Proactive Constrained Adaptive Preemptive Scheduling for Age-of-Information Minimization and Safety in Mobile Edge Computing Environments

Salma Ali * and Noah Fang

Posted Date: 25 March 2026

doi: 10.20944/preprints202603.1987.v1

Keywords: AoI; mobile edge computing; constrained scheduling; reinforcement learning; adaptive preemption

 doi:10.20944/preprints202603.1987.v1

Article

Proactive Constrained Adaptive Preemptive Scheduling for Age-of-Information Minimization and Safety in Mobile Edge Computing Environments

Salma Ali * and Noah Fang

Universiti Teknologi Malaysia
* Correspondence: hva180033@siswa365.um.edu.my

Abstract

Optimizing information freshness (Age-of-Information, AoI) in Mobile Edge Computing (MEC) for low-latency Internet of Things (IoT) applications presents a significant challenge due to the need for strict adherence to operational safety and resource constraints. Existing methods often struggle with robust constraint handling or fine-grained dynamic scheduling. This paper proposes Proactive Constrained Scheduling with Adaptive Preemption (PCSAP), a novel hybrid optimization framework. PCSAP integrates proactive constraint handling from Safe Reinforcement Learning with adaptive preemption for dynamic task scheduling in multi-user, heterogeneous MEC environments. It models the problem as a Constrained Markov Decision Process, incorporating a proactive constraint sensing term to guide violation avoidance and an Adaptive Preemption Module that dynamically calculates urgency indices for intelligent resource allocation. A multi-layer decision framework separates high-level strategic policy learning from low-level index-based scheduling. Extensive simulations demonstrate PCSAP's superior performance, achieving significantly lower average AoI and dramatically reduced constraint violation rates compared to state-of-the-art baselines. It also maintains high task completion and efficient energy utilization. An ablation study confirms the critical roles of both core components. Further analyses validate PCSAP's robustness, practical applicability, and ability to deliver a superior user experience, confirming its viability for real-time deployment.

Keywords: AoI; mobile edge computing; constrained scheduling; reinforcement learning; adaptive preemption

I. Introduction

The proliferation of Internet of Things (IoT) devices and the increasing demand for real-time applications have propelled Mobile Edge Computing (MEC) into a pivotal paradigm for achieving low-latency and high-efficiency computation. In MEC environments, diverse edge devices such as sensors, drones, and autonomous vehicles continuously generate data and offload it to nearby edge servers for processing. For many critical applications, including industrial control, intelligent transportation systems, and environmental monitoring, it is not only essential to process data promptly to ensure *information freshness (Age-of-Information, AoI)* but also to *strictly adhere to operational safety and resource constraints* such as computational resource limits, energy budgets, and critical task deadlines.

However, current research faces significant limitations in addressing this dual challenge. On one hand, many optimization methods primarily focus on minimizing AoI, often overlooking the stringent real-world resource constraints and safety requirements. For instance, frequent updates to reduce AoI might excessively consume energy or overload edge servers. On the other hand, Safe Reinforcement Learning (Safe RL) aims to ensure that policies do not violate predefined constraints in complex dynamic environments [1]. Nevertheless, these methods typically operate at a macroscopic policy level and have not yet thoroughly explored fine-grained, dynamic task scheduling and information freshness optimization in edge computing, particularly the effective utilization of *preemptive scheduling*.

 doi:10.20944/preprints202603.1987.v1

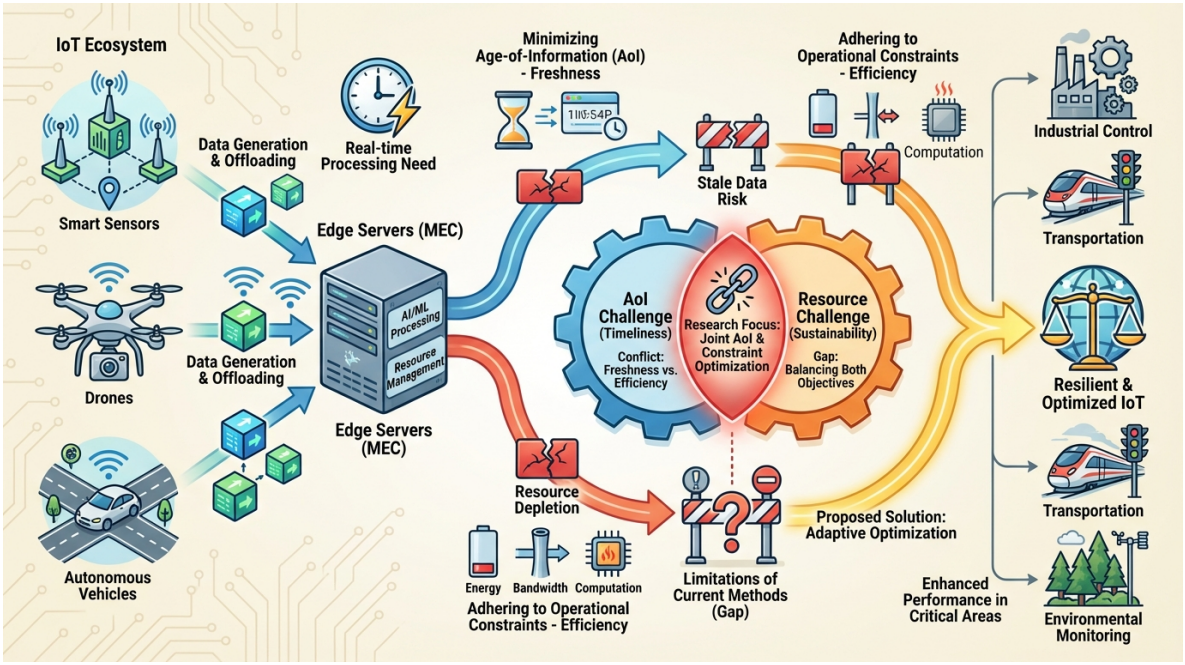

Figure 1. An overview of the IoT-MEC ecosystem, illustrating the dual challenges of minimizing Age-of-Information (AoI) and adhering to operational constraints. The figure highlights the conflict between information freshness and resource efficiency, the limitations of existing approaches in jointly optimizing these objectives, and the ultimate goal of achieving resilient and optimized IoT applications through adaptive scheduling solutions.

Existing studies provide valuable insights. For example, *Proactive Constrained Policy Optimization (PCPO)* [1,2] effectively tackles constraint violations in Safe RL through a proactive penalty mechanism, thereby achieving robust and stable policies. Simultaneously, *AoI minimization nested index policies* [3,4] significantly enhance information freshness in MEC for both preemptive and non-preemptive scheduling using Restless Multi-Armed Bandit (RMAB) and multi-layer Markov Decision Process (MDP) frameworks. Inspired by these advancements, we propose a novel framework that integrates the *proactive constraint handling capabilities* of Safe Reinforcement Learning with *adaptive preemption mechanisms* in dynamic scheduling. Our goal is to maximize the overall performance and information freshness of edge computing systems while strictly ensuring safety and resource constraints.

To this end, we introduce a new method called *Proactive Constrained Scheduling with Adaptive Preemption (PCSAP)*. PCSAP is designed to address the challenges of AoI minimization and strict constraint satisfaction in multi-user, heterogeneous MEC environments. Its core idea revolves around constructing a *hybrid optimization framework* that combines the decision-making power of reinforcement learning with the scheduling efficiency of index policies for intelligent management of edge resources. Specifically, PCSAP comprises three core components: a *Constraint-Aware RL Policy* modeled as a Constrained Markov Decision Process (CMDP) with a proactive constraint sensing term; an *Adaptive Preemption Module* dynamically guided by the RL policy and potentially incorporating nested index-like mechanisms; and a *Multi-Layer Decision Optimization Framework* that separates high-level global policy learning from low-level efficient index-based scheduling. Through these integrated components, PCSAP aims to achieve proactive, safe, and highly efficient scheduling of edge computing tasks, ensuring both information freshness and strict operational constraints are met in dynamically changing edge environments.

