MDPI

Article

Learning-based Online QoE Optimization in Multi-Agent Video Streaming

Yimeng Wang [1], Mridul Agarwal [2], Tian Lan[3], and Vaneet Aggarwal [2,4,5,*]

[1] Google; 08211018@bjtu.edu.cn. He was with the George Washington University when this work was performed.
[2] Purdue University; {agarw180,vaneet}@purdue.edu
[3] George Washington University; tlan@gwu.edu
[4] King Abdulaziz University
[5] King Abdullah University of Science and Technology; vaneet.aggarwal@kaust.edu.sa
* Correspondence: vaneet@purdue.edu

Abstract: Video streaming has become a major usage scenario for the Internet. The growing popularity of new applications, such as 4K and 360-degree videos, mandates that network resources must be carefully apportioned among different users, in order to achieve the optimal Quality of Experience (QoE) and fairness objectives. This results in a challenging online optimization problem, as networks grow increasingly complex and the relevant QoE objectives are often nonlinear functions. Recently, data-driven approaches, deep Reinforcement Learning (RL) in particular, have been successfully applied to network optimization problems, by modeling them as Markov Decision Processes. However, existing RL algorithms involving multiple agents fail to address nonlinear objective functions on different agents' rewards. To this end, we leverage MAPG-finite, a Policy Gradient algorithm designed for multi-agent learning problems with nonlinear objectives. It allows us to optimize bandwidth distributions among multiple agents and to maximize QoE and fairness objectives on video streaming rewards. Implementing the proposed algorithm, we compare MAPG-finite strategy with a number of baselines – including static, adaptive, and single-agent learning policies. The numerical results show that MAPG-finite significantly outperforms the baseline strategies with respect to different objective functions and in various settings, including both constant and adaptive bitrate videos. Specifically, our MAPG-finite algorithm maximizes QoE by 15.27% and maximizes fairness by 22.47% compared to the standard SARSA algorithm for a 2000 KB/s link.

Keywords: Video Streaming; Resource Allocation; Reinforcement Learning; Policy Gradient.

Citation: Wang, Y.; Agarwal, M.; Lan, T.; Aggarwal, V. Learning-based Online QoE Optimization in Multi-Agent Video Streaming. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Video streaming has become a major usage scenario for Internet users, accounting for over 60% of downstream traffic on the internet [1]. The growing popularity of new applications and video formats, such as 4K and 360-degree videos, mandates that network resources must be apportioned among different users in an optimal and fair manner, in order to deliver satisfactory Quality of Experience (QoE). There are many factors impacting the Quality of Experience of the video streaming service. For example, the peak signal-to-noise ratio (PSNR) of the received video [2], or the structural similarity of the image (SSIM) [3]. In particular, the stall time during streaming is a critical performance objective especially for services that require low response time and highly rely on customer experience, e.g., online video streaming and autonomous vehicle networks [4]. Further, the streaming device impacts the bitrate and in turn effects the QoE parameter (see [5] and the citations within).

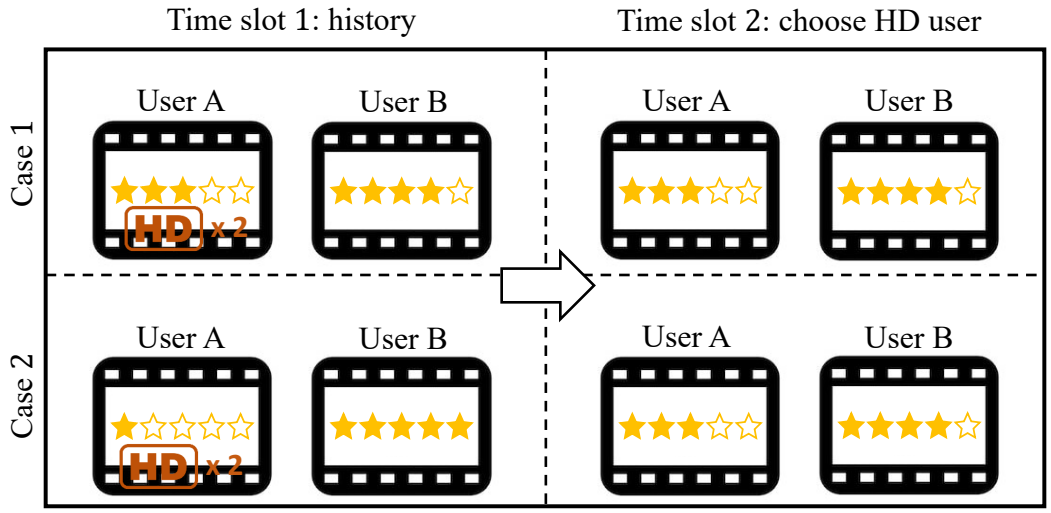
Online optimization of stall time and QoE in a dynamic network environment is a very challenging problem which can be analyzed as optimization problem [6] or learning problem [7]. Traditional optimization-based approaches often rely on precise models to

crystalize the network system and the underlying optimization problems. For instance, the authors in [8] construct the QoE aware utility functions using a two-term power series model, while the authors in [9] leverage both "bSoft probe" and "Demokritos probe" to model the QoE measurement, by analyzing the weight factors and exponents of all video-straming service parameters, as well as quantifying the "Decodable Frame Rate" of three different types of frames. However, these model-based approaches cannot solve the online QoE optimization with incomplete or little knowledge about future system dynamics.

Recently, Reinforcement Learning (RL) has been proven as an effective strategy in solving many online network optimization problems that may not yield a straightforward analytical structure, such as wireless sensor networks (WSN) routing [10], vehicle networks spectrum sharing [11], data caching [7], and network service placement [12]. In particular, deep RL employs neural networks to estimate a decision-making policy, which self-improves based on collected experiential data to maximize the rewards. Compared with traditional model-based decision making strategies, deep RL has a number of benefits: (i) It does not require a complete mathematical model or analytical formulation that may not be available in many complex practical problems; (ii) The use of deep neural networks as function approximators makes the RL algorithms easily extensible to problems with large state spaces; and (iii) It is capable of achieving fast convergence in online decision making and dynamic environments that evolve over time.

The goal of this paper is to develop a new family of multi-agent reinforcement learning algorithms to apportion download bandwidth on the fly among different users and to optimize QoE and fairness objectives in video streaming. We note that existing RL algorithms often focus on maximizing the sum of future (discounted) rewards across all agents and fail to address inter-agent utility optimization, aiming at balancing the discounted reward received by each individual agents. Such inter-agent utility optimizations are widely considered in video streaming problems, e.g., to optimize the fairness of network resource allocation and to maximize a non-linear QoE function of individual agent's performance metrics. More precisely, in a dynamic setting, the problem being solved *must* be modeled as an MDP, where agents take actions based on some policy π and observed system states, causing the system to transition to a new state. A reward r_k is fetched for each agent k. The transition probability to the new state is dependent only on the previous state and the action taken in the previous state. RL algorithms aim to find an optimal policy π to maximize the sum of (discounted) rewards$\sum_{t=1}^{\infty} \gamma^t \sum_k r_k(t)$ for all users. However, when QoE and fairness objectives are concerned, a nonlinear function f, such as the fairness utility [13] and sigmoid QoE function reported by [14], must be applied to the rewards received by different agents, resulting in the optimization of a new objective $f(\bar{r}_1, \bar{r}_2, \ldots)$, where $\bar{r}_k = \frac{1}{T}\sum_{t=1}^{T} \gamma^t r_k(t)$ is the average discounted reward for each agent k in a finite time T. It is easy to see that such nonlinear functions will potentially violate the memoryless rule of MDP which is required for RL since the optimization objective now depends on all past rewards/states. In this paper, we will develop a new family of multi-agent reinforcement learning algorithms to optimize such nonlinear objectives for QoE and fairness maximization in video streaming.

We propose *Multi-agent Policy Gradient for Finite Time Horizon* (MAPG-finite) for optimize nonlinear objective functions of cumulative rewards of multiple agents with a finite time horizon. We employ MAPG-finite in online video streaming with the goal of maximizing QoE and fairness objectives by adjusting the download bandwidth distribution. To this end, we quantify stall time for online video streaming with multiple agents, under a shared network link and dynamic video switching by the agents. At the end of the time horizon, a nonlinear function $f(\cdot)$ of the agents' individual cumulative rewards is calculated. The choice of $f(\cdot)$ is able to capture different notions of fairness – e.g., the well-known α-fairness utility [13] that incorporates proportional fairness and max-min fairness as special cases, and the sigmoid-like QoE function reported by [14] which indicates that users with mediocre waiting time tend to be more sensitive than the rest – and thus balances the performance received by different agents in online video streaming. We leverage the



Case	HD at Slot 2	Fairness	Optimal Choice
Case 1	User A	$\ln 12 + \ln 8 = 4.56$	User B HD
	User B	$\ln 9 + \ln 12 = 4.68$	
Case 2	User A	$\ln 8 + \ln 9 = 4.28$	User A HD
	User B	$\ln 5 + \ln 13 = 4.17$	

Figure 1. An illustrative example showing that the optimal choice of bandwidth assignment must depend on past rewards/states, in order to maximize a logarithmic fairness objective.

RL algorithm proposed in [15] to develop a model-free multi-agent reinforcement learning algorithm to cope with the inter-agent fairness reward for multiple users. In particular, this RL algorithm modifies the traditional Policy Gradient to find a proper ascending direction for the nonlinear objective function using random sampling. We prove the convergence of the proposed algorithm to at least a local optimal of the target optimization problem. The proposed multi-agent algorithm is model-free and shown to efficiently solve the QoE and fairness optimization in online video streaming.

