Structured Reinforcement Learning for Media Streaming at the Wireless Edge

Archana Bura[+], Sarat Chandra Bobbili[*], Shreyas Rameshkumar[*], Desik Rengarajan[*], Dileep Kalathil[*], Srinivas Shakkottai[*]
[+] University of California at San Diego, [*]Texas A&M University
United States of America
{abura@ucsd.edu},{saratb,dileep.kalathil,sshakkot}@tamu.edu,{shreyasr1097,desik.29}@gmail.com

ABSTRACT

Media streaming is the dominant application over wireless edge (access) networks. The increasing softwarization of such networks has led to efforts at intelligent control, wherein application-specific actions may be dynamically taken to enhance the user experience. The goal of this work is to develop and demonstrate learning-based policies for optimal decision making to determine which clients to dynamically prioritize in a video streaming setting. We formulate the policy design question as a constrained Markov decision problem (CMDP), and by using a Lagrangian relaxation we decompose it into single-client problems. Further, the optimal policy takes a threshold form in the video buffer length. We then derive a natural policy gradient (NPG) based constrained reinforcement learning (CRL) algorithm using the structure of our problem, and show that it converges to the globally optimal policy. We then develop a simulation environment for training, and a real-world intelligent controller attached to a WiFi access point for evaluation. We demonstrate using youtube media streaming experiments that our policy can increase the user quality of experience by over 30%. Furthermore, we show that the structured learning is fast, and can be easily deployed, taking only about 15μs to execute.

CCS CONCEPTS

• **Networks** → **Network resources allocation**; • **Computing methodologies** → **Reinforcement learning**.

KEYWORDS

Wireless Networks, Media Streaming, Reinforcement learning

ACM Reference Format:
Archana Bura[+], Sarat Chandra Bobbili[*], Shreyas Rameshkumar[*], Desik Rengarajan[*],, Dileep Kalathil[*], Srinivas Shakkottai[*]. 2024. Structured Reinforcement Learning for Media Streaming at the Wireless Edge. In *The Twenty-fifth Internation Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MOBIHOC'24),October 14–17,2024,Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641512.3686386

MOBIHOC'24, October 14–17,2024, Athens, Greece

ACM ISBN 979-8-4007-0521-2/24/10
https://doi.org/10.1145/3641512.3686386

1 INTRODUCTION

Video streaming has grown in recent years to occupy about 70% of downstream mobile data traffic, with YouTube being the most popular [26]. At the same time, increasing softwarization is taking place in both cellular and WiFi domains, allowing fine grained dynamic control of wireless resource usage to support mobile applications via intelligent control [20, 29]. Indeed, intelligent control has been shown to enable dynamic priorities over the wireless edge (radio access network) for specific applications that require it [17].

The growing interest in intelligent control of wireless edge resources raises the question as to whether dynamic control policies for prioritized spectrum access can be used to obtain appreciable benefits in terms of quality of experience (QoE) at the user end? The answer to this question is nuanced, since such a system would be viable only if the policy can first be computed in some straightforward manner, and it is both simple to implement and robust enough that it can be utilized in a variety of scenarios with temporal variations in the number of clients and their channel qualities.

The goal of this work is to explore the problem of designing policies for dynamic resource allocation at the wireless edge in the specific context of media streaming to mobile devices. A diagram illustrating this use case appears in Figure 1, wherein several YouTube sessions are simultaneously supported by a wireless access point. Each YouTube session has an application state in terms of the video packets buffered and the number of stalls experienced thus far, while the quality of the wireless channel is part of the state of the host mobile device. This state is communicated back to an intelligent controller, which provides a policy for prioritization of selected YouTube sessions. The impact of such a policy in terms of QoE is measured at the end-user and communicated back to the controller, thus completing the feedback loop. Can we show appreciable benefits of intelligent control in this scenario?

The feedback loop shown in Figure 1 suggests that the problem of deciding how best to allocate resources to clients in order to maximize the QoE across all of them could be posed as a Markov Decision Process (MDP). The resource allocation problem could take the form of assigning resource blocks to each end-user over sub-frames in a cellular setting, or assigning YouTube sessions to queues of different priority levels in a WiFi setting. In either case, there are several different service classes, and since resources are limited, there is a constraint across allowable allocations of YouTube sessions to service classes. Thus, we have a constrained MDP (CMDP) as our problem formulation.

Solving such a CMDP is not easy due to the unknown statistics of the many sources of randomness in the system, including interactions between channel quality, the transport protocols used,

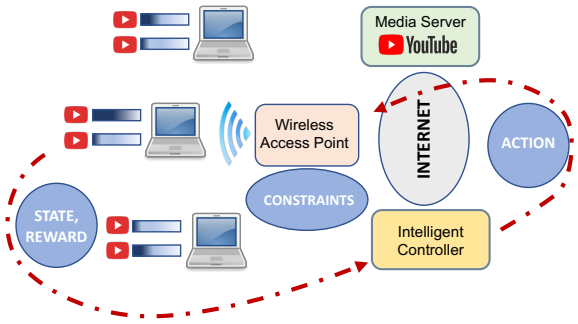

Figure 1: Feedback loop in a media streaming application.

and the video playout rate. This suggests that while perhaps the CMDP might give insight into the nature of the optimal policy, the actual policy to be employed needs to be determined via a data-driven approach using constrained reinforcement learning (CRL). However, reinforcement learning (RL) is known as extremely data hungry. To overcome this, our strategy involves first uncovering the underlying structure of the optimal policy. Using this structure, we can significantly accelerate the learning process, making CRL more efficient and practical. Ultimately, our goal is to determine a sufficiently robust policy that will generalize to a variety of scenarios and is deployable in a real-world edge network.

1.1 Main Results

We first decompose the centralized CMDP problem across all clients into decentralized single-client problems using Lagrangian relaxation techniques. As in several other queueing problem contexts, we find that the optimal policy of such a single-client problem has a threshold structure. Specifically, for any given stall count and constraint cost, the decision on whether to assign high priority service to a client only depends on whether or not its video buffer is below a fixed threshold.

We next consider whether we can provably learn the optimal policy. Here, the threshold structure of the optimal policy enables us to confine our search to learning the threshold parameter using a single neuron (soft threshold), as opposed to a complex function that needs to be approximated with a large neural network. We develop a primal-dual natural policy gradient algorithm, designed for learning such threshold policies under constraints. We show that the error in both the objective and constraints decay with time T as $O(1/\sqrt{T})$. Existing results on the convergence of policy gradient only apply to direct parametrization and soft-max parametrization. Hence, our methodological contribution is to prove the fast convergence of threshold-parametrized natural policy gradient, which applies to a variety of queueing systems.

We next develop a simulation environment to train optimal policies. We show that the single-client decomposition approach learns at about four times faster than centralized learning while attaining a similar performance. Furthermore, the inference time for decision making is only about 10 to 15 μs, as opposed to 50 to 60 μs taken by an unstructured policy, i.e., the approach is much lower in computational requirements and can be run in realtime. We also develop a heuristic index-variant that is robust to a dynamic number of clients or channel variabilities.

Finally, we instantiate a simple intelligent controller platform to evaluate the trained policies in a real-world setting. The controller uses a pub-sub approach to dynamically obtain performance information and to select priority service for particular YouTube clients at a WiFi access point. We implement all policies on our intelligent controller and show using over 50 hours of YouTube streaming experiments that we can attain an improvement in QoE of over 30% as compared to a vanilla policy, while maintaining a perfect QoE score 60-70% of the time. Additionally, we tested the robustness of the index policy under varying network loads and channel conditions. The results showed that our policy not only maintained a higher QoE than other approaches but also adapted effectively to different environmental challenges.

1.2 Related Work

There is an extensive amount of work on CMDPs focusing on the problems of control [2]. In the context of media streaming over wireless channels, several works identify structural properties of the optimal policy [14, 21, 28] that often take a threshold form. While [21, 28] consider maximizing QoE for a finite system, and [28] utilizes a dual-based decentralization approach, [14] focuses on the heavy traffic limit.

