1676,8781, indicando que aproximadamente 95,23% da variância no consumo de energia é explicada pelas características selecionadas. Esse nível de acurácia valida o modelo como um substituto confiável para a estimativa de energia, fornecendo precisão suficiente para apoiar a alocação de módulos energeticamente consciente em tempo real em ambientes dinâmicos de contínuo computacional.

5. Resultados e Discussão

Esta seção avalia experimentalmente o desempenho do CANOPY em comparação com estratégias do estado da arte. Para garantir a robustez estatística e a reprodutibilidade dos resultados, cada cenário experimental foi executado 10 vezes, utilizando diferentes sementes aleatórias para a geração de cargas de trabalho e eventos de rede.

5.1. Análise de Latência

A Figura 4 ilustra o impacto do aumento da intensidade da carga de trabalho sobre a latência fim a fim da aplicação VSOT. Sob cargas leves (uma a duas instâncias), todas as

estratégias avaliadas mantêm níveis de latência comparáveis, indicando que a disponibilidade de recursos na borda é suficiente para atender à demanda. No entanto, à medida que o número de instâncias aumenta para três e quatro, as abordagens de referência apresentam uma degradação acentuada de desempenho. Em particular, Edgewards e MGA exibem picos de latência superiores a 175 ms, refletindo sua capacidade limitada de lidar com a saturação de recursos nas camadas inferiores.

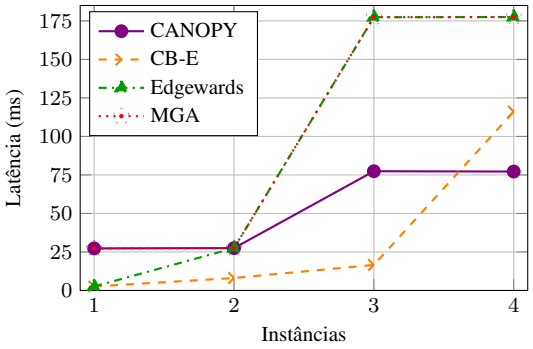


Figura 4. Latência por instância

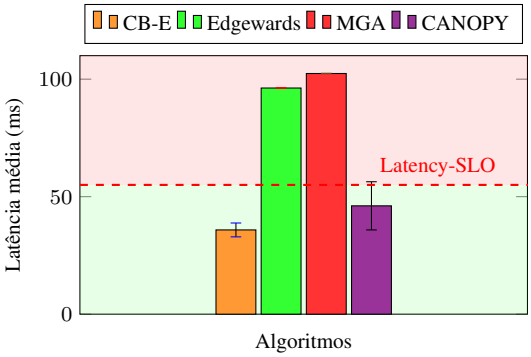


Figura 5. Latência média

Em contraste, o CANOPY mantém um perfil de latência estável à medida que a intensidade da carga de trabalho aumenta, estabilizando-se em aproximadamente 75 ms para maiores números de instâncias. Esse comportamento resulta da lógica de alocação preditiva, que aloca proativamente módulos em nós de névoa com maior capacidade de processamento e melhor eficiência energética por unidade de computação, evitando os efeitos de congestionamento observados em estratégias baseadas apenas em proximidade.

Os resultados de latência média, resumidos na Figura 5, reforçam essa observação. Enquanto Edgewards e MGA atingem latências médias de 96,23 ms e 102,41 ms, respectivamente, ambas acima do limiar Latency-SLO de 55 ms, o CANOPY mantém uma latência média de 46,10 ms, permanecendo dentro do limite operacional aceitável. Embora o CB-E alcance a menor latência média (35,85 ms) ao priorizar a execução na borda, essa vantagem é frágil e se deteriora rapidamente quando as restrições de capacidade local são violadas.

5.2. Análise de Utilização de Rede

A evolução do tráfego de rede por instância é apresentada na Figura 6. De forma semelhante ao comportamento da latência, a utilização da rede permanece comparável entre todas as estratégias sob condições de carga leve. À medida que o sistema entra em um regime de sobrecarga, as estratégias de alocação passam a divergir significativamente. O CANOPY apresenta um aumento gradual no tráfego de rede conforme a intensidade da carga cresce, refletindo o descarregamento de módulos para camadas superiores do contínuo.