To demonstrate the challenge associated with optimizing nonlinear objective functions, consider the example shown in Figure 1. Two users, A and B, share a download link for video streaming. Due to the bandwidth constraint, the link is only able to stream one high-definition (HD) video and one low-definition (LD) video at a time. In each time slot t, the two users' QoE, denoted by $r_A(t)$ and $r_B(t)$, are measured by a simple policy that multiplies the quality of the *content* (from 1 to 5 stars) and the resolution of the video (1 for LD and 2 for HD). The service provider is interested in optimizing a logarithmic utility of the users' aggregate QoE, i.e., $u = \ln(\sum_t r_A(t)) + \ln(\sum_t r_B(t))$ (which corresponds to the notion of maximum proportional fairness [13,16]), for the two time slots $t = \{1, 2\}$. It is easy to see that in Case 1, user A received $r_A(1) = 6$ in time slot 1 and user B $r_B(1) = 4$. If we stream HD to user A and LD to user B in the next time slot, then the total received utility becomes $\ln(6+6) + \ln(4+4) \approx 4.56$, while the opposite assignment achieves a higher utility $\ln(6+3) + \ln(4+8) \approx 4.68$. However, in Case 2, choosing user A in time slot 2 to receive HD and user B LD gives the highest utility of $\ln(2+6) + \ln(5+4) \approx 4.28$. Thus, the optimal decision in time slot 2 depends on the reward received in all past time slots (while we have only shown the rewards in time slot 1 for simplicity in this example). In general, the dependence of utility objective on all past rewards implies a violation of the Markovian property, as the actions should only be affected by the currently observed system state in MDP. This mandates a new family of RL algorithms that are able to cope with nonlinear objective functions, which motives this paper.

We note that in video streaming optimization, the action space for learning-based algorithms can still be prohibitively large, since the bandwidth assignment decisions are

continuous (or at least fine-grained if we sample the continuous decision space), while the action space suitable for RL algorithms should be small as the output layer sizes of neural networks are limited. The action space for bandwidth assignment increases undesirably with both the growing number of users and the increasing amount of network resources in the video streaming system. To overcome this challenge, we propose an adaptive bandwidth adjustment process. It leverages two separate reinforcement learning modules running in parallel, each tasked to select a target user to increase or decrease his current bandwidth by one unit. This effectively reduces the action space of each RL module to exactly the number of users in the system, while both learning modules can be trained in parallel with the same set of data through backpropagation. This technique significantly reduces the action spaces in MAPG-finite and make the optimization problem tractable.

To evaluate the proposed algorithm, we develop a modularized testbed for event-driven simulation of video streaming with multi-agent bandwidth assignment. In particular, a Bandwidth Assigner is developed to observe the agent states, obtain an optimized distribution from the activated Action Executor, and then adjust the bandwidth of each agent on the fly. We implement this distribution generating solution along with the model-free multi-agent deep Policy Gradient algorithm, and compare the performance with static and dynamic baseline strategies, including "Even" (which guarantees balanced bandwidths for all users), "Adaptive" (which assigns more bandwidth to users consuming higher bitrates), and SARSA (which is a standard single-agent RL-driven policy that fails to consider inter-agent utility optimization). By simulating various network environments as well as both constant and adaptive bitrate policies, we validate that the proposed MAPG-finite outperforms all other tested algorithms. With Constant Bitrate (CBR) streaming, MAPG-finite is able to improve the achieved QoE by up to 169.66%, and the fairness by up to 8.28% compared with baseline strategies. Further, with the Adaptive Bitrate (ABR) streaming, up to 41.25% QoE improvement can be obtained.

We conclude the key contributions of this work are:

- We model the bandwidth assignment problem for optimizing QoE and fairness objectives in multi-user online video streaming. The stall time is quantified for general cases under system dynamics.
- Due to the nature of the inter-agent fairness problem, we propose a multi-agent learning algorithm that is proven to converge and leverages two reinforcement learning modules running in parallel to effectively reduce the action space size.
- The proposed algorithm is implemented and evaluated on our testbed, which is able to simulate various configurations, including different reward functions, network conditions, and user behavior settings.
- The numerical results show that MAPG-finite outperforms a number of baselines, including "Even", "Adaptive", and single-agent learning policies. With CBR, MAPG-finite achieves up to 169.66% improvement in the achieved QoE, and 8.28% improvement in the logarithmic fairness; with ABR, MAPG-finite achieves up to 41.25% WoE improvement.

2. Related Work

Multi-Agent Reinforcement Learning: In the past, the Multi-agent Reinforcement Learning (MARL) technique [17] has been discussed for scenarios where all the agents make decisions individually to achieve a global optimal. Existing works include Coordinated Reinforcement Learning [18], coordinates both the action selections and the parameter updates between users, Sparse Cooperative Q-learning [19] allows agents to jointly solve a problem when the global coordination requirements are available, [20] uses the max-plus algorithm as the elimination algorithm of the coordination graph, [21] compares multiple known structural abstractions to improve the scalability, and [22] automatically expand an agent's state space when the convergence is lacking. Apart from the standard Q-learning [23] and policy gradient [24] algorithms, there is rich literature on meta-heuristic algorithms for reinforcement learning. [25] provide an ant-colony optimization method for swarm

reinforcement learning which improves empirically over the Q-learning based methods by using parallel learning inspired by ant-swarms. Building on biological inspired algorithms, [26] provide a genetic algorithm to search for parameters for deep-reinforcement learning. [27] provide a modification of ant-colony optimization by considering ϵ-greedy policies combined with Levy flight for random exploration to search for possible global optima. [28] considered a multi-period optimization using an ant-colony optimization inspired algorithm relaxation induced neighborhood search algorithm for performing search in large neighborhoods.

Recently, along with the development of neural networks and deep learning, the deep-MARL [29] is proposed to resolve real-world problems with larger state spaces. With various aspects of deep-MARL researched – such as investigating the representational power of network architectures [30], applying deep-MARL with discrete-continuous hybrid action spaces [31], enhancing the experience selection mechanism [32], etc. – real-world applications can be solved including wireless sensor networks (WSN) routing [10], vehicle networks spectrum sharing [11], online ride-sourcing (driver-passenger paring) services [33,34], video game playing [35], and linguist problems [29]. Comparing with existing work, our proposed solution in this paper focuses on optimizing inter-agent fairness objectives in reinforcement learning.

Video Streaming Optimization: To improve the performance of data streaming, various techniques have been proposed. The mostly discussed method is Adaptive Bitrate (ABR) [36] streaming, which dynamically adjusts the streaming bitrate to reduce the stall time. The different algorithms include BBA [37], Bola [38], FastMPC [39], LBP [40], FastScan [36], and Pensieve [41]. In addition to ABR and bandwidth allocation considered in this paper, caching is also a popular technique to reduce the stall time and further improve QoE. Inspired by the LRU cache replacement policy, [42] analyzes an alternative gLRU designed for video streaming application, and DeepChunk [7] proposes a Q-learning-based cache replacement policy to jointly optimize hit ratio and stall time. Within an edge network environment, the placement of calculations will also affect the streaming performance, thus work [12] breaks hierarchical service placement problem into sub-trees, and further solve it using Q-learning.

For video streaming services still using Constant Bitrate (CBR) systems, [43] proposed QUVE which estimates the future network quality and controls video-encoding accordingly. [44] considered maximizing QoE by optimizing cache content in edge servers. This is different from our setup where we consider caching chunks at client devices. Similar to us, [45] also provide a bandwidth allocation strategy to maximize QoE. However, they use model-predictive control whereas we pose it as a learning problem and use reinforcement learning. [46] consider a multi-user encoding strategy where the encoding schemes for each user varies depending on their network condition. However, they use a Markovian model and do not consider the possible future network conditions into account. [47] consider a future dependent adaptive strategy where they estimate the TCP throughput and success probability of chunk download. Similar to us, [48] consider a reinforcement learning protocol to maximize QoE for multiple clients. However, they use average client QoE at time t as reward for time t and use deterministic policies learnt from Q-learning [23]. We show that our formulation outperforms standard Q-learning algorithms by considering stochastic policies and reward as function of QoE of the clients.

Our work, by using a model-free deep-RL policy [15], aims to maximize the overall Quality of Experience of multiple agents. To measure the QoE, [14] considers the web page loading time as a factor, [49] tracks graphic settings, [50] focuses on mobile networks such that signal-to-noise ratio, load, and handovers matter. More mapping methodologies could be found in the survey [51].

3. System Model

We consider a server with a total bandwidth of B streams videos to users in set $[K] = \{1, 2, \cdots, K\}$, in which all the users are consuming videos continuously. We consider

a streaming session to be divided into nonidentical logical slots. In each time slot, all the users will maintain requesting/playing chunks from the same videos. Once any user $k \in [K]$ starts a request for a new video in the current time slot l, the slot counter increments, thus the new slot $l+1$ starts for all users in $[K]$, even if the video does not change for users $k' \in [K]/\{k\}$.

Using the logical time slot setting described above, at time slot $l \in [L] = \{1, \cdots, L\}$, a user $k \in [K]$ consumes downloading rate $d_k(l) \geq 0$ to fetch video v_k which is coded with bitrate $r_k(l)$. The downloading speed is limited by $\sum_{k \in K} d_k(l) = B, \forall l \in [L]$, and may update for all users when the time slot increments in the system. The video server continuously sends video chunks to the user with downloading speed $d_k(l)$, and the user plays the video with bitrate $r_k(l)$ which is defined by the property of the video. For Adaptive Bitrate (ABR) videos, we update the bitrate of chunk on starting a new slot only. So, all the chunks sent in slot l are of bitrate $r_k(l)$. For Constant Bitrate (CBR) videos, $r_k(l)$ may remain constant across time slots $l \in [L]$ if multiple slots spans the video v_k.

Table 1. List of the variables used in the paper.

Variable	Description
K	number of clients in the system
k	index for agents, runs from 1 to K
B	total bandwidth of the system
l	slot index
L	total slots considered
$d_k(l)$	download rate for user k in slot l
$r_k(l)$	bitrate of chunk sent to user k in slot l
$v_k(l)$	index of video streamed by user k in slot l
m	chunk index of the video $v_k(l)$
$t_k(l,m)$	time at which user k starts playing chunk m for video $v_k(l)$
$t'_k(l,m)$	time at which server starts sending chunk m for video $v_k(l)$
$\bar{t}_k(l,m)$	time at which user k finishes playing chunk m for video $v_k(l)$

In each slot, we reset the clock to zero. We use $t'_k(l,m)$ to denote the time when the server starts to send the m-th chunk of video v_k in the time slot l, $t_k(l,m)$ to denote the time when the user starts to play video chunk m, and $\bar{t}_k(l,m)$ to denote the moment that chunk m is finished playing. For analysis, we consider that the size of each chunk is normalized to 1 unit.