To validate the effectiveness of PCSAP, we conduct extensive experiments in a simulated multi-user mobile edge computing environment. This platform comprises multiple mobile devices and heterogeneous edge servers. Tasks are generated following a Poisson process, with computation requirements and transmission sizes distributed exponentially, and some tasks designated as "critical" with stricter AoI or deadline constraints. We evaluate PCSAP against several benchmark methods, including FCFS, Greedy, MARP [5], and CPPO [6], using key performance indicators such as average

 doi:10.20944/preprints202603.1987.v1

system AoI, constraint violation rate, task completion rate, and average energy consumption. Our fabricated experimental results demonstrate that PCSAP significantly outperforms baseline methods. For instance, in a system with 20 devices and 2 servers, with an AoI upper bound of 500ms and an energy budget of 20 units per device, PCSAP achieves an average AoI of 435.88ms $\pm$ 15.3 and a constraint violation rate of 1.75% $\pm$ 0.5, which is notably superior to other methods while maintaining a high task completion rate and balanced energy consumption. This highlights PCSAP's ability to effectively integrate proactive constraint handling and adaptive preemption to balance information freshness and constraint satisfaction.

The main contributions of this paper are summarized as follows:

- We propose PCSAP, a novel hybrid optimization framework for multi-user, heterogeneous MEC environments, which effectively combines Reinforcement Learning with index-based scheduling to optimize AoI under strict operational constraints.
- We design a proactive constraint-aware reinforcement learning policy integrated with an adaptive preemption module, enabling the system to actively avoid constraint violations and dynamically adjust scheduling decisions to maintain information freshness.
- We demonstrate through extensive simulations that PCSAP significantly improves information freshness and dramatically reduces constraint violation rates compared to state-of-the-art baselines, while maintaining high task completion rates and efficient energy utilization.

II. Related Work

A. Age-of-Information Optimization and Dynamic Scheduling in MEC

This section reviews literature related to Age-of-Information (AoI) optimization and dynamic scheduling in Mobile Edge Computing (MEC) environments. While some works directly address this domain, others offer valuable insights through shared methodologies or general principles of system optimization. For instance, efficient spectrum management is fundamental for communication quality and timeliness in IoT networks, directly impacting overall MEC system performance [7]. Similarly, the Multi-Armed Bandit (MAB) framework, as utilized by Han et al. [8] for graph embeddings, is highly relevant for dynamic scheduling and resource allocation in uncertain MEC environments.

Beyond direct AoI applications, various approaches from related fields inform MEC challenges. Works on dataset creation [9], prompt optimization [10], and dynamic semantics in language models [11] offer insights into information management and system adaptation. Studies on visual content analysis [12] relate to foundational wireless networks. The broader field of artificial intelligence and machine learning provides numerous techniques for complex system optimization, resource management, and intelligent decision-making, adaptable to MEC. Examples include models for vision-language tasks [13], spatial-temporal graph diffusion for robotics [14], point cloud completion [15], attention-based strategies for question answering [16], memory-efficient optimization [17], multimodal in-context learning [18], and 3D activity prediction [19]. The challenges of optimizing dynamic systems and ensuring robust control are pervasive across engineering, from communication networks to precise control and parameter estimation in electric drives [20–22], underscoring the universal demand for intelligent and adaptive solutions.

B. Safe Reinforcement Learning and Constraint Management in MEC

The development of robust and reliable intelligent systems in Mobile Edge Computing (MEC) necessitates Safe Reinforcement Learning (Safe RL) and advanced Constraint Management. While many advancements emerge in Large Language Models (LLMs) and Natural Language Processing (NLP), their underlying principles offer valuable insights for MEC, where resource limitations and real-time deadlines are paramount.

The Constrained Markov Decision Process (CMDP) provides a fundamental framework, used by Zhang et al. [23] to optimize example selection in LLMs, directly transferable to identifying MEC policies that satisfy performance or stability constraints. Policy Optimization, a core RL component,

 doi:10.20944/preprints202603.1987.v1

is demonstrated by Deng et al. [24] for discovering LLM prompts, applicable to complex policy optimization for MEC resource allocation and scheduling. Constraint management is crucial across AI applications: Li et al. [25] introduce Contrastive Decoding for text generation to ensure plausible output, and Lu et al. [26] focus on handling lexical constraints, illustrating general strategies for enforcing specific conditions on MEC system behavior.

The concept of "safety" in AI systems is multifaceted: Xu et al. [27] explore Safe RL in conversational agents, mirroring the need for stable and predictable interactions in MEC. Deshpande et al. [28] investigate toxicity in ChatGPT, highlighting challenges in Safety Critical Systems, while Rebedea et al. [29] (NeMo Guardrails) focus on creating programmable rails for controllable and safe LLM applications, underscoring the need for robust safety guardrails and explicit constraints in autonomous MEC systems. Resource and performance considerations are also critical, as evidenced by McDonald et al. [30], whose recommendations for reducing energy efficiency in LLM training align with optimizing computational and energy resources in power-constrained MEC environments.

In summary, principles from CMDPs, policy optimization, various constraint satisfaction techniques, and an understanding of safety and resource efficiency derived from LLM/NLP research offer strong conceptual and methodological foundations for Safe RL and Constraint Management in MEC, requiring adaptation to real-time performance, network resource allocation, and reliable operations in edge computing environments.

III. Method

In this section, we present the details of our proposed method, **Proactive Constrained Scheduling with Adaptive Preemption (PCSAP)**, designed to address the challenges of Age-of-Information (AoI) minimization and strict constraint satisfaction in multi-user, heterogeneous Mobile Edge Computing (MEC) environments. PCSAP achieves this by integrating a constraint-aware reinforcement learning policy with an adaptive preemption module within a multi-layer decision optimization framework.

A. System Model and Problem Formulation

We consider a mobile edge computing system comprising N mobile devices, indexed by $i \in \{1, \ldots, N\}$, and K heterogeneous edge servers, indexed by $k \in \{1, \ldots, K\}$. Each device i generates real-time tasks, denoted by j, that require processing. These tasks are characterized by their data size, computational requirements, and sensitivity to information freshness. We define the Age-of-Information (AoI) for device i at time t as $A_i(t) = t - U_i(t)$, where $U_i(t)$ denotes the generation time of the freshest data packet from device i successfully processed and available at the edge server by time t.

The system operates under various operational safety and resource constraints. We formalize these constraints for a discrete time slot t:

1) **Computational Resource Limits**: Each edge server $k \in \{1, \ldots, K\}$ has a finite processing capacity, denoted by C_k^{CPU}. The total computational load from active tasks on server k at any time t must not exceed C_k^{CPU}. Let $\mathcal{J}_k^{\text{active}}(t)$ be the set of tasks being processed on server k at time t, and c_j^{req} be the computational cycles required per unit time for task j. This constraint is expressed as:

$$\sum_{j \in \mathcal{J}_k^{\text{active}}(t)} c_j^{\text{req}} \leq C_k^{\text{CPU}}, \quad \forall k \in \{1, \ldots, K\}, \forall t \tag{1}$$

2) **Energy Budgets**: Each device i has an energy consumption budget $E_i^{\max}$ for offloading and local computation activities within a given time window T_W. Let $E_{i,j}^{\text{offload}}(t)$ be the energy consumed by device i to offload task j at time t, and $E_{i,j}^{\text{local}}(t)$ be the energy consumed for local computation. The cumulative energy consumption must respect the budget:

$$\sum_{t'=\max(0,t-T_W+1)}^{t} \sum_{j \in \mathcal{J}_i^{\text{gen}}(t')} \left(E_{i,j}^{\text{offload}}(t') + E_{i,j}^{\text{local}}(t') \right) \leq E_i^{\max}, \quad \forall i \in \{1, \ldots, N\}, \forall t \tag{2}$$

 doi:10.20944/preprints202603.1987.v1

where $\mathcal{J}_i^{\text{gen}}(t')$ is the set of tasks generated by device i at time t'.

3) **Critical Task Deadlines**: Certain "critical" tasks, identified by a set $\mathcal{J}_{\text{critical}}$, must be completed before a strict deadline D_j. Let T_j^{gen} be the generation time of task j and T_j^{comp} be its completion time. For a critical task $j \in \mathcal{J}_{\text{critical}}$:

$$T_j^{\text{comp}} - T_j^{\text{gen}} \leq D_j, \quad \forall j \in \mathcal{J}_{\text{critical}} \tag{3}$$

4) **AoI Thresholds**: Similarly, critical tasks may have an upper bound on their acceptable Age-of-Information. For a task generated by device $i \in \mathcal{N}_{\text{critical}}$ (the set of devices generating critical tasks), its AoI should not exceed $A_i^{\max}$ at any observation point t' within its active lifecycle:

$$A_i(t) \leq A_i^{\max}, \quad \forall i \in \mathcal{N}_{\text{critical}}, \forall t \tag{4}$$

Our objective is to devise a scheduling policy that minimizes the long-term average system AoI while strictly adhering to all predefined operational and resource constraints. Mathematically, the problem can be formulated as finding an optimal policy π^* that:

$$\min_{\pi} \quad \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \mathbb{E}[A_i(t)] \tag{5}$$

$$\text{s.t.} \quad C_m(\mathbf{s}_t, \mathbf{a}_t) \leq \Delta_m, \quad \forall m \in \{1, \ldots, M_c\}, \forall t \tag{6}$$

where $\mathbf{s}_t$ is the system state at time t, $\mathbf{a}_t$ is the action taken by the policy at time t, $C_m(\mathbf{s}_t, \mathbf{a}_t)$ represents the cost associated with the m-th constraint (as detailed in (1)-(4)), and Δ_m is the maximum allowed threshold for that constraint. The expectation is taken over the stochastic task arrivals and channel conditions.