Reinforcement learning approaches to find the optimal policies for CMDP problems are well studied in recent literature. In the context of a tabular setting, recent works focus on characterizing the regret [6, 18, 32] or the sample complexity [11, 12, 31] of learning algorithms. There also a number of works on policy gradient approaches for CMDP [1, 7, 35]. However, these works are for the general CMDPs and they do not exploit the structure of the problem to learn a threshold policy with global convergence guarantees.

Other work that considers a learning approach to streaming is [25], which develops a two time-scale stochastic approximation algorithm [4] for an *unconstrained* MDP, and shows certain structural properties of the value function (unimodality), which suggests that a gradient algorithm might converge to the optimal. However, optimality of the fixed point of gradient descent is not shown. We take a very different approach via constrained natural policy gradient, and obtain guarantees for fast convergence to the optimal policy.

A variety of systems have been proposed to improve the QoE performance of video streaming. In software defined networks (SDN), assigning the video streaming flows to network links according to path selection algorithms is considered in [16]. VQOA [22] and QFF [9] employ SDN to monitor the traffic and adapt the bandwidth assignment of video flows to achieve better streaming performance in the home network. Reinforcement learning has also been applied to a variety of video streaming applications [3, 19], which show significant improvements over existing methods. [19] proposes an algorithm Penseive, to train adaptive bit rate (ABR) algorithms to optimize the QoE of the clients. In the similar context, [15] deals with video bit-rate selection for a single client. Although these problems admit RL-based approaches, there is neither a scaling issue, nor any structure of the optimal policy.

Finally, [3] considers the problem of service prioritization for streaming, and shows empirically that off-the-shelf model-based

and model-free RL approaches can result in major improvements in QoE. While completely heuristic in its approach, the work motivates us to consider the problem of optimal policy design, while exploiting problem structure. Thus, we provide a simple, structured RL algorithm with provable convergence guarantees, which is empirically able to attain the same performance as more complex deep learning approaches, while only using significantly fewer data samples, and with much less compute needed during runtime.

2 PROBLEM FORMULATION

We consider the problem of media streaming from a wireless access point (AP) to a set of clients. The AP is capable of instantiating multiple service classes by allocating (limited) resources across clients. We will consider a simple setup with exactly two service classes, referred to as "high" and "low" service classes. The high service class provides a better service rate than the low priority class. An intelligent controller periodically takes decisions on the prioritization of clients to maximize the overall QoE at all clients, under the constraint that only a fixed number may be assigned to the high class.

We consider a discrete time system with N clients that are connected to the AP and compete for resources. At any time t, a media server sends some number of media packets to the AP, which in turns forwards them to the appropriate client, which buffers up this content, while continually attempting to play out a smooth stream. The media playout is said to *stall* when the media buffer is empty, and is an undesirable event. Stall events cause end-user dissatisfaction, and we will utilize a standard model of this disutility in the video streaming context.

State: We denote the state of client n at time t by $s_n(t) = (x_n(t), y_n(t))$, where $x_n(t)$ is the number of packets in the buffer, and $y_n(t)$ is the number of stalls the client n has encountered until time t. We consider a finite state buffer for every client, i.e., $x_n(t) \in [L] \triangleq \{0, 1, \ldots, L\}$. We keep track of stalls up to a finite number, consistent with popular QoE models [8, 10, 34] that observe that user perception of stalls does not change significantly after a few stalls. Hence, we choose $y_n(t) \in [M] \triangleq \{0, 1, \ldots, M\}$. Thus, the state space of client n, denoted by $\mathcal{S}_n$ is $[L] \times [M]$ and the joint system state space is given by $\mathcal{S} \triangleq \otimes_{n=1}^{N} \mathcal{S}_n$. Client state can be expanded to include its channel quality (which evolves independently of controller actions) without changing our analysis.

Actions: The service class assigned to a client n is denoted by action $a_n(t)$, i.e., $a_n(t) \in \mathcal{A}_n \triangleq \{1, 2\}$, where 1 means the high priority service, and 2 means low priority service. Depending upon the choice of the service class, client n obtains $A_n(t)$ incoming packets into its media buffer at time t. Let $\mu_n(a)$ represent the service rate obtained by the n^{th} client, for any action a. We assume that the packet arrival process $A_n(t)$ for every n is a Bernoulli distributed random variable, with a parameter $\mu_n(a)$, depending only upon a.

$$\mu_n(a) = \begin{cases} \mu_{n,1} & a = 1 \\ \mu_{n,2} & a = 2. \end{cases}$$

We assume that $\mu_{n,1} > \mu_{n,2}$, for each n. This means that when the client is assigned to high priority service, its service rate is higher than that of the low priority service. The joint system action space is $\mathcal{A} \triangleq \otimes_{n=1}^{N} \mathcal{A}_n$. Note that we could easily include video bitrate adaptation into the model by simply increasing the action space to multiple dimensions. This causes no changes to the approach.

Policy: We define the joint policy space Π is a mapping from joint state space $\mathcal{S}$ to joint action space $\mathcal{A}$, $\Pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$.

State Transition Structure: The system state has two component-wise transition processes. The first process corresponds to the media buffer evolution, while the second corresponds to the stall count evolution. We assume that the media playback process of client n, denoted by $B_n(t)$ is distributed according to a Bernoulli distribution with mean $\beta_n < \mu_{n,1}$, i.e., stall-free playout is possible if the client is given a high priority service. Furthermore, a stall event corresponds to a transition from a non-zero media buffer state to a zero media buffer state (the buffer cannot be negative).

We assume that the media stream that is currently being played by the client may be terminated at any time with probability $\alpha > 0$ by the end-user (i.e., the user simply stops the playout), which causes a reset of the client state to $(0, 0)$. Hence, the possible state transitions from $s_n(t)$ to $s_n(t+1)$ are

$$(x_n(t+1), y_n(t+1)) = \begin{cases} (x_n(t), y_n(t)), & \text{w.p } (1-\alpha)P_{n,1}(a_n(t)), \\ (\min\{x_n(t)+1, L\}, y_n(t)), & \text{w.p } (1-\alpha)P_{n,2}(a_n(t)), \\ ((x_n(t)-1)^+, \min\{y'_n(t), M\}), & \text{w.p } (1-\alpha)P_{n,3}(a_n(t)), \\ (0, 0), & \text{w.p } \alpha, \end{cases}$$

where $x^+ = \max(x, 0)$, $y'_n(t) = y_n(t) + \mathbb{1}\{x_n(t) = 1\}$, $P_{n,1}(a) = 1 - P_{n,2}(a) - P_{n,3}(a)$, $P_{n,2}(a) = \mu_n(a)(1-\beta_n)$, $P_{n,3}(a) = (1-\mu_n(a))\beta_n$, for $a \in \{1, 2\}$, and for every n. The state transition probability of the n-th client is denoted by $p_n(s'|s, a)$. The starting state distribution of the n-th client is denoted by ρ_n. Note that we can include client-specific transition probabilities based on their individual channels without changing the transition structure. These transitions indicate the evolution of media buffer and the stall states based on the probabilities obtained by the bernoulli parameters $\mu_n(a)$ and β.

Notation: Since our state space is discrete and two-dimensional, we assume the following algebraic operations on the state space. Let $\mathbf{0} \triangleq (0, 0)$, unit vectors $e_x \triangleq (1, 0)$ and $e_y \triangleq (0, 1)$. For a a given state $s = (x, y) \in \mathcal{S}$ and any integer $k \in \mathbb{N}_0$:

(1) $s + ke_x \triangleq (\min\{x + k, L\}, y)$
(2) $s - ke_x \triangleq ((x - k)^+, y + \mathbb{1}_{\{x=k\}})$.

Note that the operations above are not associative. For example, $(s - e_x) + e_x \neq s + (-e_x + e_x)$. This is due to the fact that the number of stalls can increase in a state transition.

Reward Structure: The client's quality of experience (QoE) depends upon the occurrence of stall events, their duration, and the buffer state. A *stall period* begins at time t if the media buffer becomes empty at t. A simple analytically tractable structure for the instantaneous cost at time $t+1$ takes the form $c(s_n(t), s_n(t+1))$ where the instantaneous media buffer evolution from $x_n(t)$ to $x_n(t+1)$ captures whether a stall is ongoing, and the stall count $y_n(t+1)$ allows us to pick the cost function based on how many stalls have occurred thus far.