With our formulation, there will be two classes of users in a time slot l. The first class is of the user who requested a new video and triggered the increment of time slot to l. Since the user has requested a new video, it can purge the already downloaded chunks for the previous video. Users in this class may observe a new downloading rate $d_k(l)$ and video bit rate $r_k(l)$. The second class of users are those who do not request a new video, but a new streaming rate $d_k(l)$ is assigned to them because some other user $k' \in [K]$ has requested a new video and triggered a slot change. For these users, the video bitrates will remain the same from the previous slot $l-1$, or be adjusted solely by the ABR streaming policy when CBR or ABR policies are activated. While for the downloading rate $d_k(l)$, it is updated by the bandwidth distribution policy. Note that a resource allocation scheme may still allocate bandwidth to the user such that $d_k(l) = d_k(l-1)$, however, it may not be always true.

Next, we calculate the stall time in a slot l for both classes of users.

3.1. Class 1: User Requests a New Video

We first calculate the stall durations for user k that has requested a video change. As shown in Figure 2, user k starts to fetch a video from the beginning of the slot l. We assume that the chunk m is played in time-slot l, if not, the calculations for the stall duration for

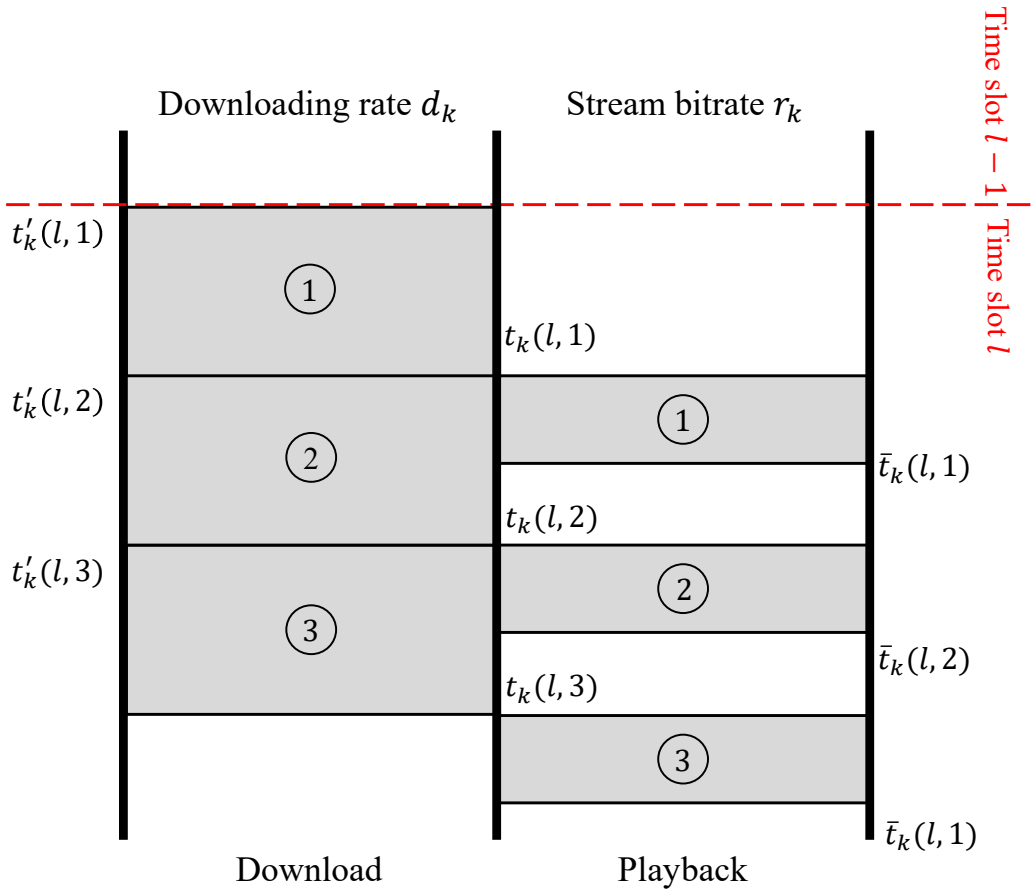


Figure 2. The stall time calculation for users request a new video triggering the system to move to slot l.

those chunks will be studied in Case 2. With the given downloading speed and bitrate, we can observe the relationships between $t'(l,m)$, $t(l,m)$, and $\bar{t}(l,m)$:

$$t'_k(l,m) = \begin{cases} t'_k(l,m-1) + \frac{1}{d_k(l)}, & m > 1, \\ 0, & m = 1, \end{cases} \tag{1}$$

$$t_k(l,m) = \begin{cases} \max(t'_k(l,m), \bar{t}_k(l,m-1)), & m > 1, \\ t'_k(l,m), & m = 1, \end{cases} \tag{2}$$

$$\bar{t}_k(l,m) = t_k(l,m) + \frac{1}{r_k(l)}, m \geq 1. \tag{3}$$

Since we will be limiting the analysis for user k in slot l, we will drop the subscript k and the argument l from $t_k(l,m)$, $t'_k(l,m)$, and $\bar{t}_k(l,m)$ for brevity. Let T denote the time elapsed in time slot l, and let $T_s(l,T,k)$ denote the stall time in slot l till elapsed time T for user k. Clearly, when the video downloading speed is equal to or higher than the video bitrate ($d_k(l) \geq r_k(l)$), the user only has to wait for the first chunk to arrive, then experience a stall-free video playback. Otherwise, for $d_k(l) < r_k(l)$, three conditions need to be considered for T. If T is smaller than or equal to $t(1)$, no video chunk has been played and the stall time is exactly the time elapsed T in the time slot. Otherwise, if T lands in an interval in which a video chunk m is being played, the stall time of T equals to the stall time of $t(m)$, and if T lands in an interval where the user is waiting for the chunk $m+1$ to be

downloaded, the stall time needs an additional wait after the m-th chunk is played. Hence, stall time till end of the slot l, $T_s(l,T,k)$, can be defined as a recursive conditional function,

$$T_s(l,T,k) = \begin{cases} T_s(l,t(m),k), & t(m) < T \leq \bar{t}(m), \\ & d_k(l) < r_k(l), m \geq 1, \\ T - \bar{t}(m) + & \bar{t}(m) < T \leq t(m+1), \\ T_s(l,t(m),k), & d_k(l) < r_k(l), m \geq 1, \\ \min(T,t(1)), & \text{otherwise.} \end{cases} \tag{4}$$

In the condition of $d_k(l) < r_k(l)$, the stall time before the m-th chunk is downloaded is the key to obtain the stall time of T. The stall time of $t(m)$ fits the second condition of Equation (4). So we have:

$$T_s(l,t(m),k) = T_s(l,t(m-1),k) + t(m) - \bar{t}(m-1), m > 1. \tag{5}$$

According to Equations (1), (3), and (2), the difference between $t(m)$ and $\bar{t}(m-1)$ can be calculated. Thus, Equation (5) can be written in a recursive form:

$$T_s(l,t(m),k) = \begin{cases} T_s(l,t(m-1),k) + \frac{1}{d_k(l)} - \frac{1}{r_k(l)}, & m > 1, \\ \frac{1}{d_k(l)}, & m = 1, \end{cases} \tag{6}$$

and further solved as:

$$T_s(l,t(m),k) = \frac{m}{d_k(l)} - \frac{m-1}{r_k(l)}, m \geq 1. \tag{7}$$

Finally, substitute $T_s(t_k(l,m))$ into Equation (4), the stall time of time slot length T is:

$$T_s(l,T,k) = \begin{cases} \frac{m}{d_k(l)} - \frac{m-1}{r_k(l)}, & t(m) < T \leq \bar{t}(m), \\ & d_k(l) < r_k(l), m \geq 1, \\ T - \frac{m}{r_k(l)}, & \bar{t}(m) < T \leq t(m+1), \\ & d_k(l) < r_k(l), m \geq 1, \\ \min(T,t(1)), & \text{otherwise.} \end{cases} \tag{8}$$

Note that if some other user $k' \neq k$ requests a new video triggering increment in time slot from l to $l+1$, the stall duration analysis will fall to the second class of users. We discuss the stall duration for the second class of users in the next section.

3.2. Class 2: Users Continuing with the Old Video

We now discuss the stall time model for users who continue the video from time slot $l-1$ to time slot l (Figure 3).

Assume that in the previous time slot $l-1$, the total slot duration is T'. At the moment of T', a chunk – denoted by 0 – is being downloaded. Because of the chunks were continuously downloaded, by evaluating T' and the download speed $d_k(l-1)$, we can calculate the length or ratio of chunk 0 which hasn't been downloaded by

$$l_0 = 1 - d_k(l-1) \cdot \left(T' \bmod \frac{1}{d_k(l-1)} \right). \tag{9}$$

Note that since the length of chunks are normalized, we have: $l_m = 1, \forall m \neq 0$.