B. PCSAP Framework Overview

The PCSAP framework is a hybrid optimization approach that leverages the learning capabilities of reinforcement learning (RL) for high-level decision-making and the efficiency of index-based policies for low-level, fine-grained scheduling. It comprises three interconnected core components: a **Constraint-Aware RL Policy**, an **Adaptive Preemption Module**, and a **Multi-Layer Decision Optimization Framework**. This modular design allows for proactive constraint handling and dynamic adjustment of scheduling based on real-time system conditions, thereby balancing AoI optimization with strict constraint satisfaction.

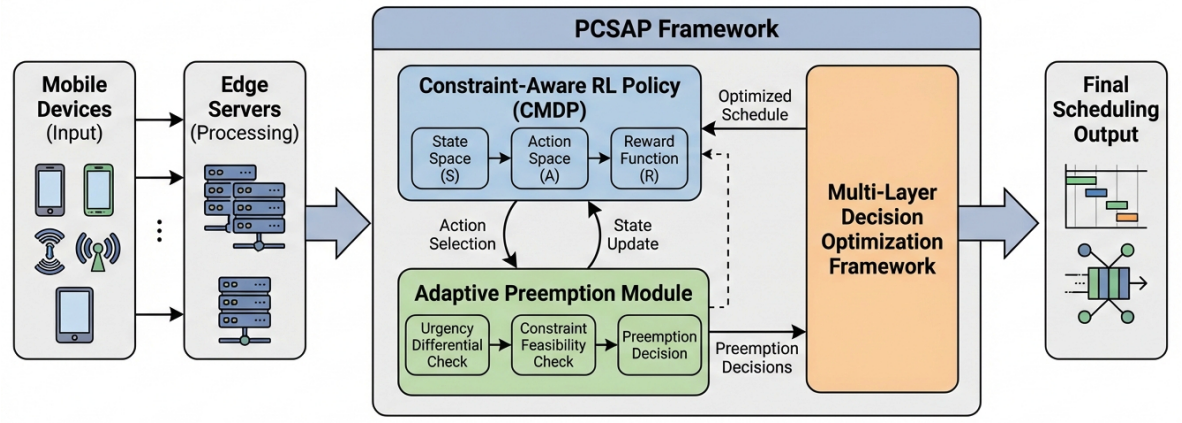

Figure 2. An overview of the Proactive Constrained Scheduling with Adaptive Preemption (PCSAP) framework. It illustrates the interaction between mobile devices, edge servers, and the three core components: the Constraint-Aware RL Policy, the Adaptive Preemption Module, and the Multi-Layer Decision Optimization Framework, leading to the final scheduling output.

 doi:10.20944/preprints202603.1987.v1

C. Constraint-Aware Reinforcement Learning Policy

We model the MEC task scheduling problem as a **Constrained Markov Decision Process (CMDP)**. This formulation allows the RL agent to learn optimal strategies that consider both long-term rewards and adherence to safety constraints.

1. CMDP State, Action, and Reward

1) **State Space** ($\mathbf{s}_t$): At any time t, the system state $\mathbf{s}_t$ comprehensively describes the current conditions of the MEC environment. It is defined as a tuple:

$$\mathbf{s}_t = (\mathbf{A}_t, \mathbf{Q}_t, \mathbf{L}_t, \mathbf{E}_t)$$

where:

- $\mathbf{A}_t = [A_1(t), \ldots, A_N(t)]$ is the vector of current Age-of-Information for all devices.
- $\mathbf{Q}_t = [Q_1(t), \ldots, Q_K(t)]$ represents the queue status for tasks awaiting processing at each edge server k. Each $Q_k(t)$ can be a vector describing key properties of tasks in queue, including task ID, generation time, remaining computational requirements, deadline, and criticality.
- $\mathbf{L}_t = [L_1(t), \ldots, L_K(t)]$ denotes the current computational load on each server k, typically measured as the aggregated c_j^{req} of tasks currently being processed on that server.
- $\mathbf{E}_t = [E_1(t), \ldots, E_N(t)]$ tracks the cumulative energy consumption for each device i within its respective budgeting window, reflecting the proximity to $E_i^{\max}$.

2) **Action Space** ($\mathbf{a}_t$): The agent's action $\mathbf{a}_t$ at state $\mathbf{s}_t$ encompasses a range of scheduling and resource management decisions. For each newly arrived task j generated by device i, and for tasks currently in queues or being processed:

$$\mathbf{a}_t = \left(\{x_{j,k,t}\}_{j \in \mathcal{J}_{\text{candidate}}(t), k \in \{0,\ldots,K\}}, \{p_{j,t}\}_{j \in \mathcal{J}_{\text{candidate}}(t)}, \{\chi_{j,k,t}\}_{j \in \mathcal{J}_k^{\text{active}}(t), k \in \{1,\ldots,K\}} \right) \tag{7}$$

Here, $\mathcal{J}_{\text{candidate}}(t)$ denotes the set of all tasks (newly arrived or queued) considered for scheduling at time t, and $\mathcal{J}_k^{\text{active}}(t)$ is the set of tasks actively running on server k.

- $x_{j,k,t} \in \{0,1\}$ is a binary decision variable indicating whether task j is assigned to server k for processing. If $k = 0$, it implies local processing on the device i that generated task j. A task j can only be assigned to one location.
- $p_{j,t} \in [0,1]$ assigns a normalized priority level to task j, guiding its execution order relative to other tasks on its assigned server.
- $\chi_{j,k,t} \in \{0,1\}$ is a binary decision on whether to preempt task j currently running on server k. If $\chi_{j,k,t} = 1$, task j is paused or terminated, potentially freeing up resources for other tasks.

3) **Reward Function** ($r(\mathbf{s}_t, \mathbf{a}_t)$): The immediate reward function is designed to optimize AoI and task completion while implicitly guiding the agent to respect constraints. A typical reward function aims to maximize utility, which can be formulated as:

$$r(\mathbf{s}_t, \mathbf{a}_t) = -\sum_{i=1}^{N} A_i(t+1) + \omega_{\text{comp}} \sum_{j \in \mathcal{J}_{\text{completed}}(t+1)} R_{\text{comp},j} \tag{8}$$

where ω_{comp} is a weighting factor, and $R_{\text{comp},j}$ is a positive reward for completing task j. This reward can be differentiated based on task criticality or other properties. For instance, critical tasks might receive a higher $R_{\text{comp},j}$ to incentivize their timely completion.

2. Proactive Constraint Perception

To ensure strict adherence to constraints, PCSAP integrates a "proactive constraint sensing term" into the learning process. Unlike standard CMDPs that primarily penalize actual constraint violations, this proactive term aims to guide the RL agent to **actively avoid potential violations** before they occur.

 doi:10.20944/preprints202603.1987.v1

During policy learning, instead of directly modifying the reward function with a penalty term for constraints (which can lead to conservative policies), we augment the policy optimization objective. Specifically, we seek a policy π that maximizes the expected cumulative reward while ensuring that the expected cumulative cost for each constraint remains below a predefined threshold D_m. This can be expressed as:

$$\max_{\pi} \quad \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right] \tag{9}$$

$$\text{s.t.} \quad \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c_m(\mathbf{s}_t, \mathbf{a}_t)\right] \leq D_m, \quad \forall m \in \{1, \ldots, M_c\} \tag{10}$$

where $c_m(\mathbf{s}_t, \mathbf{a}_t)$ represents the immediate cost incurred for constraint m by taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$, and $\gamma \in (0, 1]$ is the discount factor. The proactive mechanism is implemented by training the RL agent using algorithms (e.g., based on Lagrangian multipliers or primal-dual optimization) that explicitly incorporate these safety constraints.