Note that we choose to use a cost function, rather than a QoE-based reward function for consistency with a generic MDP/RL

approach that focuses on cost minimization. It is easy to translate the instantaneous cost to an instantaneous reward by simply multiplying by -1 and maximizing. As pointed out above, we can easily include the video bitrate (corresponding to the video resolution) in this cost function but choose not to do so for simplicity.

We can model a variety of QoE metrics using this approach, such as Delivery Quality Score (DQS) [34], generalized DQS [8], and Time-Varying QoE (TV-QoE) [10] or the ITU Standard P.1203 [24]. The models account for both stall events and video bitrate adaptation and have been validated by laboratory human studies, and we use them to to gain insight into the nature of an appropriate cost function $c(s_n(t), s_n(t+1))$. In all these models, the client starts the video with a QoE of 5. It decreases when the stall event happens, and further decreases when the stall period continues. The QoE recovers when the stall period ends. The recovery is slow with the increase in the number of stall events the client experiences.

Motivated by these models, we make the following assumptions about the instantaneous cost function $c(\cdot,\cdot)$.

Assumption 1. *$c(\cdot,\cdot)$ satisfies the following conditions:*

- **A1.** *The cost increases with the number of stalls i.e., $c(s, s \pm ke_x) > c(s', s' \pm ke_x)$ for $s = (x, y)$, $s' = (x, y')$, such that $y > y'$ and $k \in \{0, 1\}$ for all x.*
- **A2.** *The cost remains constant during a play period i.e. $c(s, s \pm ke_x) = c(s', s' \pm ke_x)$ for $s = (x, y)$, $s' = (x', y)$, such that $k \in \{0, 1\}$ for all $x > 1, x' > 1$.*
- **A3.** *The cost $c(s, 0) = c(s', 0)$ for all s, s' i.e. the cost of client under no stalls is same for all the states.*

Constrained Markov Decision Process: Our resource allocation problem takes the form of the constrained infinite horizon discounted optimization

$$\min_{\pi\in\Pi} \mathbb{E}^{\pi}\left[\sum_{n=1}^{N}\sum_{t=0}^{\infty}\gamma^t c(s_n(t), s_n(t+1))\right]$$
$$\text{s.t} \quad \sum_{n=1}^{N} g(a_n(t)) \le K, \quad \forall t,$$

where $\gamma < 1$ is the discount factor, and $g(a_n(t)) = \mathbb{1}\{a_n(t) = 1\}$. Note that the constraint enforces the condition that only K clients may be allowed high priority service at a given time.

Relaxed CMDP: A commonly used approach to solving CMDPs, starting with seminal work by Whittle [33] is to relax hard constraints and making them hold in expectation to obtain a near-optimal policy. The hard constraint can then be re-enforced by an indexing approach wherein the actions leading to the highest value are chosen, while subjected to the constraint. We follow this approach to soften the constraint to obtain the following problem:

$$\begin{aligned}&\min_{\pi\in\Pi} \mathbb{E}^{\pi}\left[\sum_{n=1}^{N}\sum_{t=0}^{\infty}\gamma^t c(s_n(t), s_n(t+1))\right]\\ &\text{s.t} \quad \mathbb{E}^{\pi}\left[\sum_{n=1}^{N}\sum_{t=0}^{\infty}\gamma^t g(a_n(t))\right] \le \bar{K},\end{aligned} \tag{1}$$

where, $\bar{K} = K/(1-\gamma)$, $a(t) \sim \Pi(s(t))$, $a(t) = (a_n(t))_{n=1}^N$, $s(t) = (s_n(t))_{n=1}^N$. The constraint is now the discounted total number of times the high priority service is allocated to all the clients. Notice that the policy here depends on the joint state of the system as a whole, and a centralized controller is needed to compute the policy, adding to the complexity of the problem. However, under the assumption of strict feasibility of problem (1) we can find an approximately optimal policy.

Assumption 2 (Slater's condition)**.** *There exists a $\xi > 0$ and a policy $\bar{\pi} \in \Pi$ such that $\xi \le \bar{K} - \mathbb{E}^{\bar{\pi}}[\sum_{n=1}^{N}\sum_{t=0}^{\infty}\gamma^t g(a_n(t))]$.*

An immediate consequence of Assumption (2) is that there exists a primal-dual pair (π^*, λ^*) achieving strong duality. For the ease of notation, we denote the individual value and discounted constraint functions of client n under policy π by,

$$J_c^{\pi}(\rho_n) \triangleq \mathbb{E}_{s_n(0)\sim\rho_n}[\mathbb{E}^{\pi}(\sum_{t=0}^{\infty}\gamma^t c(s_n(t), s_n(t+1))|s_n(0))], \tag{2}$$

$$J_g^{\pi}(\rho_n) \triangleq \mathbb{E}_{s_n(0)\sim\rho_n}[\mathbb{E}^{\pi}(\sum_{t=0}^{\infty}\gamma^t g(a_n(t))|s_n(0))]. \tag{3}$$

Consider the associated max-min formulation of problem (1),

$$\max_{\lambda\in\mathbb{R}_+}\min_{\pi\in\Pi}\sum_{n=1}^{N} J^{\pi}(\rho_n;\lambda) \tag{4}$$

where $J^{\pi}(\rho_n;\lambda) \triangleq J_c^{\pi}(\rho_n) + \lambda(J_g^{\pi}(\rho_n) - \frac{\bar{K}}{N})$, λ is the non-negative dual variable, and the associated dual function is given by $D(\lambda) \triangleq \min_{\pi\in\Pi}\sum_{n=1}^{N} J^{\pi}(\rho_n;\lambda)$. Let λ^* be the optimal dual-variable i.e. $\lambda^* = \arg\max_{\lambda\in\mathbb{R}_+} D(\lambda)$, and π^* be an optimal solution to problem (1). $J^{\pi}(\rho_n;\lambda)$ can be interpreted as the Lagrangian value function for a given client n and the equivalence is exact when $N = 1$.

Lemma 1. *Given that assumption (2) holds true, then, $D(\lambda^*) = \sum_{n=1}^{N} J_c^{\pi^*}(\rho_n)$*

However, finding the optimal joint policy even for a fixed λ in the dual function can be computationally hard because the complexity increases exponentially with the number of users. It can be shown that the joint problem can be decomposed in to N smaller problems [28].

Let $\pi_n : \mathcal{S}_n \times \mathcal{A}_n \to [0, 1]$ be the policy of client n where the actions $a_n(t)$ at each time step t, are taken independently of the state of clients other than n. Denoting the optimal policy for the Lagrangian value function of a given client n and λ by,

$$\pi_n^*(\lambda) \triangleq \arg\min_{\pi_n} J^{\pi_n}(\rho_n;\lambda) \tag{5}$$

Theorem 1. *The joint randomized policy $\pi^* = \otimes\pi_n^*(\lambda^*)$ obtained by individually optimizing each client's Lagrangian value function $J^{\pi}(\rho_n;\lambda)$ for the optimal dual variable λ^* is indeed optimal for the original centralized CMDP problem (1) i.e., $\sum_{n=1}^{N} J_c^{\pi^*}(\rho_n) = \sum_{n=1}^{N} J_c^{\pi_n^*(\lambda^*)}(\rho_n)$*

The proof is similar to that of [28, Theorem 1], for completeness, we provide the details in our full technical report [5].

Note that Theorem (1) states that it is sufficient to look at decentralized class of policies of the form $\pi = \otimes\pi_n$.

3 STRUCTURE OF THE OPTIMAL POLICY

We begin our analysis by showing two important properties of the optimal policy and value function. In particular, we show that the optimal policy has a threshold structure. In the next section, we will exploit these structural properties to show that our proposed RL algorithm will provably converge to the globally optimal policy.We first show that the optimal policy for each client has a simple threshold structure, following an approach similar to [28]. We present this result as a theorem below.