A Figura 7 quantifica esse compromisso. Enquanto o CB-E minimiza a utilização da rede (aproximadamente 6,70 KB) ao restringir a execução aos dispositivos locais de borda, essa abordagem frequentemente resulta em gargalos de processamento e instabilidade de desempenho. Em contraste, o CANOPY incorre em um custo médio de rede mais elevado (aproximadamente 96,72 KB), que permanece abaixo do limiar operacional definido. Esse aumento é uma consequência deliberada do mecanismo recursivo de

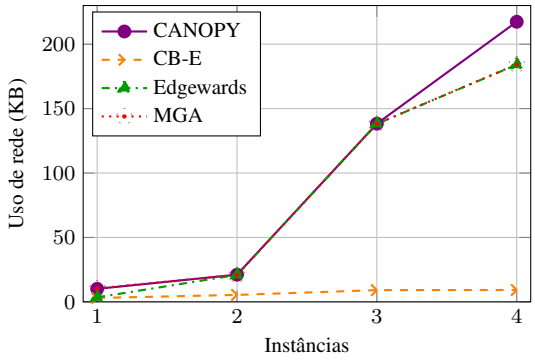


Figura 6. Tráfego por instância

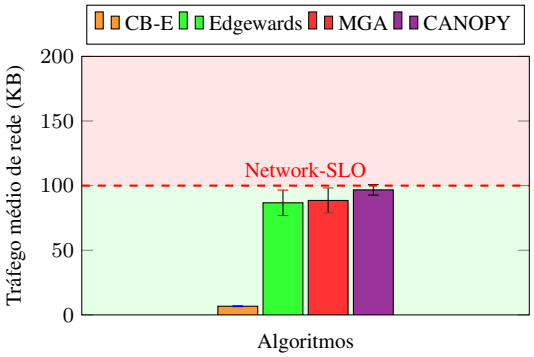


Figura 7. Utilização média de rede

realocação definido no Algoritmo 3, que redistribui módulos entre camadas de névoa e nuvem quando as restrições de capacidade local (C_d) são atingidas. Embora o CANOPY aceite um aumento controlado no tráfego de rede, ele evita penalidades severas de latência e exaustão prematura de recursos.

5.3. Análise de Consumo de Energia

A eficiência energética é o principal objetivo de otimização da abordagem proposta. A Figura 8 apresenta o consumo total de energia à medida que a carga de trabalho é escalonada. O CANOPY alcança consistentemente o menor consumo de energia entre todas as estratégias avaliadas, com a diferença de desempenho tornando-se mais pronunciada sob maiores intensidades de carga.

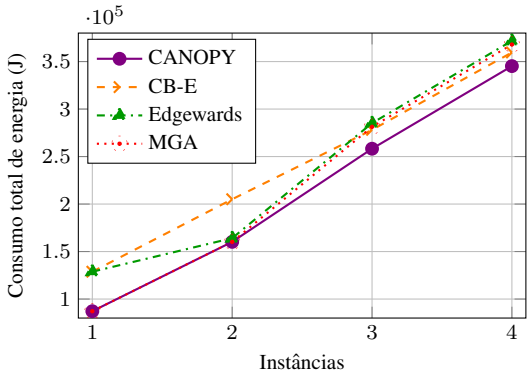


Figura 8. Energia por instância

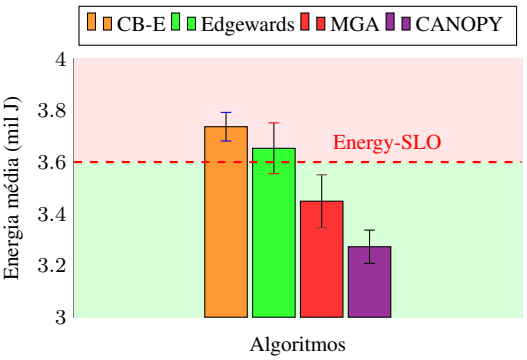


Figura 9. Consumo médio energia

Os resultados médios de consumo de energia, apresentados na Figura 9, fornecem uma comparação mais clara sob condições de carga de trabalho idênticas. Enquanto CB-E e Edgewards consomem aproximadamente 3736,89 J e 3653,41 J, respectivamente, ambos acima do limiar Energy-SLO de 3600 J, o CANOPY atinge um consumo médio de apenas 3272,59 J. Isso corresponde a uma redução de 12,42% em relação ao CB-E, 10,42% em relação ao Edgewards e 5,11% em relação ao MGA.

Esses resultados demonstram que o CANOPY equilibra eficazmente a carga computacional no contínuo computacional por meio de modelagem preditiva de energia, priorizando dispositivos energeticamente eficientes e adaptando-se à saturação de recursos, com economias sustentadas de energia sem comprometer a latência e a estabilidade do sistema.

6. Conclusão

Este trabalho apresentou o CANOPY, uma abordagem orientada por IA para a alocação de módulos em ambientes hierárquicos do contínuo computacional, que integra predição de consumo energético diretamente ao processo decisório. Ao empregar um regressor de Floresta Aleatória com elevada acurácia preditiva ($R^2 = 0{,}9523$), o CANOPY transforma a energia em um critério explícito e antecipável de alocação, permitindo minimizar a pegada energética de aplicações de IoT sem comprometer a estabilidade operacional, mesmo sob saturação severa de recursos. Avaliações experimentais no simulador iFogSim2, utilizando a aplicação intensiva VSOT, demonstraram reduções médias de consumo energético de até 12,42% em relação ao CB-E, 10,42% ao Edgewards e 5,11% ao MGA. Esses resultados evidenciam que características do ambiente, dos nós e das aplicações são suficientes para que modelos baseados em ensemble capturem de forma eficiente as dinâmicas energéticas não lineares do contínuo computacional, oferecendo uma solução escalável, eficiente e adequada para operação em tempo real em cenários de borda, névoa e nuvem.