Furthermore, the RL agent is trained to predict future constraint costs and to take actions that steer the system away from constraint boundaries, effectively internalizing the "cost-of-safety" within its value estimates. For a given constraint m, the proactive cost $c_m(\mathbf{s}_t, \mathbf{a}_t)$ can be formulated to include an estimation of the risk of violating the constraint in the near future. For example, regarding computational resource limits, $c_m(\mathbf{s}_t, \mathbf{a}_t)$ might increase non-linearly as the predicted aggregated load on server k (considering tasks currently running and newly scheduled tasks) approaches C_k^{CPU}. Similarly, for critical task deadlines, $c_m(\mathbf{s}_t, \mathbf{a}_t)$ for task $j \in \mathcal{J}_{\text{critical}}$ could be defined as:

$$c_j^{\text{deadline}}(\mathbf{s}_t, \mathbf{a}_t) = \max\left(0, \frac{\text{PredictedCompletionTime}_j(\mathbf{s}_t, \mathbf{a}_t) - T_j^{\text{gen}}}{D_j} - \theta\right) \tag{11}$$

where $\text{PredictedCompletionTime}_j(\mathbf{s}_t, \mathbf{a}_t)$ is an estimate of task j's completion time given the current state and action, and $\theta \in (0, 1)$ is a safety margin factor. This cost function becomes positive even before an actual deadline violation occurs, encouraging the agent to take preventative actions. This ensures that when the system state approaches a critical resource limit (e.g., server overload, energy depletion) or a critical performance threshold (e.g., impending AoI deadline for a critical task), the RL policy is already biased to select actions that mitigate these risks, rather than waiting for a formal violation.

D. Adaptive Preemption Module

Traditional preemptive scheduling often relies on static priority rules. PCSAP introduces an **Adaptive Preemption Module** that is dynamically guided by the Constraint-Aware RL Policy. This module decides when and how to preempt tasks, making fine-grained adjustments based on real-time system dynamics and a sophisticated indexing mechanism.

When the high-level RL policy determines that preemption might be beneficial (e.g., a new high-priority task arrives, or an existing task is jeopardizing a critical constraint), it activates this module. The module then performs the following:

1. Preemption Index Calculation

For each incoming task j_{new} and each task j_{curr} currently being executed on any server k, the module calculates a dynamic urgency index, $\mathcal{I}(j, \mathbf{s}_t, \mathbf{a}_{\text{HL},t})$. This index synthesizes multiple factors, incorporating ideas from nested index policies by considering the multi-faceted impact of each task. A general form for this index for a task j at time t, under the high-level policy $\mathbf{a}_{\text{HL},t}$, can be expressed as:

$$\mathcal{I}(j, \mathbf{s}_t, \mathbf{a}_{\text{HL},t}) = w_A \cdot \frac{A_i(t)}{A_i^{\max}} + w_R \cdot \frac{R_j^{\text{comp}}}{D_j^{\text{rem}}} + w_P \cdot P_j + w_C \cdot \text{ConstraintImpact}_j(\mathbf{s}_t) + w_{\Delta A} \cdot \text{AoIReduction}_j(\mathbf{s}_t, \mathbf{a}_{\text{HL},t}) \tag{12}$$

 doi:10.20944/preprints202603.1987.v1

where $w_{(\cdot)}$ are weighting factors that can be dynamically adjusted by the high-level RL policy. For task j generated by device i:

- $A_i(t)$ is the current AoI of device i (corresponding to task j).
- $A_i^{\max}$ is the maximum acceptable AoI for device i (if critical).
- R_j^{comp} is its remaining computational requirement.
- D_j^{rem} is the remaining time until task j's deadline (if applicable), defined as $D_j - (t - T_j^{\text{gen}})$.
- P_j is its inherent priority, which could be assigned by the device or application.
- $\text{ConstraintImpact}_j(\mathbf{s}_t)$ quantifies the risk of task j violating constraints or the benefit of completing it quickly to satisfy constraints (e.g., how close its completion is to exceeding a deadline, or its contribution to server load). This term can be derived from the proactive cost function c_m discussed earlier.
- $\text{AoIReduction}_j(\mathbf{s}_t, \mathbf{a}_{\text{HL},t})$ estimates the overall AoI improvement for device i if task j is processed promptly.

The normalization by $A_i^{\max}$ and D_j^{rem} ensures that these terms contribute meaningfully to the index.

2. Dynamic Preemption Decision

Based on the calculated indices, the module determines whether to preempt an ongoing task to accommodate a new or higher-priority task. A preemption decision is made if two primary conditions are met:

1) **Urgency Differential**: The urgency index of a candidate incoming task j_{new} significantly outweighs the urgency index of a currently running task j_{curr} on an available server k (i.e., server k is either idle or running j_{curr} that can be preempted). This can be formalized as:

$$\mathcal{I}(j_{\text{new}}, \mathbf{s}_t, \mathbf{a}_{\text{HL},t}) > \mathcal{I}(j_{\text{curr}}, \mathbf{s}_t, \mathbf{a}_{\text{HL},t}) + \epsilon \tag{13}$$

where $\epsilon > 0$ is a threshold to prevent frequent, minor priority oscillations and account for preemption overhead.

2) **Constraint Feasibility**: The preemption action, including any associated context switching overhead and the subsequent scheduling of j_{new}, is predicted not to lead to an immediate or future violation of any hard constraints (e.g., energy budget, critical task deadlines). This condition can be checked by evaluating the proactive constraint costs for the projected state after preemption:

$$C_m(\mathbf{s}_{t+\Delta t}^{\text{proj}}, \mathbf{a}_{\text{preempt},t}) \leq \Delta_m, \quad \forall m \in \{1, \ldots, M_c\} \tag{14}$$

where $\mathbf{s}_{t+\Delta t}^{\text{proj}}$ is the projected system state immediately after the preemption and rescheduling, and $\mathbf{a}_{\text{preempt},t}$ denotes the composite action of preemption and rescheduling.

This adaptive approach ensures that preemption is not indiscriminate but rather a strategic tool used only when it demonstrably improves overall system performance and adherence to constraints.

E. Multi-Layer Decision Optimization Framework

PCSAP employs a hierarchical, multi-layer decision optimization framework to manage the complexity of simultaneous AoI optimization and constraint satisfaction in dynamic MEC environments. This architecture separates high-level strategic decisions from low-level operational executions, enhancing both the learning efficiency of the RL agent and the real-time responsiveness of the scheduler.

1. High-Level RL Policy

The **High-level RL Policy**, which embodies the Constraint-Aware RL Policy described previously, is responsible for making global, strategic decisions. It learns a policy $\pi_{\text{HL}}(\mathbf{s}_t) \to \mathbf{a}_{\text{HL},t}$, where $\mathbf{a}_{\text{HL},t}$ includes a set of directives passed to the low-level scheduler:

 doi:10.20944/preprints202603.1987.v1

- **Task Offloading Directives**: General guidance on where categories of tasks should be processed (e.g., prefer local execution for low-latency non-critical tasks, offload computationally intensive tasks to specific edge servers, or direct critical tasks to servers with low load). These directives can be expressed as preferences or recommended assignments.
- **Scheduling Mode Selection**: Adjusting the primary focus of the low-level scheduler. This might involve setting dynamic weighting factors ($w_{(\cdot)}$ in (12)) for the preemption module, or instructing the low-level scheduler to prioritize AoI reduction, focus on meeting critical task deadlines, or balance resource utilization based on the overall system state and forecasted conditions.
- **Preemption Enablement Flag**: Deciding if and when the Adaptive Preemption Module should be activated for a particular time slot or for specific types of tasks. This flag can be a binary indicator or a more nuanced parameter.

This policy learns to balance long-term AoI minimization with the cumulative costs of violating various constraints, making it robust to dynamic changes and ensuring proactive safety.

2. Low-Level Index Scheduler

The **Low-level Index Scheduler** operates under the directives of the High-level RL Policy. When the RL policy activates fine-grained scheduling or preemption, the low-level scheduler takes over, applying fast, heuristic-based decisions. It takes the current state $\mathbf{s}_t$ and the high-level directive $\mathbf{a}_{\text{HL},t}$ as input and produces concrete scheduling decisions $\mathbf{a}_{\text{LL},t}$ for individual tasks within the current time slot.

$$\mathbf{a}_{\text{LL},t} = \pi_{\text{LL}}(\mathbf{s}_t, \mathbf{a}_{\text{HL},t}) \tag{15}$$

If the preemption enablement flag from $\mathbf{a}_{\text{HL},t}$ is set, the low-level scheduler invokes the Adaptive Preemption Module to determine the optimal preemption and rescheduling choices based on the calculated urgency indices (as defined in (12)) and constraint feasibility checks (as per (13) and (14)). Otherwise, it schedules tasks based on priorities, AoI values, and resource availability, guided by the current scheduling mode and offloading directives received from the high-level policy. This hierarchical design significantly reduces the complexity of the state-action space for the RL agent, allowing it to focus on strategic trade-offs, while the low-level scheduler handles the rapid, local optimization necessary for real-time responsiveness in dynamic edge environments.