Theorem 2. *The optimal policy π_n^* for any single client $n \in [N]$ has a threshold structure. More precisely, for a given state $s = (x, y)$, there exists a threshold function $f^*(\cdot)$ such that*

$$\pi_n^*(s,a) = \begin{cases} \mathbb{1}\{x \le f^*(y)\}, & \text{if } a = 1, \\ \mathbb{1}\{x > f^*(y)\}, & \text{if } a = 2. \end{cases}$$

We will use the following results to prove the above theorem. Since the transition function of all the clients have the same properties (even though they may not be identical), we drop the the client index n for ease of notation, and denote the corresponding optimal value function for state s, and parameter λ by $J^*(s;\lambda)$.

Lemma 2. *Suppose $s \triangleq (x+1, y)$. Then, $J^*(s;\lambda) - J^*(s - e_x;\lambda)$ is non-decreasing in x, for any given λ, and y.*

Lemma 3. *The optimal action obtained in state $s \triangleq (x, y)$ is 2, if and only if, $(1-\beta)[J^*(s+e_x;\lambda)-J^*(s;\lambda)]+\beta[J^*(s;\lambda)-J^*(s-e_x;\lambda)] \ge r$, where $r = c_0 - \frac{\lambda}{\gamma(1-\alpha)(\mu_1-\mu_2)}$, and $c_0 = \frac{1}{\gamma}((1-\beta)[c(s,s) - c(s, s+e_x)] + \beta[c(s, s-e_x) - c(s,s)])$.*

The proofs of Lemma 2 and Lemma 3 are given in [5]. We now present the proof of Theorem 2.

Proof of Theorem 2. From Lemma 3, we have that the optimal action in state $s \triangleq (x, y)$ is 2 iff $(1-\beta)[J^*(s+e_x;\lambda) - J^*(s;\lambda)] + \beta[J^*(s;\lambda)-J^*(s-e_x;\lambda)] \ge r$, where r is defined as above. Moreover, Lemma 2 establishes the non-decreasing property of value function differences, for each stall y. Together, these imply that for a given stall count y, there exists a buffer level $f^*(y)$ such that if the above inequality is satisfied for the first time when the buffer level $x = f^*(y)$, then this inequality is satisfied for all $x > f^*(y)$, and not satisfied for the buffer levels $x \le f^*(y)$. Hence, the optimal policy is to take action 1 when buffer level is below $f^*(y)$, and to take action 2 when the buffer level is above $f^*(y)$, implying that the optimal policy is of threshold type with threshold $f^*(y)$. □

Remark: It is straightforward to show that increasing the number of actions commensurately increases the number of thresholds, following the same monotonicity argument as in Lemma 2. The ability to accommodate multiple thresholds highlights the robustness and flexibility of our approach.

4 RL FOR LEARNING THE OPTIMAL THRESHOLD

In this section, we develop a primal-dual algorithm to solve the constrained Markov decision process problem, and provide results for its convergence to the globally optimal policy. Our implementation of the proposed algorithm follows the methodology of actor-critic algorithms with function approximation. To begin with, we slightly modify our hard-threshold policy to a parameterized soft-threshold policy. For the soft-threshold parametrization, the probability of taking action a in state s corresponding to a threshold parameter vector θ, $\pi_\theta(s,a)$ is defined as

$$\pi_\theta(s,a) = \begin{cases} 1 - \frac{1}{1+e^{\theta(s)}} & \text{if } a = 1, \\ \frac{1}{1+e^{\theta(s)}} & \text{if } a = 2, \end{cases}$$

where $\theta : [L] \times [M] \to \mathbb{R}$.

Note that there exists a one-to-one correspondence between soft-threshold policy parametrization and the hard-threshold function. That is, given a threshold parameter θ, there exists a unique f under the linear relation, $\theta(s) = f(y) - x,\ s = (x, y) \in [L] \times [M]$.

In the parameterized setting, we denote the respective value function and the discounted constraint functions by

$$J_c^{\pi_{\theta_n}}(\rho_n) \triangleq \mathbb{E}_{s_n(0)\sim\rho_n}[\mathbb{E}^{\pi_{\theta_n}}(\sum_{t=0}^{\infty}\gamma^t c(s_n(t), s_n(t+1)|s_n(0))] \quad (6)$$

$$J_g^{\pi_{\theta_n}}(\rho_n) \triangleq \mathbb{E}_{s_n(0)\sim\rho_n}[\mathbb{E}^{\pi_{\theta_n}}(\sum_{t=0}^{\infty}\gamma^t g(a_n(t)))|s_n(0))] \quad (7)$$

Similarly, the Lagrangian value function for a given client n with initial state distribution ρ_n, and λ is given by

$$J^{\pi_{\theta_n}}(\rho_n;\lambda) \triangleq J_c^{\pi_{\theta_n}}(\rho_n) + \lambda\left(J_g^{\pi_{\theta_n}}(\rho_n) - \frac{\bar{K}}{N}\right). \quad (8)$$

It is instructive to view $J^{\pi_{\theta_n}}(\rho_n;\lambda)$ as the value function with the cost function $\bar{c}(s_n(t), s_n(t+1)) \triangleq c(s_n(t), s_n(t+1)) + \lambda(g(a_n(t)) - K/N)$. Therefore, we denote the corresponding advantage functions for $J^{\pi_{\theta_n}}(\rho_n;\lambda)$, $J_c^{\pi_{\theta_n}}(\rho_n)$ and $J_g^{\pi_{\theta_n}}(\rho_n)$ by $A_\lambda^{\pi_{\theta_n}}(s,a)$,$A_c^{\pi_{\theta_n}}(s,a)$, and $A_g^{\pi_{\theta_n}}(s,a)$ respectively. It is easy to show that for any state-action pair (s,a), the following relation holds true: $A_\lambda^{\pi_{\theta_n}}(s,a) = A_c^{\pi_{\theta_n}}(s,a) + \lambda A_g^{\pi_{\theta_n}}(s,a)$.

The above parametrization enables us to use policy gradient methods. In particular, our approach to prove convergence is inspired by the framework of the Natural Policy Gradient (NPG) method for Constrained Markov Decision Processes [7]. We point out that we consider a CMDP with multi-dimensional state, for which we develop a soft-threshold policy class and a corresponding NPG algorithm with finite time global convergence guarantees. None of the above applies to [7], which develops a NPG algorithm for the generic soft-max policy class, which has higher complexity and slower learning rate in our resource allocation regime.

4.1 Convergence of NPG for Threshold Parametrization

Consider the following primal-dual method to solve the minimization problem (1).

$$\theta_{n,t+1} = \theta_{n,t} - \eta_1 F(\theta_{n,t})^\dagger \nabla_{\theta_n} J^{\pi_{\theta_{n,t}}}(\rho_n;\lambda_t)$$
$$\lambda_{t+1} = \mathcal{P}_\Lambda\left(\lambda_t + \eta_2 \sum_{n=1}^{N} \nabla_\lambda J^{\pi_{\theta_{n,t}}}(\rho_n;\lambda_t)\right), \quad (9)$$

where $\theta_{n,t}$ is the threshold parameter of client n at time t, $F(\theta) \triangleq \mathbb{E}_{(s,a)\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s)(\nabla_\theta \log \pi_\theta(a|s))^\top]$ is the Fisher information matrix induced by the policy π_θ, $A^\dagger$ is the Moore-Penrose

inverse of matrix A, and $\mathcal{P}_\Lambda$ is the projection operator onto the interval Λ. η_1 and η_2 are constant step-sizes.

Equation (9) can be obtained directly from the standard primal-dual approach to solve the saddle point problem (4). At each iteration $t+1$, the primal threshold parameters of client n follow an NPG update independent of the others. The dual variable update at each iteration, however, uses the sum of gradients from each client n. Our approach reduces to the standard primal-dual method when $N = 1$. The following lemma shows that the soft-threshold parametrization has a rather simple update rule under the natural policy gradient method.

Denoting actions 1, 2 by a_1 and a_2, respectively,

Lemma 4. *For the soft-threshold policy parametrization, the following are equivalent:*

(1) $\theta_{n,t+1} = \theta_{n,t} - \eta_1 F(\theta_{n,t})^\dagger \nabla_\theta J^{\pi_{\theta_{n,t}}}(\rho_n; \lambda_t)$

(2) $\theta_{n,t+1}(s) = \theta_{n,t}(s) - \frac{\eta_1}{(1-\gamma)}[A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s, a_1) - A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s, a_2)]$.