Downloaded with speed $d_k(l)$, the leftover chunk 0 with length l_0 needs time $l_0/d_k(l)$ to be ready for the user to play it. Following the continuous downloading rule, in time

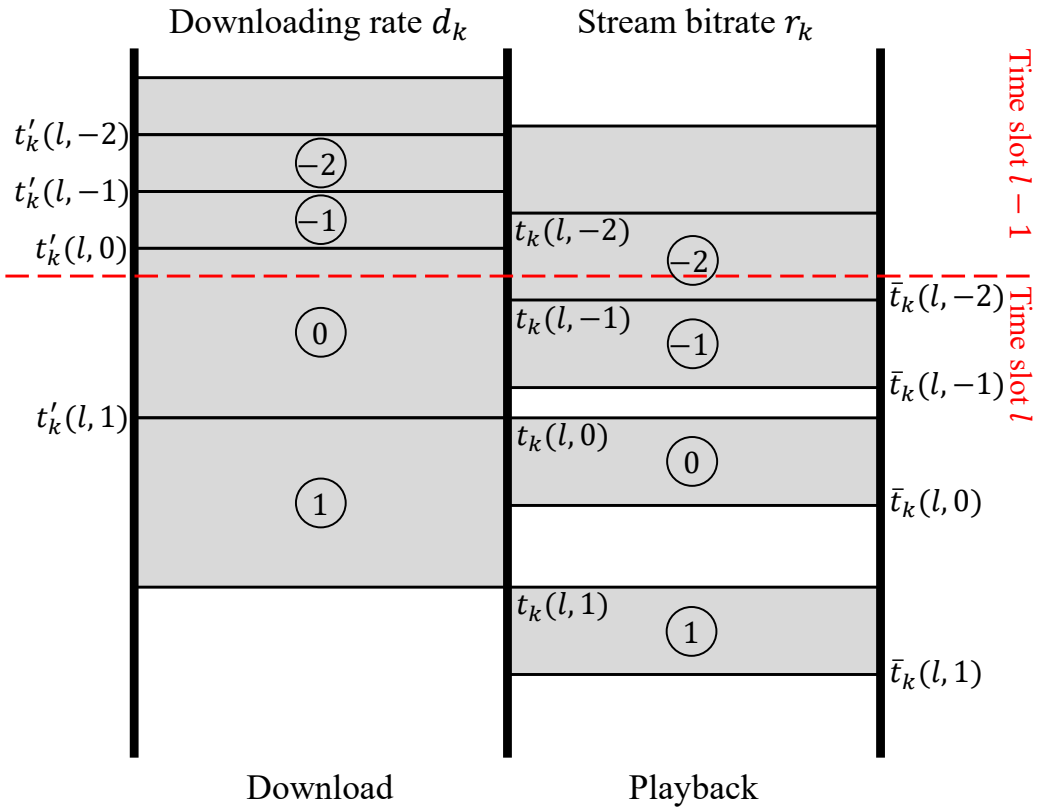


Figure 3. The stall time calculation for users who continue with the video from previous time slot.

slot l, we have $t'(1) = l_0/d_k(l)$. Then similar with Equation (1), the rest of the $t'(m)$ can be recursively obtained.

$$t'(m) = \begin{cases} t'(m-1) + \frac{1}{d_k(l)}, & m > 1, \\ \frac{l_0}{d_k(l)}, & m = 1, \\ 0, & m = 0. \end{cases} \tag{10}$$

We denote the last chunk being played in time slot $l-1$ as chunk $-n$, which is the video chunk ahead of chunk 0 by n, and we denote its finish time calculated in the previous time slot by $\bar{t}'(-n)$. If $n = 1$, and $\bar{t}'(-1) \leq T'$, we know that all chunks before chunk 0 are finished playing in slot $l-1$. Otherwise, chunk $-n$ is being played half-way at the moment of the time slot transition. For the latter case, user k will continue the play of video chunk $-n$ at the beginning of time slot l. Then in the new time slot l, because the video bitrate is not changed, chunk $-n$ will be finished at $\bar{t}(-n) = \bar{t}'(-n) - T'$. Since at the beginning of slot l, chunk 0 is being downloaded, we know that chunks in interval $\{-n, \cdots, -1\}$ in t are all ready to be played. So we can derive the play finish time of chunks $\{-n, \cdots, -1\}$ in slot l as:

$$\bar{t}(m) = \begin{cases} \bar{t}(m-1) + \frac{1}{r_k(l-1)}, & -n < m \leq -1, \\ \bar{t}'(m) - T', & m = -n. \end{cases} \tag{11}$$

As the download finish time $t'(1)$ and play finish time $\bar{t}(-1)$ are defined, the leftover video chunk 0 is played at time $t(0) = \max(t'(1), \bar{t}(-1))$, and finished at $\bar{t}(0) = t(0) + 1/r_k(l)$.

With all the leftover chunk issues tackled, we finally obtain the chunk time equations for time slot l:

$$t'(m) = \begin{cases} t'(m-1) + \frac{1}{d_k(l)}, & m > 1, \\ \frac{l_0}{d_k(l)}, & m = 1, \end{cases} \tag{12}$$

$$t(m) = \max(t'(m+1), \bar{t}(m-1)), m \geq 0, \tag{13}$$

$$\bar{t}(m) = \begin{cases} t(m) + \frac{1}{r_k(l)}, & m > 0, \\ t(m) + \frac{1}{r_k(l-1)}, & 0 \geq m > -n, \\ \bar{t}'(m) - T', & m = -n. \end{cases} \tag{14}$$

With all the time equations obtained, we now can calculate for the stall time using the similar procedure shown in the previous sub-section. Since for $m < 0$, all the chunks are being played stall-free. So if the slot ends at time $T < \bar{t}(-1)$, the stall time will be zero. From chunk $m > 0$, it's possible that the stall appears between the gap where chunk $m-1$ is finished, while chunk m is not downloaded yet ($\bar{t}(m-1) < t(m) = t'(m+1)$). If T happens to be in this gap, the stall time $T_s(T)$ will be the accumulated waiting time of chunks $[0, m-1]$ (denoted as $T_s(t(m-1))$) plus the additional time between T and $\bar{t}(m-1)$. Otherwise, if T happens to be during a chunk m being played, then the stall time $T_s(T)$ should be the accumulated stall time for chunks $\{0, \cdots, m\}$, which can be denoted as $T_s(l, t(m), k)$.

$$T_s(l, T, k) = \begin{cases} T_s(l, t(m), k), & t(m) < T < \bar{t}(m), \\ & m \geq 0, \\ T_s(l, t(m), k) + & \bar{t}(m) \leq T \leq t(m+1), \\ T - \bar{t}(m), & m \geq -1, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

3.3. Quality of Experience

The goal of this work is to maximize the inter-agent QoE utility for all users. In this paper, we consider the fairness utility functions in [13] and optimize the inter-agent fairness with two existing evaluations – the sigmoid-like QoE function and the logarithmic fairness function. By analyzing real-world user rating statistics, a sigmoid-like relationship between the web page loading time and the user QoE was reported in [14]. Inspired by that, we draw a similar nonlinear, sigmoid-like QoE curve to map the streaming stall time ratio, and fulfill that (i) Reducing the stall time for users who already have very low stall time or (ii) Increasing the stall time for users who already suffer from high stall time do not impact the QoE values, while (iii) Users with mediocre QoE are more sensitive to stall time changes:

$$f(x) = \frac{1}{1 + e^{10(x-0.35)}}. \tag{16}$$

We also consider a logarithmic utility function that achieves the well-known proportional fairness [13] among the users:

$$f(x) = \log_2(-x + 2). \tag{17}$$

It is easy to see that, with unit stall time decrease, this utility function provides (i) Larger QoE increment for users experiencing higher stall time, and (ii) Smaller increment for users already enjoying good performance with low stall time.

In both Equations (16) and (17), x represents the stall time ratio for playing a video. It is defined by:

$$x(L_v, k) = \sum_{l \in L_v} \frac{T_s(l, T_l, k)}{T_l}, \tag{18}$$

where L_v denotes the time slots that video v has been played in, and $T_s(l, T_l, k)$ denotes the stall time for user k in time slot l with slot length T_l.

We note that (16) is only one representative QoE function, while other functions may be used. Suppose that in L time slots, $V_k(L)$ be the set of videos played by user k. Then the total QoE of the L time slots obtained by user k are given as:

$$Q_k = \sum_{v \in V_k(L)} f(x(L_v, k)), \tag{19}$$

and for all users, the total QoE is:

$$Q(L) = \sum_{k \in K} Q_k. \tag{20}$$

Substituting (19) and (18) in (20), we have

$$Q(L) = \sum_{k \in [K]} \sum_{v \in V_k(L)} f\left(\sum_{l \in L_v} \frac{T_s(l, T_l, k)}{T_l} \right). \tag{21}$$

Note that the QoE function defined in Equation (19) assigns higher Quality of Experience to lower stall time. The QoE metric remains constant for small stall times. If the stall times are lower than a certain value and are not noticeable, QoE does not vary as obtained in sigmoid-like function of Equation (19). Also, the QoE decreases rapidly with increasing stall times and remains zero if the stall times exceed a certain value therefore ruining the viewing experience.

4. Problem Formulation

In this section, we propose a slice assignment system to distribute the download link bandwidth to users. Let $\bar{\pi}(l) = (\pi_1(l), \cdots, \pi_K(l))$ be a vector in $[0, 1]^K$ such that $\sum_{k \in K} \pi_k(l) = 1$. Each element $\pi_k(l)$ denotes the portion of total bandwidth that user k is assigned to. By this definition, user k's downloading bandwidth $d_k(l)$ under policy $\bar{\pi}(l)$ can be calculated as $\pi_k(l)B$.

The Multi-Agent Video Streaming (MA-Stream) optimization problem is defined as the following:

Problem MA-Stream :

$$\max \quad \sum_{k \in [K]} \sum_{v \in V_k(L)} f\left(\sum_{l \in L_v} \frac{T_s(l, T_l, k)}{T_l} \right), \tag{22}$$

$$\text{s.t.} \quad \sum_{k \in [K]} d_k(l) = B \; \forall l \in \{1, \cdots, L\}, \tag{23}$$

$$\text{var.} \quad \bar{\pi}. \tag{24}$$

We now discuss the MA-Stream optimization problem described in Equation (22)-(24). The Equation (22) denotes the sum of the Quality of Experience for each user $k \in [K]$ across each video played in L time slots. The control variable is the policy π (in Equation (24)) which directly controls the bandwidth allocation. This gives the constraint in Equation (23) where the sum of allocated bandwidths, $d_k(l)$, to each user $k \in [K]$ can be at most the total bandwidth of the system for all slots $l \in [L]$. Moreover, the QoE for any video is a non-linear function of the cumulative stall-durations over each chunk in the video played.

We utilize the deep Reinforcement Learning technique to optimize the bandwidth distribution $\bar{\pi}(l)$. In the following sub-sections, we define the state, action, and objective for the decision making.