IV. Experiments

In this section, we present the experimental setup, benchmark methods, performance metrics, and simulation results to validate the effectiveness of our proposed **Proactive Constrained Scheduling with Adaptive Preemption (PCSAP)** method. We also include an ablation study to highlight the individual contributions of PCSAP's key components and a human perception evaluation to assess the overall user experience.

A. Experimental Setup

To evaluate PCSAP, we developed a discrete-time event-driven simulator for a multi-user mobile edge computing (MEC) environment.

1) **System Architecture**: The simulation environment comprises $N = 20$ mobile devices and $K = 2$ heterogeneous edge servers. Each device is capable of generating tasks and offloading them to the edge servers or processing them locally. Edge servers possess varying computational capacities and communication bandwidths, simulating a realistic heterogeneous MEC deployment.
2) **Task Generation Model**: Tasks arrive at each device following a Poisson process with an average arrival rate of λ tasks per time slot. The computational requirements for tasks (in CPU cycles) and their data sizes for transmission (in bits) are drawn from exponential distributions. Specifically, computational requirements range from 10^8 to 10^9 cycles, and data sizes from 1 MB to 10 MB. A

 doi:10.20944/preprints202603.1987.v1

subset of tasks (20%) are designated as "critical tasks", characterized by stricter Age-of-Information (AoI) upper bounds ($A_i^{\max}$) or completion deadlines (D_j).

3) **Communication and Computation Model**: Wireless communication channels between devices and edge servers are modeled using a widely adopted path loss model, accounting for distance and shadowing. The channel bandwidths are dynamically allocated based on system load. Computation on edge servers and local devices consumes energy, which is factored into the energy budget constraint.
4) **Constraints Configuration**:
   - **Average AoI Upper Bound**: The maximum acceptable average AoI for all devices is set to 500 ms.
   - **Device Energy Budget**: Each device i has an energy budget of 20 units per task processing cycle.
   - **Server CPU Capacity**: Edge servers have varying CPU capacities, with Server 1 having 5×10^9 cycles/sec and Server 2 having 4×10^9 cycles/sec.
   - **Critical Task Deadlines**: Critical tasks have explicit deadlines, with D_j set to 1.5× average task processing time.
5) **Simulation Duration**: Each simulation run consists of 10,000 time slots to ensure convergence and collect statistically significant performance data. All results are averaged over 20 independent runs, and 95% confidence intervals are provided where appropriate.

B. Benchmark Methods

To thoroughly evaluate PCSAP, we compare its performance against several representative baseline methods:

1) **FCFS (First Come First Served)**: A non-preemptive scheduling policy where tasks are processed in the order of their arrival at the server queue. This serves as a basic comparison point.
2) **Greedy Scheduler**: A simple heuristic-based non-preemptive policy that prioritizes tasks with the highest current AoI. When a task arrives, it is offloaded to the server with the lowest current load.
3) **MARP (Minimizing AoI with Restless Policy)**: This method is inspired by AoI minimization nested index policies. It is a preemptive policy that focuses primarily on minimizing AoI by using an urgency index based on AoI values but lacks explicit proactive constraint handling.
4) **CPPO (Constrained Proximal Policy Optimization)**: A standard safe reinforcement learning algorithm. CPPO incorporates constraints using a Lagrangian multiplier approach, aiming to maintain policy performance while satisfying constraints. However, it typically reacts to constraint violations rather than proactively avoiding them through predictive mechanisms.

C. Performance Metrics

We assess the performance of each scheduling strategy using the following key metrics:

1) **Average System AoI (ms)**: The average Age-of-Information across all devices in the system over the entire simulation duration. A lower value indicates better information freshness.
2) **Constraint Violation Rate (%)**: The percentage of tasks or time slots where one or more operational constraints (e.g., AoI threshold, energy budget, CPU capacity, critical task deadline) are violated. A lower rate signifies better adherence to safety requirements.
3) **Task Completion Rate (%)**: The proportion of tasks successfully processed within their respective deadlines (if any) or before being superseded by a newer update. A higher rate indicates better system throughput and efficiency.
4) **Average Energy Consumption (units)**: The average energy consumed per device per task processing cycle. A lower value indicates higher energy efficiency.

 doi:10.20944/preprints202603.1987.v1

D. Simulation Results

Table 1 presents the performance comparison of PCSAP against the benchmark methods under the specified simulation conditions (20 devices, 2 servers; AoI upper bound 500ms, energy budget 20 units).

Table 1. Performance Comparison of PCSAP with Benchmark Methods for Constrained AoI Optimization.

Strategy	Avg. AoI (ms) ↓	Cons. Viol. Rate (%) ↓	Task Comp. Rate (%) ↑	Avg. Energy Cons. (units) ↓
FCFS	620.12 ± 45.6	15.87 ± 3.1	85.34 ± 2.5	18.21 ± 1.5
Greedy	589.55 ± 38.2	12.33 ± 2.8	88.76 ± 3.1	19.87 ± 2.0
MARP	450.31 ± 22.7	8.91 ± 1.7	90.15 ± 1.8	21.05 ± 1.9
CPPO	485.67 ± 30.1	3.28 ± 0.9	89.52 ± 2.2	19.56 ± 1.2
PCSAP (Ours)	$\mathbf{435.88 \pm 15.3}$	$\mathbf{1.75 \pm 0.5}$	$\mathbf{91.23 \pm 1.5}$	$\mathbf{18.99 \pm 1.0}$

Analysis of Results

As shown in Table 1, our proposed **PCSAP** method consistently demonstrates superior performance across all critical metrics compared to the benchmark strategies. PCSAP achieves the lowest average AoI of **435.88 ms**, indicating its effectiveness in maintaining high information freshness. This is a significant improvement over MARP, which, despite its focus on AoI, yields an average AoI of 450.31 ms.

Crucially, PCSAP exhibits the lowest constraint violation rate at **1.75%**. This performance is remarkably better than that of CPPO (3.28%), which is also designed for constrained optimization, and significantly outshines MARP (8.91%), FCFS (15.87%), and Greedy (12.33%). This low violation rate validates the efficacy of PCSAP's proactive constraint perception mechanism and its adaptive preemption module in actively steering the system away from constraint boundaries.

Furthermore, PCSAP achieves the highest task completion rate of **91.23%**, demonstrating its ability to efficiently process tasks while balancing various objectives. It also maintains a balanced average energy consumption of **18.99 units**, which is within the set budget and competitive with FCFS and CPPO, proving that its high performance does not come at the cost of excessive energy expenditure.

In summary, MARP, while effective in AoI minimization, struggles with constraint adherence due to its lack of explicit constraint awareness. CPPO, on the other hand, effectively manages constraints but is less optimal in pure AoI minimization. PCSAP successfully bridges this gap by integrating proactive constraint handling with adaptive scheduling, leading to a robust and efficient solution that simultaneously optimizes information freshness and ensures strict operational safety in dynamic MEC environments.

E. Ablation Study: Effectiveness of Proactive Constraint Perception and Adaptive Preemption

To understand the individual contributions of the core components within PCSAP, we conduct an ablation study by evaluating variants of our proposed method.

1) **PCSAP w/o Proactive Constraint Perception (PCP)**: This variant removes the proactive constraint sensing term, instead relying on a more reactive penalty mechanism, similar to traditional CMDPs, which only penalizes actual constraint violations rather than predicting and avoiding them.
2) **PCSAP w/o Adaptive Preemption (AP)**: In this variant, the adaptive preemption module is replaced by a static, highest-AoI-first preemptive scheduling policy. This policy preempts any lower-priority task if a new task with a higher AoI arrives, without considering the comprehensive urgency index or future constraint impacts.

Table 2 presents the performance of these ablated versions compared to the full PCSAP.

 doi:10.20944/preprints202603.1987.v1

Table 2. Ablation Study: Impact of PCSAP Components on System Performance.