Proof. For the soft-threshold policy class,

$$\frac{\partial \log \pi_\theta(s,a)}{\partial \theta_{s'}} = \mathbb{1}\{s' = s\}(\mathbb{1}\{a = a_1\} - \mathbb{1}\{a = a_2\})(1 - \pi_\theta(s,a)). \tag{10}$$

Therefore,

$$\pi_\theta(s,a_1)\nabla_\theta \log \pi_\theta(s,a_1) + \pi_\theta(s,a_2)\nabla_\theta \log \pi_\theta(s,a_2) = 0, \tag{11}$$

for any s, and

$$\nabla_\theta \log \pi_\theta(s,a_1) - \nabla_\theta \log \pi_\theta(s,a_2) = e_s, \tag{12}$$

where $e_s \in \mathbb{R}^S$ such that the $i-$th entry is 1 if $i = s$ and 0 otherwise.

We now consider the following quadratic minimization problem:

$$\min_{w \in \mathbb{R}^{|\mathcal{S}|}} \mathbb{E}_{s\sim d_{\rho n}^{\pi_{\theta_{n,t}}}, a\sim\pi_{\theta_{n,t}}(s,\cdot)}[(A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a) - w \cdot \nabla_\theta \log \pi_{\theta_t}(s,a))^2]. \tag{13}$$

The optimal solution to the above minimization problem is given by $w^* = (1-\gamma)F(\theta_{n,t})^\dagger \nabla_\theta J^{\pi_{\theta_{n,t}}}(\rho_n; \lambda_t)$. Notice that the direction of primal update in equations (9) parallels this solution. Hence,

$$\begin{aligned}
&\mathbb{E}_{s\sim d_{\rho n}, a\sim\pi_{\theta_{n,t}}(s,\cdot)}^{\pi_{\theta_{n,t}}}[(A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a) - w \cdot \nabla_\theta \log \pi_{\theta_{n,t}}(s,a))^2] \\
&= \mathbb{E}_{s\sim d_{\rho n}}^{\pi_{\theta_{n,t}}}[\pi_{\theta_{n,t}}(s,a_1)(A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_1))^2 + \pi_{\theta_{n,t}}(s,a_2)(A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_2))^2 \\
&\quad - 2((A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_1) - A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_2)) \\
&\quad * \pi_{\theta_{n,t}}(s,a_1) w \cdot \nabla_\theta \log \pi_{\theta_{n,t}}(s,a_1)) \\
&\quad + \pi_{\theta_{n,t}}(s,a_1) w \cdot \nabla_\theta \log \pi_{\theta_{n,t}}(s,a_1)(w \cdot e_s)] \\
&= \mathbb{E}_{s\sim d_{\rho n}}^{\pi_{\theta_{n,t}}}[\pi_{\theta_{n,t}}(s,a_1)(A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_1))^2 + \pi_{\theta_{n,t}}(s,a_2)(A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_2))^2] \\
&\quad + \mathbb{E}_{s\sim d_{\rho n}}^{\pi_{\theta_{n,t}}}[\pi_{\theta_{n,t}}(s,a_1)\pi_{\theta_{n,t}}(s,a_2) w \cdot e_s \\
&\quad (w \cdot e_s - 2(A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_1) - A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_2)))]
\end{aligned}$$

The first equality follows from expanding the quadratic expansion and (11), and the last equality from (12). It is easy to see that the optimal solution w^* is also the optimal minimizer for the second term in the last equality. Using the fact that the stochastic policy $\pi_{\theta_{n,t}}(s,a) > 0$ for all state-action pairs, we have $w^* \cdot e_s = (A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_1) - A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_2))$. □

An immediate consequence of Lemma 4 is the following form for the policy updates.

Corollary 1. *The policy iterates for algorithm* (9) *are as follows:*

$$\pi_{\theta_{n,t+1}}(s,a) = \pi_{\theta_{n,t}}(s,a)\frac{\exp(\frac{-\eta_1}{1-\gamma}A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a))}{Z_t(s)}, \tag{14}$$

$$\text{where } Z_t(s) \triangleq \sum_{a\in\mathcal{A}} \pi_{\theta_{n,t}}(s,a)\exp(\frac{-\eta_1}{1-\gamma}A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a)). \tag{15}$$

The proof of Corollary 1 is given in [5].

The equivalent primal-dual method to algorithm (9) is given by

$$\theta_{n,t+1}(s) = \theta_{n,t}(s) - \frac{\eta_1}{(1-\gamma)}[A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_1) - A_{\lambda_t}^{\pi_{\theta_{n,t}}}(s,a_2)],$$

$$\lambda_{t+1} = \mathcal{P}_\Lambda(\lambda_t + \eta_2(\sum_{n=1}^N J_g^{\pi_{\theta_{n,t}}}(\rho_n) - \bar{K})). \tag{16}$$

We have the following convergence guarantee for algorithm (16).

Theorem 3. *Let* $\Lambda = [0, \frac{2N}{(1-\gamma)\xi}]$, $\rho_n \in \triangle_{\mathcal{S}_n}$, $\theta_{n,0} = \mathbf{0}$, *be the starting state distribution and threshold parameter initialization of client n, and* $\lambda_0 = 0$, *be the initial dual variable. For the particular choice of* $\eta_1 = \log|\mathcal{A}|$, $\eta_2 = \frac{1-\gamma}{N\sqrt{T}}$, *the iterates generated by the algorithm* (9) *satisfy*

$$\frac{1}{T}\sum_{t=0}^{T-1}\sum_{n=1}^{N}(J_c^{\pi_{\theta_{n,t}}}(\rho_n) - J_c^{\pi_n^*}(\rho_n)) \le \frac{4N}{(1-\gamma)^2\sqrt{T}}. \tag{17}$$

$$\left(\frac{1}{T}\sum_{t=0}^{T-1}\sum_{n=1}^{N} J_g^{\pi_{\theta_{n,t}}}(\rho) - \bar{K}\right)^+ \le \frac{(2/\xi + 4\xi)N^2}{(1-\gamma)^2\sqrt{T}}. \tag{18}$$

The proof sketch is as follows. We first show that the NPG method has a simple parameter update for the threshold policy class. We then prove that the average cost function generated by the iterates of the algorithm converges to the global optimal value, and bound the constraint violation. The details are provided in [5].

Note that the quadratic scaling of constraint error with the number of users N arises due the resources being fixed. If $\bar{K}$ scales linearly with N, the constraint error will also be linear in N.

Remark: The notion of soft-threshold parametrization coupled with natural policy gradient is motivated by constrained RL using soft-max parametrization with natural policy gradient [7]. However, the convergence analysis is not straightforward, since the structure of Fischer-information matrix induced by the soft-threshold policy is different, which translates to different primal dual updates.

4.2 Algorithm Design

Algorithm 1 summarizes our approach (9) in accordance with the actor-critic approaches. We iteratively update the value function (and hence the advantage function), the policy parameter, and the Lagrange multiplier. At each time step, we run one actor-critic update for each client. In particular, for each client $n \in N$, we sample an action according to the current policy of the client to update the value function of the current state $J_{n,t+1}(s_n(t))$ from which the advantage function for each action $a \in \mathcal{A}$, $A_{n,t}(s_n(t), a)$ is computed. The threshold parameter for the current state is updated according to (9). The value functions and parameters of all but the current state remain unchanged. After each client updates its value

function and its policy, the experienced actions are retrieved, and the Lagrange multiplier is updated using the constraint formulation.

Remark 1. We update the threshold parameter with just the advantage of action a_1 instead of the difference between advantages. This variant of the threshold update does not change the direction of the gradient. In the case of inexact gradients, assuming a δ precision introduces a residual error but still ensures finite time convergence, similar to exact gradients [23]. Our work guarantees global convergence in finite time, unlike existing results that only demonstrate asymptotic convergence.