4.1. State

At time slot l, the observed state is defined by a $4K$ dimensional vector $s(l) = (v_1(l), \cdots, v_K(l), d_1(l), \cdots, d_K(l), z_1(l), \cdots, z_K(l), c_1(l), \cdots, c_K(l))$, where $v_k(l)$ denotes the video bitrates, $d_k(l)$ represents the currently assigned download speeds, $z_k(l)$ tracks the accumulated stall time for the current playing video until slot l, and $c_k(l)$ counts the number of chunks which are downloaded but not yet played for user $k \in [K]$. For brevity, we use the notation $s(l) = (\bar{v}(l), \bar{d}(l), \bar{z}(l), \bar{c}(l))$ where $\bar{v}(l) = (v_1(l), \cdots, v_K(l)), \bar{d}(l) = (d_1(l), \cdots, d_K(l)), \bar{z}(l) = (z_1(l), \cdots, z_K(l)), \bar{c}(l) = (c_1(l), \cdots, c_K(l))$. We will expand the corresponding vector when necessary. By considering the variables $v_k(l)$ and $d_k(l)$, the learning model should be able to estimate the downloaded and played video chunk information in the current time slot l, while $z_k(l)$ and $c_k(l)$ provide the objective-related history information.

4.2. Action and State Transition

At the beginning of time slot l, in order to find the optimal download speed distribution, multiple decisions are needed to adjust the observed speed distribution. We utilize two decision processes to get the optimal distribution $\bar{\pi}(l)$ while maintaining the constraint shown in Equation (23). One of the process is a *decreasing* process that decides for which user the download speed will be decreased by 1 unit of rate, and the other process is an *increasing* process that decides the user which will obtain the released 1 unit of download speed.

The download speed distribution is iteratively adjusted to a final distribution by recursively running the *decreasing* and *increasing* decision processes. A distribution will not be assigned to the system until the final decision is concluded, and the system will not transit into the next time slot. Assuming at time slot l, with the observed state $s(l) = (\bar{v}(l), \bar{d}(l), \bar{z}(l), \bar{c}(l))$, actions $a_-, a_+, a_- \neq a_+$ are made by the decreasing and increasing processes, the intermediate state $s(\tau')$ can be derived by

$$s(\tau') = (\bar{v}(l), d_1(l), \cdots, d_{a_-}(l) - 1, \cdots, d_{a_+}(l) + 1, \cdots, d_K(l), \bar{z}(l), \bar{c}(l)). \tag{25}$$

Now, this intermediate state $s(\tau')$ is used in the decision making for both processes. New actions will be made to push the distribution towards the final state. Finally, at state $s(\tau)$, when both the increasing and decreasing processes give the same action $a_+ = a_-$, the distribution $\bar{\pi}$ is obtained as $\bar{\pi}(l) = (d_1(\tau), d_2(\tau), ..., d_K(\tau))/B$.

According to $\bar{\pi}(l)$, the system distributes the bandwidth to each user for the time slot l. The next time slot $l+1$ will be triggered when a user switches its playing video. We assume that the new content request for all users follow Poisson arrival processes with arrival rate λ_k for user k, so the mean value of slot duration T_l can be derived by $1/\sum_{k\in K}\lambda_k$, and the probability that user k triggers the state transition is $\lambda_k/\sum_{\kappa\in K}\lambda_\kappa$. For time slot $l+1$, the initial system state $s(l+1) = (\bar{v}(t+1), \bar{d}(t+1), \bar{z}(t+1), \bar{c}(t+1))$ should have video bitrates $v_\kappa(l+1) = v_\kappa(l), (\forall \kappa \in K, \kappa \neq k)$ if CBR is activated as the bitrate policy, and downloading speeds $d(l+1) = \bar{\pi}(l)B$ calculated in the previous time slot.

The accumulated stalls $\bar{z}(l)$ can be calculated using Equations (4) and (15). Let $v_k(l)$ be the video played by the user k in time slot l, and let $l_{v_k(l),0}$ be the time slot where user k starts playing video $v_k(l)$. Let $T_{l'}$ denote the length of time slot l', we have

$$z_k(l) = \sum_{l'=l_{v_k(l),0}}^{l-1} T_s(l', T_{l'}, k). \tag{26}$$

The number of remaining chunks $\bar{c}(l)$ can easily be tracked during the downloading/playing procedures, and observed whenever the information is needed. Both the downloading and playing processes can be monitored. For the downloading process, let $c_d(l) = h_d(l) + \rho_d(l)$, where $h_d(l) \in \mathbb{N}$ denotes the chunk being downloaded of the video being played at the beginning of time slot l, and $\rho_d(l) \in [0, 1)$ denotes the ratio or percentage of chunk $h_d(l)$ that has been completed. The similar mechanism holds for the playing process, $c_p(l) = h_p(l) + \rho_p(l)$. Using both the processes, the remaining chunks $c(l)$ can be calculated by $c(l) = c_d(l) - c_p(l)$ in any time slot l.

4.3. Feedback

As pointed in Equation (22), the goal of the controller is to maximize the average QoE. For our RL algorithm to learn an optimal policy to maximize the objective every slot provides a feedback of the value of the objective calculated from the average stall times for all users.

In Section 4.2, we mention that when the decreasing and increasing processes take decisions $a_-(\tau') \neq a_+(\tau')$, the state transition only happens in a logic domain instead of the realistic time domain. During this intermediate state transition, no real stall time calculations exist and we assign zero rewards for actions $a_-(\tau') \neq a_+(\tau')$ in the intermediate state before converging to $\bar{\pi}(l)B$. When the final distribution is achieved ($a_+ = a_-$), the slot duration T_l can be obtained and hence stall times $T_s(l, T_l, k)$ can be calculated for all users. We can also obtain rewards from the calculated stall times using Equation (21).

The complete schema is presented in Algorithm 1

Algorithm 1 Proposed MA-Stream Algorithm

```
Input: Set of users [K], maximum bandwidth B
for slot l ≥ 0 do
    Observe state s(l) as described in Section 4.1
    Compute bandwidth allocations d_k(l) for all k ∈ [K] using RL engine
    while No user switches video do
        Continue streaming with d_k(l) for all k ∈ [K]
        Store Stall duration, T_s(l, T_l, k) for slot l for all k ∈ [K]
    end while
end for
```

5. Policy Gradient for MA-Stream

In the previous section, we define the network streaming problem MA-Stream for multiple users. Note that the objective defined in Section 4.3 is a nonlinear function (Equation (19)) of total stall duration till the current time instant. At any time slot l, the reward not only depends on the stall times observed by the users in the slot l, but also on the stall times observed by users in the previous time slots. Hence, the decision making module needs to track not only the current state but also the history of the decision and the rewards obtained to select current action. Hence, we are not able to utilize standard RL algorithms that require modeling the problems into MDP. To this end, we leverage Multi-agent Policy Gradient for Finite Time Horizon (MAPG-finite) [15], a novel multi-agent policy gradient RL algorithm, which aims to solve optimization problems without the requirements of MDP modeling. In the following Section 5.2, we give a short description of this algorithm.

5.1. Standard RL Algorithms

Standard RL problems consider an agent that interacts with a Markov Decision Process $\mathcal{M}$. The agent, at time t, observes the state s_t of the environment and plays action a_t to obtain a reward r_t and causes the environment to transition to state s_{t+1} [52]. Let the next state transition probability be $\mathcal{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$ and the expected reward of playing action a in state s be $\mathcal{R}^a_s = \mathbb{E}[r_t | s_t = s, a_t = a]$. The goal of the agent is to find a

policy $\pi(s,a;\theta) = P(a_t = a|s_t = s,\theta)$ parameterized on θ that maximizes the discounted cumulative reward

$$V_\pi(s_0) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t r_t|s_0\right], \tag{27}$$

where s_0 is the initial state, and $\gamma < 1$ is a discounted factor. Using linearity of cumulative rewards in Equation (27), state action value function $Q_\pi(s,a)$ for policy π is defined as

$$Q_\pi(s,a) = \mathcal{R}_s^a + \gamma\sum_{s'}\sum_{a}\mathcal{P}_{ss'}^a\pi(s,a;\theta)V_\pi(s'). \tag{28}$$

Based on Equation (28) many algorithm have been proposed, e.g., SARSA [53–55], Q-learning [23], Policy Gradient [24], etc. Based on these fundamental algorithms, many deep learning based implementations are also proposed [52].

In many network optimization problems, the reward metrics are nonlinear when multiple subjects are jointly optimized. One typical example is resource fairness among agents/users in one network [56,57]. With a nonlinear reward function, the decision making engine must be aware of the historical decisions and states in the past. To demonstrate the requirement of policies that require history, we take the following example. Suppose that there are $K = 2$ users who share the network resource and we want to allocate this network resource fairly between the two users. If we use proportional fairness, the fairness for the two users can be calculated as the sum of logarithms of the QoE indicators of the two users. We call the users 1 and 2, and we assume that both users 1 and 2 start with the same video. In slot 1 the bandwidth allocated to user 1 is higher than user 2 ($d_1(1) > d_2(1)$). This results in higher stall times for user 2 compared to user 1.

Now, in the next time slot 2, user 2 switches the video with video bitrate same as the previous video, and user 1 continues with the old video. Then, since the video bitrates remain the same, allocating a higher download rate to user 2 ($d_2(2) > d_1(1)$) will result in lower stall times for user 2 and hence the fairness can be maximized. This requires the controller to ensure that the decisions at time slot 2 are made keeping account of the decision made in time slot 1. However, the state defined in Section 4.1 does not store the cumulative stall duration of the previous video for the user triggering the current time slot. The algorithm used in this paper to deal with such nonlinear problem is explained in the next subsection.

We further note that one possible way to tackle the Non-Markovian nature is to introduce a high-order Markov model by including the objective value till time slot $l-1$ in the state. This approach, however, potentially increases the state space dramatically to $(SA)^L$, where S is the number of states and A is the number of actions the controller can take. Hence, we consider a multi-agent learning algorithm in the next session, which does not require the use of high-order Markov models and allows individual agent to improve its policy.