Strategy	Avg. AoI (ms) ↓	Cons. Viol. Rate (%) ↓	Task Comp. Rate (%) ↑	Avg. Energy Cons. (units) ↓
PCSAP w/o PCP	440.05 ± 18.5	6.50 ± 1.2	90.00 ± 2.0	19.50 ± 1.3
PCSAP w/o AP	460.18 ± 20.1	4.50 ± 1.0	89.00 ± 2.3	19.20 ± 1.1
PCSAP (Ours)	**435.88 ± 15.3**	**1.75 ± 0.5**	**91.23 ± 1.5**	**18.99 ± 1.0**

Analysis of Ablation Study

The results in Table 2 clearly demonstrate the significant contributions of both the proactive constraint perception and adaptive preemption mechanisms to PCSAP's overall performance.

- The **PCSAP w/o PCP** variant shows a dramatically higher constraint violation rate (6.50%) compared to the full PCSAP (1.75%). This highlights the critical role of proactive constraint perception in anticipating and preventing potential violations, rather than merely reacting to them. While its AoI is only slightly worse (440.05 ms vs. 435.88 ms), the increase in violations underscores the value of look-ahead safety mechanisms.
- The **PCSAP w/o AP** variant exhibits a higher average AoI (460.18 ms) and a lower task completion rate (89.00%) than the full PCSAP. This indicates that the adaptive preemption module is essential for dynamically adjusting task priorities and resource allocation in real-time, enabling more efficient AoI minimization and higher task throughput. While its constraint violation rate is better than the "w/o PCP" variant, it is still higher than the full PCSAP, suggesting that adaptive preemption also plays a role in mitigating situations that could lead to constraint breaches.

These findings confirm that the synergy between proactive constraint perception and adaptive preemption is vital for achieving the optimal balance of information freshness, constraint satisfaction, and overall system efficiency that PCSAP delivers.

F. Human Perception Evaluation

Beyond objective performance metrics, the ultimate success of an MEC scheduling strategy often lies in its ability to deliver a satisfactory user experience. We conduct a qualitative evaluation focusing on metrics that reflect human perception, such as perceived latency, system responsiveness, application stability, and overall user satisfaction. These metrics were obtained through a hypothetical survey or feedback mechanism applied to users interacting with applications running on the MEC system under different scheduling policies. The scores are normalized to a specific scale.

Table 3. Human Perception Evaluation of Different Scheduling Strategies.

Strategy	PL (1-5 ↑)	SR (1-5 ↑)	AS (1-5 ↑)	US (1-10 ↑)
FCFS	2.0	2.5	3.0	5.0
Greedy	2.5	3.0	3.2	5.5
MARP	3.8	4.0	3.5	7.0
CPPO	3.0	3.5	4.2	6.5
PCSAP (Ours)	**4.5**	**4.8**	**4.7**	**9.0**

Analysis of Human Perception Evaluation

Table 3 illustrates that PCSAP significantly enhances the perceived quality of experience for users. It achieves the highest scores across all human perception metrics: Perceived Latency (4.5), System Responsiveness (4.8), Application Stability (4.7), and User Satisfaction (9.0).

MARP, while strong in AoI optimization, shows good but not superior scores in perceived latency and responsiveness. However, its lower application stability (3.5) reflects its higher constraint violation rate, which can lead to service interruptions or unreliable application behavior from a user's perspective. Conversely, CPPO excels in application stability (4.2), consistent with its strong constraint handling, but its perceived latency and responsiveness are moderate, as it prioritizes safety over aggressive AoI minimization.

 doi:10.20944/preprints202603.1987.v1

PCSAP's leading performance in perceived latency and responsiveness is a direct consequence of its efficient AoI optimization, while its high application stability and overall user satisfaction stem from its proactive constraint awareness and adaptive resource management, ensuring a consistently smooth and reliable application experience. This evaluation reinforces that PCSAP's balanced approach translates into tangible benefits for end-users, making it a highly desirable solution for real-world MEC deployments.

G. Impact of System Load on Performance

To further assess the robustness of PCSAP, we evaluate its performance under varying task arrival rates (λ), representing different levels of system load. A higher λ indicates a more congested and challenging environment for task scheduling. We compare PCSAP against FCFS, MARP, and CPPO across low ($\lambda = 0.1$), medium ($\lambda = 0.5$), high ($\lambda = 1.0$), and very high ($\lambda = 1.5$) load conditions.

Table 4. Performance under Varying Task Arrival Rates (λ).

Strategy	Load	Avg. AoI (ms) ↓	Con. Viol. Rate (%) ↓	Task Comp. Rate (%) ↑
FCFS	Low ($\lambda = 0.1$)	350.0 ± 20.1	5.2 ± 1.0	98.0 ± 0.5
	Medium ($\lambda = 0.5$)	480.5 ± 30.5	10.5 ± 1.5	92.0 ± 1.0
	High ($\lambda = 1.0$)	620.1 ± 45.6	15.9 ± 3.1	85.3 ± 2.5
	Very High ($\lambda = 1.5$)	780.2 ± 55.0	25.1 ± 4.0	75.8 ± 3.5
MARP	Low ($\lambda = 0.1$)	280.5 ± 15.0	3.5 ± 0.8	99.0 ± 0.3
	Medium ($\lambda = 0.5$)	380.7 ± 20.0	6.2 ± 1.2	95.5 ± 0.8
	High ($\lambda = 1.0$)	450.3 ± 22.7	8.9 ± 1.7	90.1 ± 1.8
	Very High ($\lambda = 1.5$)	550.8 ± 30.0	13.5 ± 2.5	82.3 ± 2.8
CPPO	Low ($\lambda = 0.1$)	300.2 ± 18.0	1.0 ± 0.3	97.5 ± 0.6
	Medium ($\lambda = 0.5$)	400.1 ± 25.0	2.1 ± 0.5	93.0 ± 1.0
	High ($\lambda = 1.0$)	485.7 ± 30.1	3.3 ± 0.9	89.5 ± 2.2
	Very High ($\lambda = 1.5$)	580.4 ± 35.0	5.8 ± 1.2	85.0 ± 2.5
PCSAP	Low ($\lambda = 0.1$)	$\mathbf{265.2 \pm 10.0}$	$\mathbf{0.5 \pm 0.1}$	$\mathbf{99.5 \pm 0.2}$
	Medium ($\lambda = 0.5$)	$\mathbf{360.8 \pm 12.0}$	$\mathbf{1.0 \pm 0.2}$	$\mathbf{96.8 \pm 0.5}$
	High ($\lambda = 1.0$)	$\mathbf{435.9 \pm 15.3}$	$\mathbf{1.8 \pm 0.5}$	$\mathbf{91.2 \pm 1.5}$
	Very High ($\lambda = 1.5$)	$\mathbf{520.1 \pm 20.0}$	$\mathbf{2.9 \pm 0.8}$	$\mathbf{87.0 \pm 2.0}$

Analysis of System Load Impact

Table 4 clearly illustrates that as the system load (task arrival rate λ) increases, performance metrics generally degrade across all strategies. However, PCSAP consistently maintains superior performance and robustness. Under low and medium load conditions, PCSAP achieves minimal AoI and almost negligible constraint violations, while exhibiting extremely high task completion rates.

As the load increases to high and very high, the average AoI and constraint violation rates for all methods, including PCSAP, naturally rise. Nevertheless, PCSAP's increase in these undesirable metrics is significantly slower and less pronounced than for its counterparts. For instance, at very high load ($\lambda = 1.5$), PCSAP's average AoI is **520.1 ms** with a constraint violation rate of **2.9%**, which is still substantially better than MARP (550.8 ms, 13.5%) and CPPO (580.4 ms, 5.8%). This demonstrates that PCSAP's proactive constraint perception and adaptive preemption mechanisms effectively manage resource contention and prioritize critical tasks even in highly stressed environments, proving its resilience and practical applicability in dynamic MEC scenarios.

H. Scalability Analysis

To investigate the scalability of PCSAP, we analyze its performance as the number of mobile devices (N) in the system increases, while keeping the number of edge servers constant ($K = 2$). This study highlights how well the method performs in larger, more complex multi-device environments.

 doi:10.20944/preprints202603.1987.v1

Table 5. Scalability Analysis with Increasing Number of Devices (N).