Disponibilidade de Artefatos

Em aderência aos princípios da Ciência Aberta, o código-fonte e o dataset utilizados neste trabalho podem ser acessados em: https://github.com/gisellyreis/canopy.

Agradecimentos

Este estudo foi financiado, em parte, pela CAPES – Código de Financiamento 001. Também contou com apoio, em parte, da FAPESB, bem como do CNPq, por meio do processo nº 403231/2023-0.

Referências

Ansere, J. A., Gyamfi, E., Sharma, V., Shin, H., Dobre, O. A., and Duong, T. Q. (2024). Quantum deep reinforcement learning for dynamic resource allocation in mobile edge computing-based iot systems. *IEEE Transactions on Wireless Communications*, 23(6):6221–6233.

Atwal, S., Singh, D., and Chhabra, A. (2025). Resource allocation strategies for edge and fog computing in the cloud continuum. In *2025 International Conference on Intelligent Control, Computing and Communications (IC3)*, pages 208–213.

Bai, W. and Qian, C. (2021). Deep reinforcement learning for joint offloading and resource allocation in fog computing. In *2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS)*, pages 131–134.

Goudarzi, M., Palaniswami, M., and Buyya, R. (2023). A distributed deep reinforcement learning technique for application placement in edge and fog computing environments. *IEEE Transactions on Mobile Computing*, 22(5):2491–2505.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., and Buyya, R. (2017). ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296.

Huang, Y. and Xie, T. (2024). Edge computing resource allocation method based on federated reinforcement learning. In *2024 2nd International Conference on Signal Processing and Intelligent Computing (SPIC)*, pages 254–258.

Karthiga, M., Paramasivam, J., Raja, P. V., Kumar Suman, S., Bhagyalakshmi, L., and Poddar, H. (2025). Optimizing edge computing performance using machine learning for low-latency applications. In *2025 World Skills Conference on Universal Data Analytics and Sciences (WorldSUAS)*, pages 1–6.

Li, X., Wang, N., Li, D., Ruan, J., and Liu, H. (2025). A cloud-edge-devices computing power allocation algorithm based on intelligent collaborative computing. In *2025 IEEE 3rd International Conference on Image Processing and Computer Applications (ICIPCA)*, pages 157–161.

Oliveira, L. T., Bittencourt, L. F., Genez, T. A., de Lara, E., and Peixoto, M. L. (2024). Enhancing modular application placement in a hierarchical fog computing: A latency and communication cost-sensitive approach. *Computer Communications*, 216:95–111.

Peixoto, M. L. M., Genez, T. A. L., and Bittencourt, L. F. (2022). Hierarchical scheduling mechanisms in multi-level fog computing. *IEEE Transactions on Services Computing*, 15(5):2824–2837.

Rajagopal, S. M., Supriya, M., and Buyya, R. (2023). Resource provisioning using meta-heuristic methods for iot microservices with mobility management. *IEEE Access*, 11:60915–60938.

Reis, G., Oliveira, L., Prazeres, C., Veiga, L., and Peixoto, M. (2026). Elsa: Energy-aware and latency-sensitive resource allocation in edge-cloud continuum. *Future Generation Computer Systems*, 182:108506.

Ren, Z., Jiang, W., Liu, A., Huang, S., Mu, J., Liu, S., and Gu, W. (2025). Network resource optimization for mobile edge computing: A deep learning approach. In *2025 IEEE 34th Wireless and Optical Communications Conference (WOCC)*, pages 117–121.

Tran-Dang, H., Bhardwaj, S., Rahim, T., Musaddiq, A., and Kim, D.-S. (2022). Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues. *Journal of Communications and Networks*, 24(1):83–98.

Wang, C., Guo, R., Xiu, P., Li, X., and Wang, H. (2024). Research on resource scheduling algorithm optimization in cloud computing environment based on deep learning. In *2024 4th International Conference on Electronic Information Engineering and Computer Communication (EIECC)*, pages 918–922.

Yuan, X., Sun, M., and Lou, W. (2022). A dynamic deep-learning-based virtual edge node placement scheme for edge cloud systems in mobile environment. *IEEE Transactions on Cloud Computing*, 10(2):1317–1328.

Zhang, D.-G., Zhang, J., Zhu, X.-M., Zhang, T., Zheng, X.-M., and Jia, H.-J. (2025). Novel approach of computational resource allocation in fog computing based on deep reinforcement learning strategies. *IEEE Internet of Things Journal*, 12(20):43829–43841.