5.2. Model-Free Multi-Agent Policy Gradient Algorithm

In [15], a novel model-free algorithm is proposed in order to solve the nonlinear (in time) optimization problems in a finite time horizon scenario. Like the traditional deep policy gradient RL algorithm, the MAPG-finite algorithm utilizes the observed system state $s(t)$ as the input of a neural network which is parameterized by θ, then takes the output of the neural network to be the decision policy π_θ, which indicates the action probabilities. Finally, according to the policy π, an action is randomly chosen to interact with the environment.

With proper training process, the neural network is improved by the reward feedback. Expectedly, the neural network parameter θ should evolve to stage θ^* which maximizes the objective function f:

$$\theta^* = \arg\max_\theta f\left((1-\gamma)J_{\pi_\theta}^1,...,(1-\gamma)J_{\pi_\theta}^K\right), \tag{29}$$

where $J_k^{\pi_\theta}$ denotes the long term reward obtained by agent k running policy π_θ:

$$J_\pi^k = \mathbb{E}_{s_0,a_0,s_1,a_1,\ldots}\left[\lim_{L\to\infty}\sum_{l=0}^{L}\gamma^l f(T_s(l,T_l,k))\right], \tag{30}$$

$$s_0 \sim \rho_0(s_0), a_l \sim \pi(a_l|s_l), s_{l+1} \sim P(s_{l+1}|s_l,a_l). \tag{31}$$

Since the objective function f is differentiable (refer to Equation (16)), the gradient estimation for Equation (29) can be obtained by:

$$\begin{aligned}\nabla_\theta f &= \sum_{k\in[K]}\frac{\partial f}{\partial(1-\gamma)J_k^\pi}\nabla_\theta J_k^\pi \\ &= (1-\gamma)(\nabla_{(1-\gamma)\bar{J}^\pi} f)^T(\nabla_\theta \bar{J}^\pi),\end{aligned}$$

where $\bar{J}^\pi = (J_1^\pi, ..., J_K^\pi)^T$ in which $\bar{J}_k^\pi$ denotes the expected cumulative reward, and further estimated as

$$\hat{J}_k^\pi = \frac{1}{N}\sum_{n=1}^{N}\sum_{l=0}^{L} T_s(l,T_l,k). \tag{32}$$

In each episode n, time slots l runs from 0 to L. Finally, with learning rate β, the step parameter update can be shown as:

$$\theta_{i+1} = \theta_i + \beta(1-\gamma)\left(\nabla_{(1-\gamma)\bar{J}^\pi} f(\hat{\bar{J}})\right)^T(\hat{\nabla}_\theta \bar{J}^\pi), \tag{33}$$

and further utilized for gradient ascent in the neural network.

We now state a formal result of convergence to a stationary point from the gradient ascent steps.

Lemma 1. *For a policy function parameterized with a neural network with softmax activations, and function f with continuous gradients, Equation (33) converges to a stationary point.*

Proof. Since we use softmax activations, gradient of policy π, $\nabla_\theta \pi$ is also continuous in θ and obtaining continuity of $\nabla_\pi J_k^\pi$ from [24, Theorem 2] with respect to parameter θ we have continuity of $\nabla_\theta J_k^\pi$. Using the continuity of gradient $\frac{\partial f}{\partial J_k^\pi}$ from definition of f, the continuity of $\nabla_\theta J_k^\pi$ and [58, Proposition 3.4] we have the convergence of policy to a stationary point. □

6. Evaluation

We conduct a hybrid simulation on a network containing five users over a shared downloading link, and evaluate the performance of the proposed learning algorithm. In particular, three users prefer to watch HD videos (with desired bitrates at $8Mbps$ and $5Mbps$), while the other two users watch videos at lower resolution (with desired bitrates at $2.5Mbps$ and $1Mbps$). The video durations of all users follow an exponential distribution with an identical 120 second average. We run the simulation on both channels with different bandwidth, i.e., $1500KB/s$ and $2000KB/s$ channels. In both settings, our proposed algorithm is shown to substantially outperform the baseline policies (relying on heuristics and single-agent learning) in terms of QoE reward and fairness.

6.1. Evaluation Setup

6.1.1. Evaluated Policies

We evaluate our model-free MAPG-finite algorithm along with three baselines, which are denoted by "Even", "Adaptive", and SARSA policies, as follows.

Policy MAPG-finite: Our proposed algorithm leverages model-free, multi-agent policy gradient to optimize the download bandwidth distribution among agents. Recall

that in the algorithm, multiple decisions/actions are made to either increase or decrease the download speed of specific users by 1 unit. During the training process, the two decision making processes (to increase and decrease download speeds) need to perform random exploration in a non-cognitive fashion, which often leads to long exploration time and thus slow convergence in the optimal policy. Suppose that the bandwidth distribution at time t is π_t. To mitigate this problem during training, we suspend the exploration process if the same bandwidth distribution is observed again in the future, i.e., $\pi_{t+x} = \pi_t$ for some $x \in \mathbb{N}^+$.

Policy "Even": The downloading bandwidth is evenly distributed to all users in the system. For instance, when the total bandwidth is $1500KB/s$, each of the five users will receive $300KB/s$ for its downloading speed, regardless of its demand and preference. This one-size-fits-all policy equally distributes bandwidth among the users.

Policy "Adaptive": The download bandwidth is split between the users in proportion to their desired video bitrates. This policy guarantees that users with high data rate demand (i.e., those watching the high-resolution video) receive a higher downloading speed, while users with low data rate demand receive a lower speed. Specifically, user k will be assigned a bandwidth of $d_k = v_k B / \sum_{\kappa \in K} v_\kappa$, where v_κ is the desired video bit rate of user κ.

Policy SARSA: This policy leverages single-agent learning, SARSA [53–55], to distribute download bandwidth to the users. It uses a standard Policy Gradient strategy with the same state/action definition of our proposed MAPG-finite. Without considering the nonlinear reward function feature, this policy simply utilizes the sum of step reward as its immediate reward for learning. Note that we make the same state variables including reward-related history information $\bar{z}(l)$ known to the SARSA policy to boost the performance of this baseline.

6.1.2. Reward Functions

Our proposed MAPG-finite algorithm allows the maximize of any nonlinear reward in a finite time period. We consider two reward functions in the evaluation, namely QoE and fairness, to compare the performances achieved by different policies.

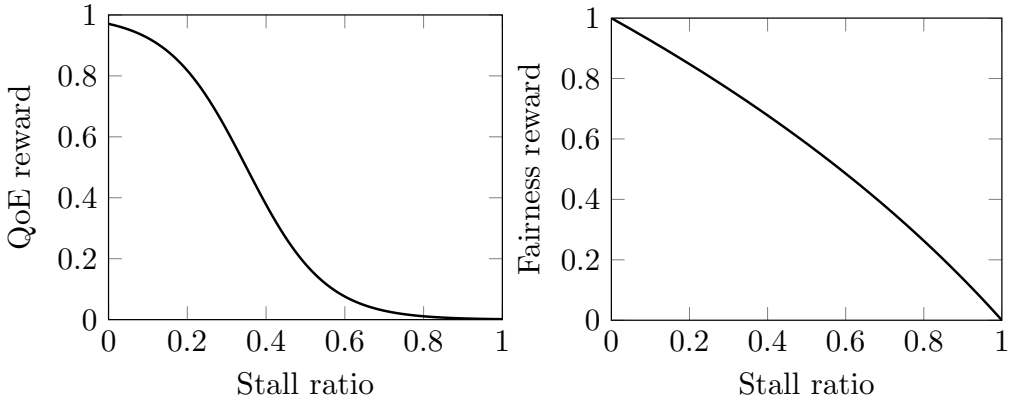
For the QoE reward, we use a sigmoid-like function to measure the reward with respect to the stall time. In particular, we choose parameters in Equation (16) to match a stall-to-QoE curve reported by [14]. We plot our fitted reward function in Figure 4a, which well matches the reported stall-to-QoE curve in [14]. While for the fairness objective, we choose a logarithmic utility function shown in Equation (17) and Figure 4b of users' received stall time. By maximizing the logarithmic function, the proportional-fair QoE assignment between the users [13] is obtained.

Naturally, our proposed multi-agent learning algorithm is able to learn and optimize any reward functions, linear or nonlinear. We note that even when the exact function is unknown, the model-free algorithm can still be trained and evaluated using the real-world user traces.

6.1.3. Users and Videos

We implement a network with five users and both high- and low-resolution videos. In particular, three users prefer to watch high-resolution videos with bitrates of $8Mbps$ (1080p) and $5Mbps$ (720p) (similar to Youtube videos [59]), while the other two users consume $2.5Mbps$ (480p) and $1Mbps$ (360p) videos randomly.

The video durations for all users follow an independent, identical exponential distribution with an average of 2 minutes. Thus, the combined video switching rate for all five users is once every 24 seconds. When a user elects to switch video, a random video is selected and starts streaming. Note that the new video may have the same or different bitrate with the previous video. For example, when User 1 finished watching a video, the new video will have a bitrate of $8Mbps$ or $5Mbps$ with the same 50% probabilities. The user preferences and their corresponding probabilities are shown in Table 2.



(a) A sigmoid QoE function. **(b)** A logarithmic fairness function.

Figure 4. The reward functions to evaluate the performances of different bandwidth assignment algorithms. The QoE function is defined by Equation (16), and the fairness function is defined by Equation (17).

Table 2. Simulated User Preferences.

User	Resolutions	Bitrates	Probabilities
1	1080p	$8Mbps$	0.5
	720p	$5Mbps$	0.5
2	1080p	$8Mbps$	0.5
	720p	$5Mbps$	0.5
3	1080p	$8Mbps$	0.5
	720p	$5Mbps$	0.5
4	480p	$2.5Mbps$	0.5
	360p	$1Mbps$	0.5
5	480p	$2.5Mbps$	0.5
	360p	$1Mbps$	0.5

6.1.4. Implementation

We implement a testbed using Python 3.5. The workflow of the testbed is depicted in Figure 5. First, at the beginning of a new cycle, according to the video switching rates (which are resulted from the known video duration distributions), the *Video Switch* module randomly schedules a user who will be the next candidate to change his video. The users then start downloading video chunks continuously, and their *Download Timers* record the timestamp when each chunk is successfully downloaded (i.e., t'_{m+1} in Figure 3). Next, based on the download timestamps and the bitrate of the current video, the *Playback Timer* schedules the playback and further obtains t_m and $\bar{t_m}$. With all timestamps confirmed, the *Stall Calculator* is able to calculate the stall time for the user/videos. Such stall time is transferred to the distribution module for training and evaluation. Finally, the chosen user randomly picks a new video from its *Video Library* at the end of the current cycle.