Algorithm 1 Threshold Natural Policy Gradient Algorithm

Initialize Set threshold $\theta_{n,0} = 0$, starting state $s_n(0) \sim \rho_n$ for every client $n \in N$, $\eta_1 = \ln 2$, $\eta_2 = \frac{1-\gamma}{N\sqrt{T}}$, and $\lambda_0 = 0$. Set state visitation count $\eta(n,s) = 1 \ \forall s \in \mathcal{S}_n$ and for each client $n \in N$.

for $t = 0, 1, \ldots,$ **do**
  for Each client $n = 0, 1, \ldots, N$ **do**
    Update the count $\eta(n, s_n(t)) \leftarrow \eta(n, s_n(t)) + 1$.
    Take the action $a_n(t) \sim \pi_{\theta_{n,t}}(s_n(t), .)$.
    Obtain the next state $s_n(t+1)$, and cost $c(s_n(t), s_n(t+1))$, $g(a_n(t))$ from the environment.
    Update the value function according to

$$J_{n,t+1}(s_n(t)) = J_{n,t}(s_n(t)) + \frac{1}{\eta(n, s_n(t))} \left[\lambda_t g(a_n(t)) + \left[c(s_n(t), s_n(t+1)) + \gamma J_{n,t}(s_n(t+1))\right] - J_{n,t}(s_n(t))\right],$$

$$J_{n,t+1}(s') = J_{n,t}(s'), \forall s' \neq s_n(t).$$

    Update the threshold parameter

$$\theta_{n,t+1}(s_n(t)) \leftarrow \theta_{n,t}(s_n(t)) - \frac{\eta_1}{(1-\gamma)}[A_{n,t}(s_n(t), a_1) - A_{n,t}(s_n(t), a_2)],$$

$$\theta_{n,t+1}(s) = \theta_{n,t}(s), \forall s \neq s_n(t).$$

  end for
  Update the Lagrange multiplier $\lambda_{t+1} = \mathcal{P}_\Lambda(\lambda_t + \eta_2[\sum_{n=1}^{N} g(a_n(t)) - \bar{K}])$.
end for

5 SIMULATION-BASED TRAINING

We first train our system by developing a simulator that captures the dynamics of the system shown in Figure 1. The simulator consists of a wireless access point (AP) with two service classes, each has a fixed bandwidth (e.g., 12 Mbps for "high", and 4 Mbps for "low") and N clients that desire service, with our typical setting being 6 or fewer clients. An intelligent controller assigns clients to one of the service classes, and the available bandwidth of that class is partitioned equally among the clients assigned to it, with the realized bandwidth being drawn from a normal distribution around the mean value to model environmental randomness. For instance, if we set a constraint of at most 2 occupants of the "high" service class, each would get a mean throughput of 6 Mbps per-client in the example. Each client has a state (x, y), consisting of the video buffer length and the number of stalls. The QoE of the client is calculated using the DQS model described in Section 2. Note that we use "reward" using the QoE, rather than "cost" as used in the analytical model, since most RL implementations use this approach (i.e., we multiply "cost" by -1 and maximize instead of minimizing).

The state of the system as a whole is the union of the states of all 6 clients. The action is the service assignment of each client. Hence, there are a total of 2^6 possible actions (2 for each client). As indicated in Section 2, when the number of clients in each service class is fixed, the system can be treated as N independent single-client systems, each having two service classes with appropriately scaled-down bandwidth. Intuitively, training on this system should be faster, since we obtain N [state, action, next-state, reward] samples per time step here, as opposed to just one in the joint system case. Furthermore, solving using the Dual approach introduces a Lagrange multiplier λ, which also has to be learned by performing a hyperparameter search to ensure that the number of clients in each service class satisfies the service class constraints. The QoE model used as reward is taken from the DQS model in [34]. It follows the reward model described in Section 2 closely.

5.1 Algorithms

Vanilla (V): This is a simple policy that merges all available throughput into a single service class and shares it equally. It is equivalent to CSMA-based random access in WiFi, or a simple round-robin scheduler in cellular. We will use this as a base policy in the real-world system.

Greedy Buffer (GB): This algorithm awards high priority service to the clients with the lowest video buffer state, subject to resource constraints. The algorithm follows the general theme of max-weight [30] and min-deficit [13] being throughput or timely-throughput optimal in queueing systems. However, it does not account for the dependence of QoE on stall count. We use it as a well-established algorithm with good performance in the real-world evaluation.

Centralized Hard PPO (CH): This algorithm represents recent efforts at applying off-the-shelf RL techniques in media streaming applications, such as those presented in [3, 19]. In particular, QFlow [3] shows that unstructured Deep Q-Learning (DQN) performs really well in a context much like ours but requires a long training duration. Policy gradient algorithms have much better empirical performance than DQN, and so we pick Proximal Policy Optimization (PPO) [27]. We implement it with a hard constraint of 2 clients with "high" service and 4 with "low" service.

Centralized Soft PPO (CS): This algorithm implements the Lagrangian relaxation by adding a penalty λ for accessing "high" service. We use the same PPO algorithm, but do not impose a hard constraint as in CH. Rather, the algorithm is trained using a hyperparameter search over λ such that we have an average of 2 clients in the "high" service class.

Decentralized PPO (DC): This algorithm takes advantage of the conditional independence property described in Section 2 that enables division into N independent systems, along with a Lagrangian relaxation of the constraints. We now use the PPO algorithm on an individual client basis, but since we obtain 6 samples per step, it

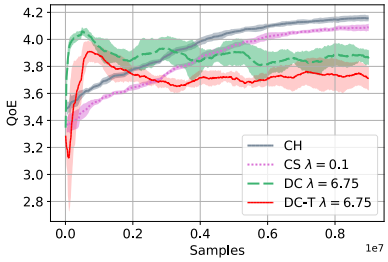

Figure 2: Training in simulation

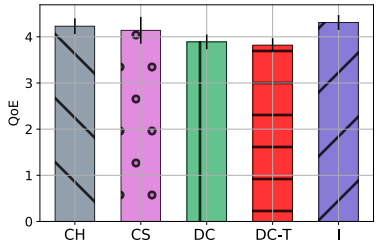

Figure 3: QoE in simulation

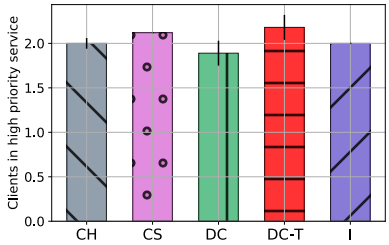

Figure 4: Clients in high priority service

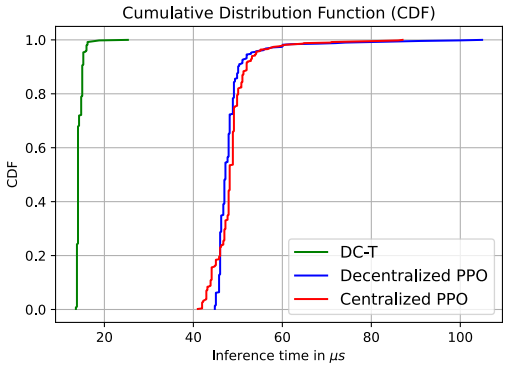

Figure 5: Inference time for DC-T and PPO algorithms for 6 clients

can train much faster. However, we do not impose any structure on the policy class. Again, we need to find the appropriate penalty λ to enforce the constraint.

Decentralized Threshold (DC-T) This algorithm is similar to DC in utilizing division into N systems for faster learning, but also imposes a threshold structure of the optimal algorithm. It requires only one neuron (logistic function) to represent a threshold policy, which makes for simple implementation. Like DC, it too can train faster than the centralized approaches, and also needs an appropriate penalty term to enforce constraints.

Index (I): This algorithm follows the philosophy behind the Whittle Index [33] in converting a soft-constrained into a hard-constrained and robust policy. We order the states of clients based on their values obtained from DC-T, and provide "high" service the two with the highest value. We will see that this can be used regardless of the number of clients, or the quality of their channels.

5.2 Training and Evaluation

Figure 2 shows the evaluation of the algorithms on a joint system during training (averaged over 5 random seeds). We observe that decentralized algorithms converge over four times faster than centralized algorithms as they exploit the structure of the environment. The performance difference between the trained centralized and decentralized algorithms is negligible as seen in Figure 3 (averaged over 100 runs). The performance of DC-T is on par with the best performing algorithms, which confirms our hypothesis that the optimal policy is indeed a threshold policy. The index policy, I that is hard-constrained version of DC-T shows similarly high performance. Figure 4 (averaged over 100 runs) shows the number of clients in the high priority queue during evaluation of the soft constrained algorithms is near 2, and so confirms our choice of λ. Finally, Figure 5 shows the efficacy of the DC-T algorithm in deployment in terms of inference times. We see that the PPO algorithms take roughly 50 to 60 μs, while the DC-T algorithm takes only about 10 to 15 μs for inferring the decisions for the 6 client system.