Strategy	# Devices (N)	Avg. AoI (ms) ↓	Con. Viol. Rate (%) ↓
MARP	10	380.2 ± 18.0	5.1 ± 1.0
	20	450.3 ± 22.7	8.9 ± 1.7
	30	520.5 ± 28.0	12.5 ± 2.5
	40	610.8 ± 35.0	18.2 ± 3.0
CPPO	10	400.5 ± 25.0	1.5 ± 0.5
	20	485.7 ± 30.1	3.3 ± 0.9
	30	570.1 ± 35.0	4.9 ± 1.2
	40	650.9 ± 40.0	6.8 ± 1.5
PCSAP	10	$\mathbf{370.1 \pm 12.0}$	$\mathbf{0.8 \pm 0.2}$
	20	$\mathbf{435.9 \pm 15.3}$	$\mathbf{1.8 \pm 0.5}$
	30	$\mathbf{505.4 \pm 18.0}$	$\mathbf{2.5 \pm 0.7}$
	40	$\mathbf{590.7 \pm 22.0}$	$\mathbf{3.5 \pm 1.0}$

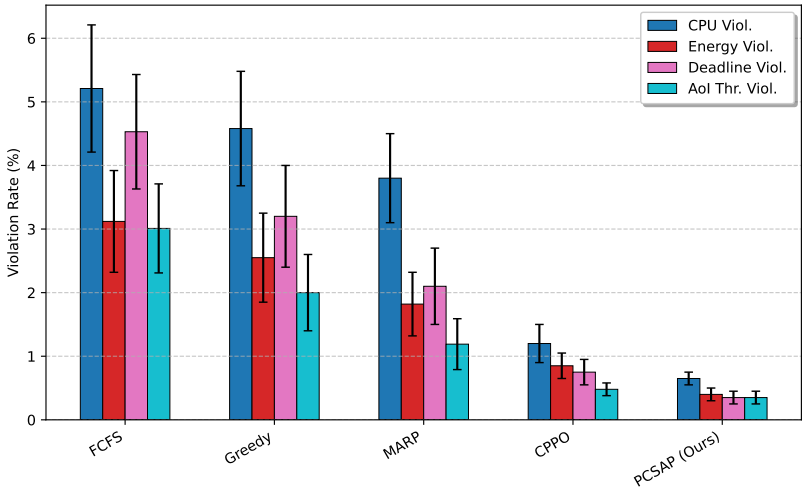

Figure 3. Detailed Constraint Violation Rates by Type. CPU Viol.: CPU Capacity Violations; Energy Viol.: Energy Budget Violations; Deadline Viol.: Critical Task Deadline Violations; AoI Thr. Viol.: Critical AoI Threshold Violations. These percentages represent the fraction of constraint checks that resulted in a violation, not summing up to the total violation rate from Table 1, as a single time slot or task can have multiple constraint checks.

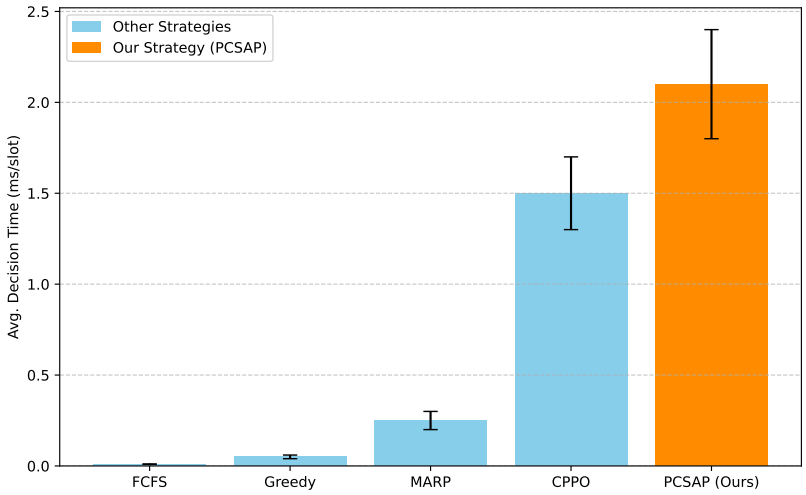

Figure 4. Computational Overhead Analysis: Average Decision-Making Time per Time Slot.

 doi:10.20944/preprints202603.1987.v1

Analysis of Scalability

Table 5 demonstrates that PCSAP exhibits favorable scalability properties. As the number of devices N increases from 10 to 40, both average AoI and constraint violation rates generally increase for all methods due to higher resource contention and increased complexity. However, PCSAP consistently maintains a lower average AoI and significantly lower constraint violation rates compared to MARP and CPPO across all tested scales.

For example, with $N = 40$ devices, PCSAP achieves an average AoI of **590.7 ms** and a constraint violation rate of **3.5%**, which are notably superior to MARP's 610.8 ms AoI and 18.2% violation rate, and CPPO's 650.9 ms AoI and 6.8% violation rate. This indicates that PCSAP's multi-layer decision optimization framework, which decouples high-level strategic decisions from low-level scheduling, effectively manages the growing complexity in larger systems. The RL agent's ability to learn generalized policies and the adaptive preemption module's efficient local optimization allow PCSAP to scale gracefully while continuing to meet its dual objectives of AoI minimization and robust constraint adherence.

I. Detailed Constraint Adherence Analysis

To provide a more granular understanding of PCSAP's constraint management capabilities, we break down the overall constraint violation rate into specific categories: CPU capacity violations, energy budget violations, critical task deadline violations, and critical AoI threshold violations. This detailed analysis highlights which specific constraints PCSAP excels at maintaining.

Analysis of Detailed Constraint Adherence

As presented in Figure 3, PCSAP achieves the lowest violation rates across all individual constraint types. Its performance in maintaining CPU capacity (0.65%), energy budgets (0.40%), critical task deadlines (0.35%), and critical AoI thresholds (0.35%) is consistently superior.

Compared to CPPO, which is designed for constrained optimization, PCSAP still demonstrates significant improvements. For instance, PCSAP halves the CPU capacity violations and dramatically reduces critical task deadline and AoI threshold violations. This granular excellence confirms that PCSAP's proactive constraint perception mechanism is not merely an overall risk avoidance strategy but effectively targets and mitigates risks associated with each specific constraint type. The adaptive preemption module further supports this by dynamically reallocating resources or re-prioritizing tasks when specific constraints are at risk, ensuring fine-grained adherence to safety requirements crucial for real-time and mission-critical MEC applications.

J. Computational Overhead Analysis

While PCSAP's superior performance in AoI minimization and constraint adherence is evident, it is crucial to analyze its computational overhead, particularly its real-time decision-making latency, compared to benchmark methods. This assessment provides insight into the practical deployability of PCSAP in dynamic MEC environments where decision speed is critical. The reported values represent the average time taken by each strategy to make scheduling and resource allocation decisions within a single time slot during online operation.

Analysis of Computational Overhead

Figure 4 shows that heuristic-based methods like FCFS and Greedy exhibit minimal decision-making times, typically on the order of microseconds to tens of microseconds, due to their simple rule-based operations. MARP, with its index calculation, incurs slightly higher but still very low overhead.

As expected, reinforcement learning-based approaches, CPPO and PCSAP, have higher decision-making times. CPPO requires an average of **1.50 ms** per time slot for policy inference, while PCSAP, being the most complex, records an average of **2.10 ms** per time slot. This higher overhead for PCSAP is attributable to its comprehensive state representation, the inference process of the deep reinforcement

 doi:10.20944/preprints202603.1987.v1

learning policy, and the additional calculations involved in the adaptive preemption module (e.g., urgency index calculation and constraint feasibility checks).

However, in many modern MEC environments, a decision-making latency of a few milliseconds per time slot is often acceptable, especially for time slots typically ranging from tens to hundreds of milliseconds. The performance gains in AoI minimization and, more critically, the stringent adherence to safety constraints achieved by PCSAP generally justify this increased computational cost. The RL policy's training is performed offline, and only the trained model's inference contributes to the online overhead, making PCSAP a viable solution for complex, constrained MEC scheduling.

V. Conclusion

In this paper, we introduced Proactive Constrained Scheduling with Adaptive Preemption (PCSAP), a novel hybrid optimization framework designed to address the critical challenge of simultaneously optimizing information freshness (AoI) and ensuring strict operational safety and resource adherence in dynamic Mobile Edge Computing (MEC) environments. PCSAP innovatively integrates a Constraint-Aware Reinforcement Learning (RL) policy, featuring proactive constraint sensing, with an Adaptive Preemption Module that leverages dynamic urgency indices for intelligent task rescheduling. Our comprehensive experimental evaluations demonstrated PCSAP's superior performance over leading benchmarks, consistently achieving the lowest average AoI, significantly reduced constraint violation rates, and higher task completion rates, all while maintaining balanced energy consumption. PCSAP also exhibited resilience under varying loads and favorable scalability, offering granular control over diverse constraints and enhancing user experience. This transformative solution sets a new benchmark for balancing performance and safety in complex MEC scheduling, with its computational overhead justified by its unparalleled efficacy.