Recall from Section 4, the state variables utilized for MAPG-finite decision making include *video bitrates*, *downloading speeds*, *accumulated stall time*, and *residue video chunks*, which can be reported through output paths ②, ④, ③, and ① respectively, in Figure 5. The state variables are then collected by the *State Listener* through input path ⑥. Further, utilizing the *Neural Network*, a bandwidth distribution – as the action – is decided and sent by the *Action Sender* to all the users via ⑦. For training and evaluation purpose, a copy of the stall ratio is also sent from ③ to ⑧. It is processed by the *Reward Function* $f(\cdot)$ to calculate either the QoE (Equation 16) or the fairness (Equation 17) reward. Finally, the reward is delivered to the *Neural Network* for policy backpropagation, and also logged for experiment evaluation. By input path ⑤ of each user, the users adjust their *Download Rates*

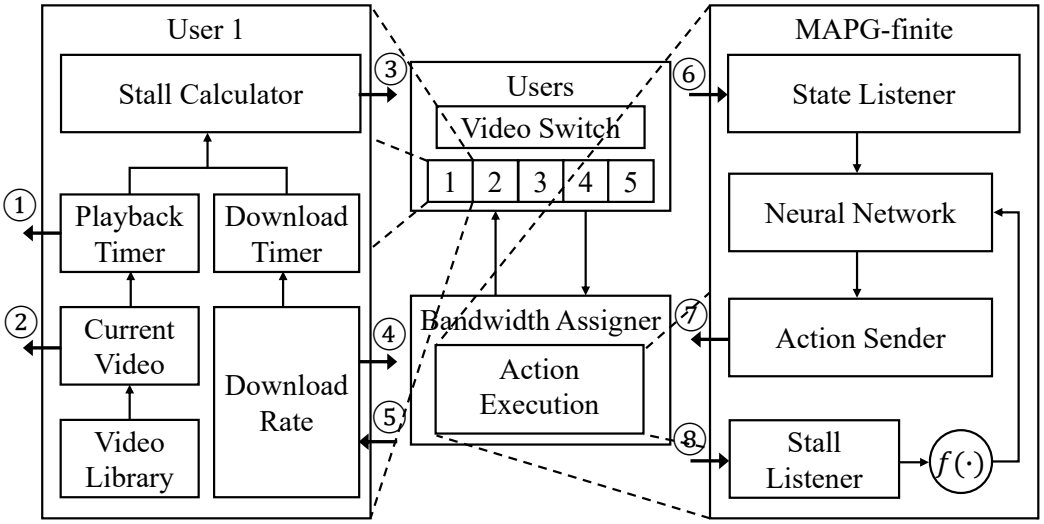


Figure 5. The implementation of our testbed, where the blocks on the left and right show the internal architectures of an individual user module and the MAPG-finite engine respectively. The four state variables $(v(t), d(t), z(t), c(t))$ are collected/calculated by each user, and delivered to the *Action Execution* module (via ②, ④, ③, ① → ⑥). The action is then sent back to each user via ⑦ → ⑤.

Table 3. Reward breakdown for QoE function on $1500KB/s$ download link.

Policy	Total Reward	User Average		
		User	Stall Ratio	Reward
"Even"	3877.05	1, 2, 3 (HD)	0.64	0.07
		4, 5 (LD)	0.13	0.86
"Adaptive"	1773.30	1, 2, 3 (HD)	0.51	0.20
		4, 5 (LD)	0.59	0.14
MAPG-finite	4781.88	1	0.34	0.52
		2	0.50	0.27
		3	0.39	0.44
		4	0.07	0.91
		5	0.08	0.91

according to the bandwidth decision from ⑦. At this point, the testbed completes one workflow cycle and prepares to initiate the next cycle starting with the *Video Switch*.

Using the modularized testbed implemented in this project, we are able to evaluate different policies under various environment configurations, including with different reward functions, network conditions, and user behavior settings. We note that with some minor logic adjustments, we can even test the streaming performance in a discrete time domain, while this paper focuses on continuous time evaluations.

.

6.2. Evaluation Results

The numerical results for the QoE reward function (Equation 16) is depicted in Figure 6. It is shown that our proposed MAPG-finite algorithm outperforms the static "Even" and dynamic "Adaptive" strategies by 23.34% and 169.66% (in terms of achieved QoE) with the shared download link of $1500KB/s$. With the $2000KB/s$ download link, MAPG-finite still obtains 15.30% and 32.58% higher QoE reward than the "Even" and "Adaptive" policies, while the improvement becomes smaller because of smaller marginal QoE improvement when stall time is already small under higher bandwidth. As for the SARSA policy, it is unable to cope with the nonlinear utility function and fails to achieve much improvement over its initial decision policy – "Even". Since the QoE reward function is nonlinear to the

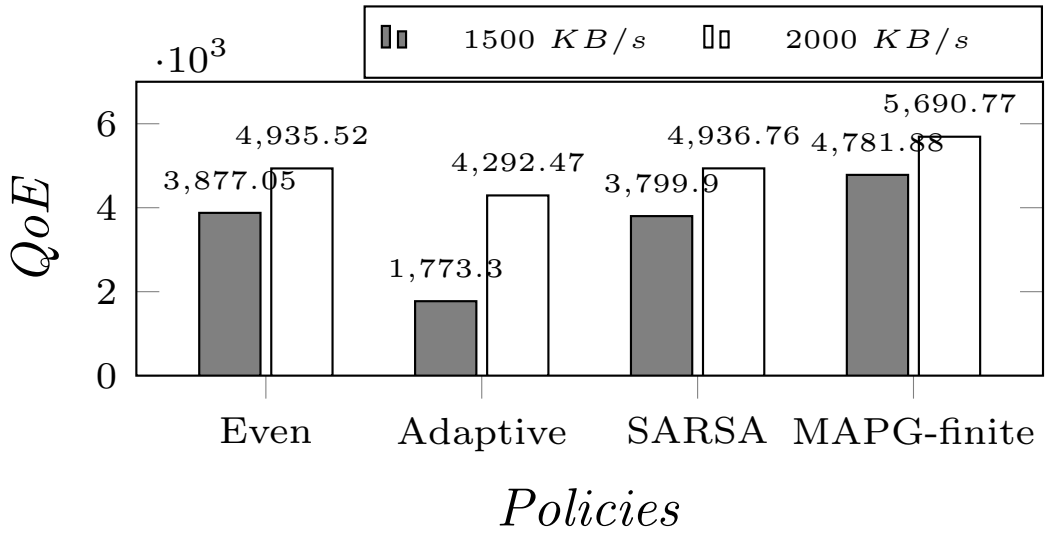


Figure 6. The QoE reward comparison. With the 1500KB/s download link, MAPG-finite outperforms "Even" by 23.34%, "Adaptive" by 169.66%, and SARSA by 25.84%. With the 2000KB/s download link, MAPG-finite outperforms "Even" by 15.30%, "Adaptive" by 32.58%, and SARSA by 15.27%.

Table 4. Reward breakdown for fairness function on 2500KB/s download link.

Policy	Total Reward	User Average		
		Users	Stall Ratio	Reward
"Even"	7796.12	1, 2, 3 (HD) 4, 5 (LD)	0.39 0.08	0.67 0.93
"Adaptive"	8251.66	1, 2, 3 (HD) 4, 5 (LD)	0.18 0.28	0.86 0.77
MAPG-finite	8441.75	1, 2, 3 (HD) 4, 5 (LD)	0.24 0.11	0.80 0.91
"Low Dev"	8263.49	1, 2, 3 (HD) 4, 5 (LD)	0.22 0.20	0.82 0.84

assigned bandwidth, we also observe that the "Adaptive" policy (allocating bandwidth proportional to desired video bitrate) performs worse than "Even" in both cases.

These can be further seen from Table 3, which shows the stall time and reward breakdown of different policies. Apparently, the "Adaptive" policy achieves similar stall time for both HD and LD users, while the "Even" policy sacrifices the performance of HD users, and in return, significantly reduces the stall time of LD users, leading to higher overall QoE. More precisely, according to the QoE reward curve shown in Figure 4a, the reward boost for the LD users is much greater than the loss suffered by the HD users, which finally results in the overall QoE improvement in the "Even" policy. As a learning-based algorithm, MAPG-finite can achieve substantially better performance since it is aware of the current network conditions and system states, in order to optimize the bandwidth distribution between the users. For example, when a user has enough cached chunks for future playback, its bandwidth can be temporarily turned over to the other users, who recently started playing a new video or is suffering from a stall. Thus, all users are able to obtain increased QoE rewards under the MAPG-finite strategy, compared with the baselines.