6 REAL-WORLD EVALUATION

We follow an experimental platform identical to [3] in which we use four Intel NUCs, with three of the NUCs hosting YouTube sessions (2 each) and the last hosting the intelligent controller and an SQL database. We use a pub-sub approach at the database to collect state and reward data and to publish control actions. The client NUCs have a list of popular 1080p YouTube videos, which they randomly sample, play and then flush their buffers. We force the players to 1080p resolution across all sessions for fairness. Relevant data such as the buffer length and number of stalls are gathered in the database and shared with the intelligent controller for decision making. We couple this system with a WiFi access point running OpenWRT, in which we create high and low priority queues using Linux Traffic Controller (TC). The controller uses OpenFlow experimenter messages to communicate its prioritization decisions to the access point every 10 seconds. The communication and computational overheads of obtaining the state information from the clients over the wireless channel and execution of simulation-trained RL policies is minimal due to the low frequency of state updates.

We perform the evaluations below in a laboratory setting with normal background WiFi traffic. We verified using iPerf that the throughput limits that we set on TC are actually attained, i.e., the background WiFi traffic does not significantly affect performance. In each of the settings below, each algorithm was run for a minimum of three hours for data collection. We verified that the collected samples do not show much statistical difference for longer collection periods. We also performed one test in an anechoic chamber, and found no significant variation in system performance.

A. Does structured RL provide high-performance?

We first determine if structured RL approaches are near-optimal. Figure 6 shows the average QoE across three hours of YouTube sessions for all 7 policies. As expected, the centralized policies CH and CS do the best, with the decentralized versions DC, DC-T and I

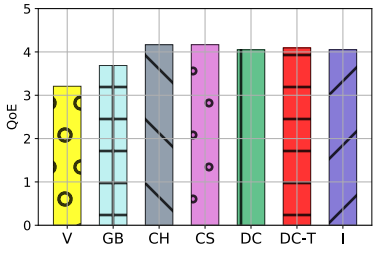

Figure 6: Comparison of average QoE

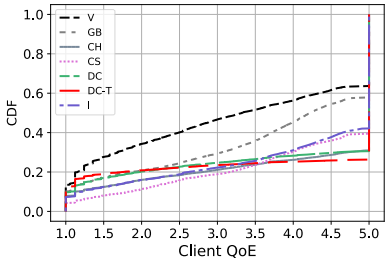

Figure 7: Comparison of QoE CDF

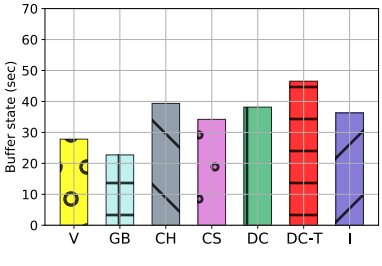

Figure 8: Comparison of average buffer

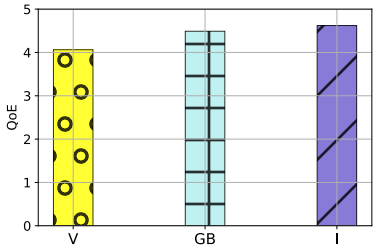

Figure 9: Comparison of average QoE with varying number clients

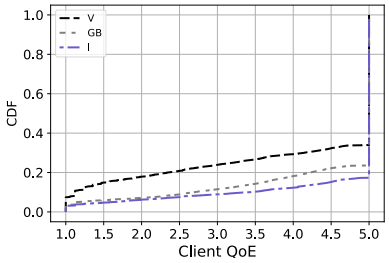

Figure 10: Comparison of QoE CDF with varying number of clients

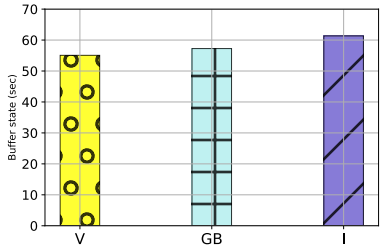

Figure 11: Comparison of average buffer state with varying number of clients

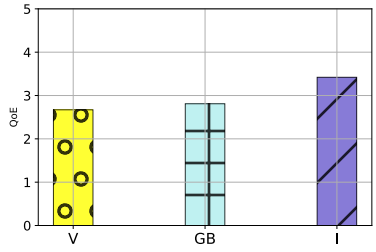

Figure 12: Comparison of average QoE in a poor channel

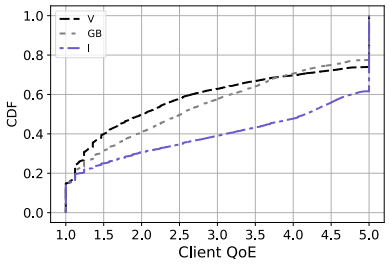

Figure 13: Comparison of QoE CDF in a poor channel

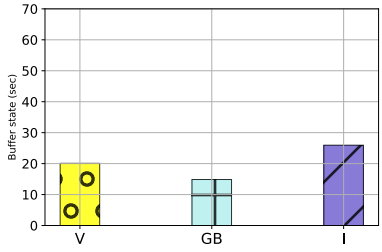

Figure 14: Comparison of average buffer state in a poor channel

following close behind. GB does reasonably well, but is unable to account for stalls influencing QoE, hence causing performance loss. The vanilla approach shows why intelligent control is necessary, as it lags behind the RL approaches by full QoE unit, i.e., it is over 30% worse. Figure 7 shows the CDF of QoE samples, where the value of the RL approach is even clearer from the fact that about 60-70% of samples have a perfect score of QoE 5, while the other approaches are about 20% worse. Finally, Figure 8 shows the average video buffer length, and appears to indicate that Greedy tries to equalize buffers, letting them go too low before prioritizing. The RL-based policies all have higher average buffers, consistent with higher QoE and fewer stalls.

B. Is indexing robust to a variable number of clients?

We next address whether the index policy I can be applied unaltered to the case where the number of clients is less than 6, which means that there is no need to train over a variable number of clients. So we perform an experiment where we decrease number of clients from 6 to 4 over three hours. As expected, the QoE of all three policies increases, with all three policies having an average QoE above 4 in Figure 9. However, the overall trend is maintained as the Index policy does the best as seen in Figures 9, 10, and 11.

C. Does the index approach generalize to variable channels?

Wireless channels are subject to variability due to mobility and obstruction, and the question arises as to whether the RL approach needs to be trained separately over many channel realizations? Since indexing simply orders clients based on their states, we wish to determine if it can be used unchanged even when the channel quality is low. Hence, we introduce packet losses and delays using a network emulator to create channels with disturbances. We first group clients with similar channels into clusters and divide the total available resources in a proportionally fair manner across clusters. We then apply our service differentiation policies across members of each cluster. We show results for a cluster seeing 10% packet loss and 20 ms delay, and, as expected, QoE drops as seen in Figure 12. However, the overall order still holds as Vanilla and Greedy Buffer have QoEs over 20% worse than Index. Further, the client QoE CDF and average buffer state indicate same trend as seen in 13 and 14, with Indexing still performing significantly better than the others.

7 CONCLUSION

In this work, we investigated the optimization of resource allocation at the wireless edge for media streaming, particularly focusing on YouTube sessions, aiming to enhance users' quality of experience (QoE). Through a CMDP framework and a data-driven approach using constrained RL (CRL), we showed a threshold structure in optimal policies and developed a primal-dual natural policy gradient (NPG) algorithm to efficiently learn such threshold policies. Specifically, we showed that soft-threshold parametrization coupled with NPG has a fast convergence rate, and only requires single neuron training. Simulation results demonstrate that our decentralized single-client decomposition learning approximately 4 times faster than centralized learning while maintaining similar performance, while inference is completed in about 4 times less time. We implemented learned policies on an intelligent controller platform, achieving over 30% QoE improvement compared to a vanilla policy. The proposed index policy also proved robust under varying load and channel conditions while maintaining high QoE levels.