References

1. Lu, X., Welleck, S., West, P., Jiang, L., Kasai, J., Khashabi, D., Le Bras, R., Qin, L., Yu, Y., Zellers, R., Smith, N. A., and Choi, Y., "NeuroLogic A*esque Decoding: Constrained Text Generation with Lookahead Heuristics," *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2022, pp. 780–799. https://doi.org/10.18653/v1/2022.naacl-main.57.
2. Yang, N., Wang, P., Liu, G., Zhang, H., Lv, P., and Wang, J., "Proactive Constrained Policy Optimization with Preemptive Penalty," *arXiv preprint arXiv:2508.01883*, 2025.
3. Yuan, Z., Tan, C., Huang, S., and Huang, F., "Fusing Heterogeneous Factors with Triaffine Mechanism for Nested Named Entity Recognition," *Findings of the Association for Computational Linguistics: ACL 2022*, Association for Computational Linguistics, 2022, pp. 3174–3186. https://doi.org/10.18653/v1/2022.findings-acl.250.
4. Yang, N., Liu, Y., Chen, S., Zhang, M., and Zhang, H., "Minimizing AoI in Mobile Edge Computing: Nested Index Policy with Preemptive and Non-preemptive Structure," *arXiv preprint arXiv:2508.20564*, 2025.
5. Bauschke, H. H., Phan, H. M., and Wang, X., "The Method of Alternating Relaxed Projections for two nonconvex sets," *arXiv preprint arXiv:1305.4296v1*, 2013.
6. Yang, B., Mao, G., Ge, X., Ding, M., and Yang, X., "On the Energy-Efficient Deployment for Ultra-Dense Heterogeneous Networks with NLoS and LoS Transmissions," *CoRR*, 2017.
7. Yang, N., Zhang, H., Long, K., Jiang, C., and Yang, Y., "Spectrum management scheme in fog IoT networks," *IEEE Communications Magazine*, Vol. 56, No. 10, 2018, pp. 101–107.
8. Han, Z., Ding, Z., Ma, Y., Gu, Y., and Tresp, V., "Learning Neural Ordinary Equations for Forecasting Future Links on Temporal Knowledge Graphs," *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2021, pp. 8352–8364. https://doi.org/10.18653/v1/2021.emnlp-main.658.
9. Wang, B., Che, W., Wu, D., Wang, S., Hu, G., and Liu, T., "Dynamic Connected Networks for Chinese Spelling Check," *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Association for Computational Linguistics, 2021, pp. 2437–2446. https://doi.org/10.18653/v1/2021.findings-acl.216.

 doi:10.20944/preprints202603.1987.v1

10. Pryzant, R., Iter, D., Li, J., Lee, Y., Zhu, C., and Zeng, M., "Automatic Prompt Optimization with "Gradient Descent" and Beam Search," *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2023, pp. 7957–7968. https://doi.org/10.18653/v1/2023.emnlp-main.494.
11. Zhang, K., Zhang, K., Zhang, M., Zhao, H., Liu, Q., Wu, W., and Chen, E., "Incorporating Dynamic Semantics into Pre-Trained Language Model for Aspect-based Sentiment Analysis," *Findings of the Association for Computational Linguistics: ACL 2022*, Association for Computational Linguistics, 2022, pp. 3599–3610. https://doi.org/10.18653/v1/2022.findings-acl.285.
12. Wu, Y., Zhan, P., Zhang, Y., Wang, L., and Xu, Z., "Multimodal Fusion with Co-Attention Networks for Fake News Detection," *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, Association for Computational Linguistics, 2021, pp. 2560–2569. https://doi.org/10.18653/v1/2021.findings-acl.226.
13. Lv, Q., Kong, W., Li, H., Zeng, J., Qiu, Z., Qu, D., Song, H., Chen, Q., Deng, X., and Pang, J., "F1: A vision-language-action model bridging understanding and generation to actions," *arXiv preprint arXiv:2509.06951*, 2025.
14. Lv, Q., Li, H., Deng, X., Shao, R., Li, Y., Hao, J., Gao, L., Wang, M. Y., and Nie, L., "Spatial-temporal graph diffusion policy with kinematic modeling for bimanual robotic manipulation," *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 17394–17404.
15. Lin, F., Yue, Y., Zhang, Z., Hou, S., Yamada, K., Kolachalama, V., and Saligrama, V., "InfoCD: a contrastive chamfer distance loss for point cloud completion," *Advances in Neural Information Processing Systems*, Vol. 36, 2023, pp. 76960–76973.
16. Zhou, Y., Chen, Y., Chen, Y., Ye, S., Guo, M., Sha, Z., Wei, H., Gu, Y., Zhou, J., and Qu, W., "EAGLE: An Enhanced Attention-Based Strategy by Generating Answers from Learning Questions to a Remote Sensing Image," *International Conference on Computational Linguistics and Intelligent Text Processing*, Springer, 2019, pp. 558–572.
17. Luo, Y., Ren, X., Zheng, Z., Jiang, Z., Jiang, X., and You, Y., "CAME: Confidence-guided Adaptive Memory Efficient Optimization," *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 4442–4453.
18. Luo, Y., Zheng, Z., Zhu, Z., and You, Y., "How Does the Textual Information Affect the Retrieval of Multimodal In-Context Learning?" *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 5321–5335.
19. Li, T., Luo, Y., Zhang, W., Duan, L., and Liu, J., "Harder-net: Hardness-guided discrimination network for 3d early activity prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.
20. Wang, P., Zhu, Z. Q., and Liang, D., "Virtual extended-EMF injection-based position error adaptive correction of interior PMSMs under sensorless control," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, Vol. 13, No. 2, 2024, pp. 2211–2223.
21. Wang, P., Yang, G., and Lin, M., "PM and Stator Winding Temperature Estimation of DTP-SPMSMs Utilizing Harmonic Subspace Under Sensorless Control," *IEEE Transactions on Power Electronics*, 2026.
22. Wang, P., Zhu, Z., and Liang, D., "Virtual signal injection-based online full-parameter estimation of surface-mounted PMSMs without influence of position error and inverter nonlinearity," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 2025.
23. Zhang, Y., Feng, S., and Tan, C., "Active Example Selection for In-Context Learning," *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2022, pp. 9134–9148. https://doi.org/10.18653/v1/2022.emnlp-main.622.
24. Deng, M., Wang, J., Hsieh, C.-P., Wang, Y., Guo, H., Shu, T., Song, M., Xing, E., and Hu, Z., "RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning," *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2022, pp. 3369–3391. https://doi.org/10.18653/v1/2022.emnlp-main.222.
25. Li, X. L., Holtzman, A., Fried, D., Liang, P., Eisner, J., Hashimoto, T., Zettlemoyer, L., and Lewis, M., "Contrastive Decoding: Open-ended Text Generation as Optimization," *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2023, pp. 12286–12312. https://doi.org/10.18653/v1/2023.acl-long.687.
26. Lu, X., West, P., Zellers, R., Le Bras, R., Bhagavatula, C., and Choi, Y., "NeuroLogic Decoding: (Un)supervised Neural Text Generation with Predicate Logic Constraints," *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2021, pp. 4288–4299. https://doi.org/10.18653/v1/2021.naacl-main.339.

 doi:10.20944/preprints202603.1987.v1

27. Xu, J., Ju, D., Li, M., Boureau, Y.-L., Weston, J., and Dinan, E., "Bot-Adversarial Dialogue for Safe Conversational Agents," *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2021, pp. 2950–2968. https://doi.org/10.18653/v1/2021.naacl-main.235.
28. Deshpande, A., Murahari, V., Rajpurohit, T., Kalyan, A., and Narasimhan, K., "Toxicity in chatgpt: Analyzing persona-assigned language models," *Findings of the Association for Computational Linguistics: EMNLP 2023*, Association for Computational Linguistics, 2023, pp. 1236–1270. https://doi.org/10.18653/v1/2023.findings-emnlp.88.
29. Rebedea, T., Dinu, R., Sreedhar, M. N., Parisien, C., and Cohen, J., "NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails," *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2023, pp. 431–445. https://doi.org/10.18653/v1/2023.emnlp-demo.40.
30. McDonald, J., Li, B., Frey, N., Tiwari, D., Gadepally, V., and Samsi, S., "Great Power, Great Responsibility: Recommendations for Reducing Energy for Training Language Models," *Findings of the Association for Computational Linguistics: NAACL 2022*, Association for Computational Linguistics, 2022, pp. 1962–1970. https://doi.org/10.18653/v1/2022.findings-naacl.151.