Limitations: The proposed approach depends on simulation environments, so adjustments based on specific streaming applications, QoE metrics, and physical environment factors are necessary. To improve practical relevance of our work in enhancing user experience, further exploration for the scalability and deployment of the intelligent controller in large-scale networks is needed.

8 ACKNOWLEDGEMENT

This work was supported in part by the grants NSF Grants CNS 2312978, ECCS 2030245, CNS 2312979, CNS 2312978 and ARO grant W911NF-19-1-0367. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International Conference on Machine Learning*. PMLR, 22–31.

[2] Eitan Altman. 1999. *Constrained Markov decision processes*. Vol. 7. CRC Press.

[3] Rajarshi Bhattacharyya, Archana Bura, Desik Rengarajan, Mason Rumuly, Srinivas Shakkottai, Dileep Kalathil, Ricky KP Mok, and Amogh Dhamdhere. 2019. QFlow: A reinforcement learning approach to high qoe video streaming over wireless networks. In *Proceedings of the twentieth ACM international symposium on mobile ad hoc networking and computing*. 251–260.

[4] Vivek S Borkar. 2009. *Stochastic approximation: a dynamical systems viewpoint*. Vol. 48. Springer.

[5] Archana Bura, Sarat Chandra Bobbili, Shreyas Rameshkumar, Desik Rengarajan, Dileep Kalathil, and Srinivas Shakkottai. 2024. Structured Reinforcement Learning for Media Streaming at the Wireless Edge. *arXiv preprint arXiv:2404.07315* (2024).

[6] Archana Bura, Aria HasanzadeZonuzy, Dileep Kalathil, Srinivas Shakkottai, and Jean-Francois Chamberland. 2022. DOPE: Doubly optimistic and pessimistic exploration for safe reinforcement learning. *Advances in neural information processing systems* 35 (2022), 1047–1059.

[7] Dongsheng Ding, Kaiqing Zhang, Tamer Basar, and Mihailo Jovanovic. 2020. Natural policy gradient primal-dual method for constrained markov decision processes. *Advances in Neural Information Processing Systems* 33 (2020), 8378–8390.

[8] N. Eswara, K. Manasa, A. Kommineni, S. Chakraborty, H. P. Sethuram, K. Kuchi, A. Kumar, and S. S. Channappayya. 2017. A Continuous QoE Evaluation Framework for Video Streaming over HTTP. *IEEE Transactions on Circuits and Systems for Video Technology* In press (2017). https://doi.org/10.1109/TCSVT.2017.2742601

[9] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. 2013. Towards Network-wide QoE Fairness Using Openflow-assisted Adaptive Video Streaming. In *Proceedings of ACM FhMN*.

[10] D. Ghadiyaram, J. Pan, and A. C. Bovik. 2018. Learning a Continuous-Time Streaming Video QoE Model. *IEEE Transactions on Image Processing* 27, 5 (May 2018), 2257–2271. https://doi.org/10.1109/TIP.2018.2790347

[11] Aria HasanzadeZonuzy, Archana Bura, Dileep Kalathil, and Srinivas Shakkottai. 2021. Learning with Safety Constraints: Sample Complexity of Reinforcement Learning for Constrained MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 7667–7674.

[12] Aria HasanzadeZonuzy, Dileep M Kalathil, and Srinivas Shakkottai. 2021. Model-Based Reinforcement Learning for Infinite-Horizon Discounted Constrained Markov Decision Processes.. In *IJCAI*. 2519–2525.

[13] I.H. Hou, V. Borkar, and PR Kumar. 2009. A theory of QoS for wireless. In *IEEE INFOCOM 2009*. Rio de Janeiro, Brazil.

[14] Ping-Chun Hsieh and I-Hong Hou. 2018. Heavy-traffic analysis of QoE optimality for on-demand video streams over fading channels. *IEEE/ACM Transactions on Networking* 26, 4 (2018), 1768–1781.

[15] Xinyue Hu, Arnob Ghosh, Xin Liu, Zhi-Li Zhang, and Ness Shroff. 2023. COREL: Constrained Reinforcement Learning for Video Streaming ABR Algorithm Design Over mmWave 5G. In *2023 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*. IEEE, 1–6.

[16] Michael Jarschel, Florian Wamser, Thomas Hohn, Thomas Zinner, and Phuoc Tran-Gia. 2013. SDN-based Application-Aware Networking on the Example of YouTube Video Streaming. In *Proceedings of EWSDN*.

[17] Woo-Hyun Ko, Ushasi Ghosh, Ujwal Dinesha, Raini Wu, Srinivas Shakkottai, and Dinesh Bharadia. 2023. EdgeRIC: Delivering Realtime RAN Intelligence. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 1162–1164.

[18] Tao Liu, Ruida Zhou, Dileep Kalathil, PR Kumar, and Chao Tian. 2021. Learning Policies with Zero or Bounded Constraint Violation for Constrained MDPs. In *Thirty-fifth Conference on Neural Information Processing Systems*.

[19] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 197–210.

[20] Open Networking Foundation. 2021. SD-RAN: Software Defined Radio Access Network. https://opennetworking.org/sd-ran/.

[21] Ali ParandehGheibi, Muriel Médard, Asuman Ozdaglar, and Srinivas Shakkottai. 2011. Avoiding interruptions—A QoE reliability function for streaming media applications. *IEEE Journal on Selected Areas in Communications* 29, 5 (2011), 1064–1074.

[22] Sangeeta Ramakrishnan, Xiaoqing Zhu, Frank Chan, and Kashyap Kambhatla. 2015. SDN Based QoE Optimization for HTTP-Based Adaptive Video Streaming. In *Proceedings of IEEE ISM*.

[23] Julian Rasch and Antonin Chambolle. 2020. Inexact first-order primal–dual algorithms. *Computational Optimization and Applications* 76, 2 (2020), 381–430.

[24] Werner Robitza, Marie-Neige Garcia, and Alexander Raake. 2017. A modular http adaptive streaming QoE model — Candidate for ITU-T P.1203 ("P.NATS"). In *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 1–6.

[25] Arghyadip Roy, Vivek Borkar, Abhay Karandikar, and Prasanna Chaporkar. 2021. Online reinforcement learning of optimal threshold policies for Markov decision processes. *IEEE Trans. Automat. Control* 67, 7 (2021), 3722–3729.

[26] Sandvine. 2023. The Global Internet Phenomena Report. https://www.sandvine.com/resources.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[28] Rahul Singh and PR Kumar. 2019. Optimal decentralized dynamic policies for video streaming over wireless channels. *arXiv preprint arXiv:1902.07418* (2019).

[29] Albert Sunny, Sumankumar Panchal, Nikhil Vidhani, Subhashini Krishnasamy, SVR Anand, Malati Hegde, Joy Kuri, and Anurag Kumar. 2017. A generic controller for managing TCP transfers in IEEE 802.11 infrastructure WLANs. *Journal of Network and Computer Applications* 93 (2017), 13–26.

[30] L. Tassiulas and A. Ephremides. 1992. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. Automat. Control* 37, 12 (Dec. 1992), 1936–1948.

[31] Sharan Vaswani, Lin Yang, and Csaba Szepesvári. 2022. Near-optimal sample complexity bounds for constrained MDPs. *Advances in Neural Information Processing Systems* 35 (2022), 3110–3122.

[32] Honghao Wei, Xin Liu, and Lei Ying. 2022. Triple-Q: A Model-Free Algorithm for Constrained Reinforcement Learning with Sublinear Regret and Zero Constraint Violation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3274–3307.

[33] Peter Whittle. 1988. Restless bandits: Activity allocation in a changing world. *Journal of applied probability* 25, A (1988), 287–298.

[34] H. Yeganeh, R. Kordasiewicz, M. Gallant, D. Ghadiyaram, and A. C. Bovik. 2014. Delivery quality score model for Internet video. In *Proceedings of IEEE ICIP*. https://doi.org/10.1109/ICIP.2014.7025402

[35] Yiming Zhang, Quan Vuong, and Keith Ross. 2020. First Order Constrained Optimization in Policy Space. *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020).