Results for the fairness reward function is shown in Figure 7. We note that due to the use of logarithmic fairness function, the improvement appears to be smaller when measured by fairness reward than by QoE reward, while the gains should be interpreted in the "multiplicative" sense. Our proposed MAPG-finite still outperforms the "Even" and "Adaptive" strategies (in terms of the logarithmic fairness reward) by 4.83% and 6.75% for 2000KB/s downloading link, 8.28% and 2.30% for 2500KB/s link. With the

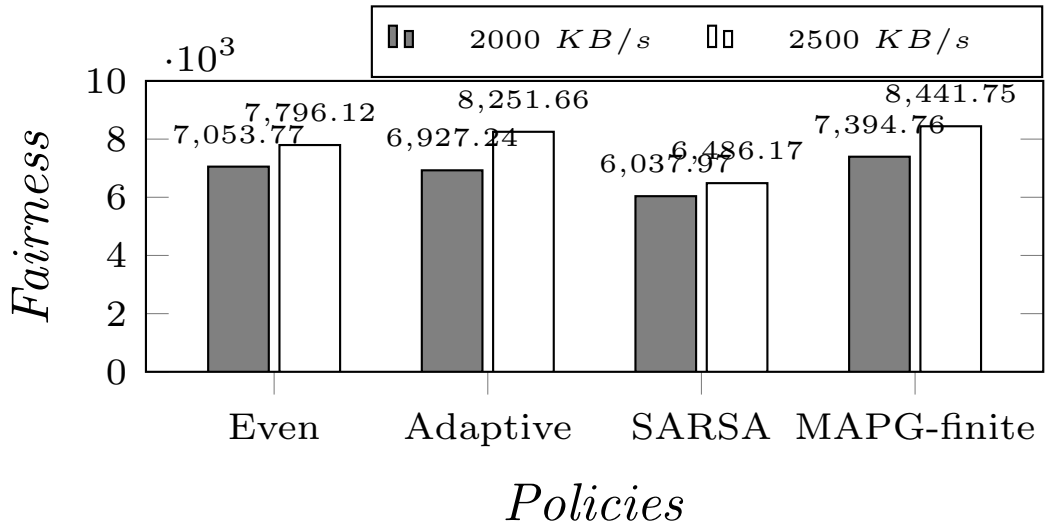


Figure 7. The fairness reward comparison. With the 2000KB/s download link, MAPG-finite outperforms "Even" by 4.83%, "Adaptive" by 6.75%, and SARSA by 22.47%. With the 2500KB/s download link, MAPG-finite outperforms "Even" by 8.28%, "Adaptive" by 2.30%, and SARSA by 30.15%.

2500KB/s downlink, Table 4 shows that the dynamic "Adaptive" strategy achieves similar performances for all users, leading to better fairness reward than the "Even" policy, who creates more difference between the HD and LD users that have significantly different bitrate requirements. On the other hand, our MAPG-finite strategy reduces the average stall ratio of HD users by 38.46%, with a cost of 37.50% higher stall ratio for LD users, compared to the "Even" policy. This way, it is able to reduce the stall ratio deviation of all users from 0.1698 to 0.0712, and improves the fairness reward. Comparing with the "Adaptive" policy, MAPG-finite has about 30% higher stall ratio deviation. However, the optimization object, known as the proportional fairness utility [13], is not solely about "equalizing" different users' performance. (To illustrate this, we construct a "Low Dev" policy in Table 4 that has close-to-zero stall ratio deviation but low fairness reward). The use of proportional fairness reward function indeed balances two important objectives – efficiency (i.e., assigning more bandwidth to users that can achieve higher reward per unit bandwidth) and fairness (i.e., balancing different users' performance). MAPG-finite is able to attain the highest reward under the choice of proportional fairness utilities, demonstrating its ability to achieve complex optimization objectives.

The evaluation results show that the "Even", "Adaptive", and SARSA policies fail to perform consistently under different application scenarios and network conditions, while our learning-based MAPG-finite policy is able to achieve the highest reward. Figure 8 depicts the average download rate distribution decided by the MAPG-finite policy. The gray bars represent the average video bitrates requested by the users. The white bars represent the average download rate achieved by our MAPG-finite policy (for presentation purpose, the unit of download rates is converted from KB/s to $Mbps$). It is can be seen that to maximize the fairness reward, MAPG-finite ensures that **(i)** for users with the same HD preference (e.g., users 1, 2, and 3), the same average downloading bandwidth is assigned to obtain similar stall time for these users, and **(ii)** for users watching videos with lower desired bitrates, less bandwidth is assigned to balance the stall time since video chunks are consumed at a slower pace. According to Figure 7, the "Adaptive" policy gets a lower fairness reward than "Even" under the total download link of 2000KB/s, which indicates that proportionally adjusting the download bandwidth does not always achieve a better result when fairness is concerned. Through exploration and training of RL, MAPG-finite is able to self-teach, improve, and finally converge to an optimal policy, making the model-free suitable to bandwidth allocation with complex networks and objectives that often do not have a straightforward mathematical formulation.

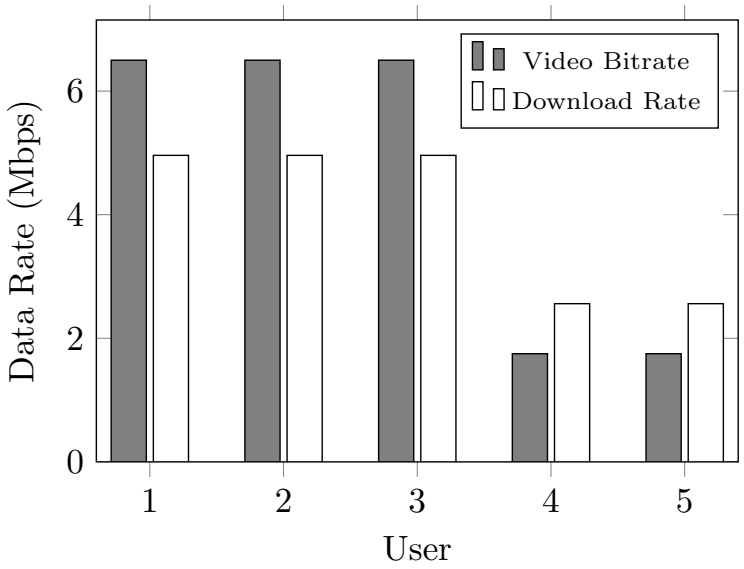


Figure 8. Average download bandwidth for all tested users based on the MAPG-finite strategy with the fairness reward function on 2500KB/s download link. MAPG-finite assures the same average bandwidths for users having HD preferences, and lower bandwidths for users who desire lower bitrates.

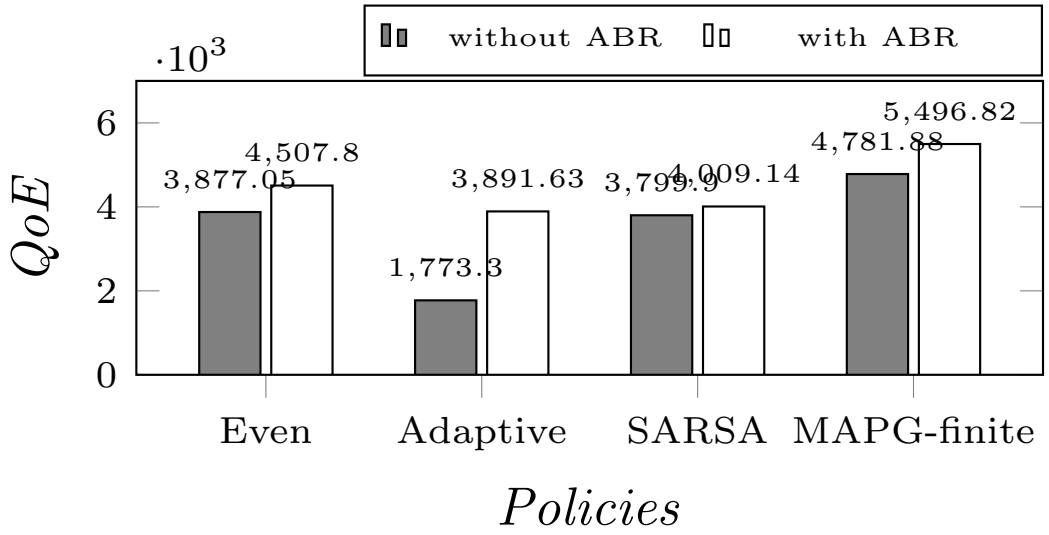


Figure 9. QoE reward comparisons with ABR feature activated/deactivated. The total download bandwidth is 1500KB/s. MAPG-finite achieves 21.94%, 41.25%, and 37.11% than the "Even", "Adaptive", and SARSA policies.

To further illustrate the agility of our proposed MAPG-finite algorithm, we perform another evaluation on a $1500KB/s$ downlink, with an ABR streaming algorithm implemented. Figure 9 depicts the QoE performances for this evaluation. We utilize a basic Buffer-Based ABR algorithm proposed in [37]. Each time a video chunk is requested, if the last chunk downloaded is already being played, the bitrate is adjusted to 80% of the max bitrate of the video to avoid high stall time. When the number of residue cached chunks is more than three, the agent starts to request for the max bitrate, and thus better display quality is obtained. Comparing Figure 9 with Figure 6, all policies receive higher rewards under ABR due to the benefits of bitrate adaptation. We note that MAPG-finite still outperforms the "Even" policy by 21.94%, the "Adaptive" policy by 41.25%, and the SARSA policy by 37.11% . In this evaluation, we choose the Buffer-Based strategy for ABR due to its efficiency for implementation. According to the numerical results, our proposed MAPG-finite is able to adapt well to a dynamic bitrate environment. We are aware that new ABR policies – some are engined by RL algorithms themselves – have been proposed and evaluated [36,38–41] to improve the streaming quality. The key aim of the evaluation was to show that the proposed framework can work on ARB streaming strategies, while not to compare the different streaming strategies. Thus, any streaming algorithm can be used in our evaluations and our results show that efficient bandwidth distribution among multiple agents can be achieved with the proposed algorithms where each agent uses any of the ABR/CBR streaming algorithm.

7. Conclusion

In this paper, we model the MA-Stream problem which apportions bandwidth to multiple users in a video streaming network, to maximize nonlinear, non-convex objectives such as QoE and fairness objectives. We propose a novel multi-agent reinforcement learning algorithm MAPG-finite that is able to work with nonlinear objective functions to solve this optimization problem. Using our testbed implemented in Python, we verify that our proposed solution outperforms existing baseline policies (including "Even", "Adaptive", and single-agent SARSA) measured by both QoE and fairness. Our algorithm improves QoE by 15.27% and fairness by 22.47% for 2000 KB/s link data link. Further, it is able to adapt well in collaboration with existing Adaptive Bitrate (ABR) streaming algorithm by improving QoE more than 30% over the Adaptive algorithm. The interaction between bandwidth distribution and ABR policies could be considered in future work to further improve the performance of video streaming. Another interesting future work is to perform large scale experiments with network scale users.

1. Cisco, V. Cisco visual networking index: Forecast and methodology, 2015-2020. *CISCO White paper* **2016**.
2. Avcibas, I.; Sankur, B.; Sayood, K. Statistical evaluation of image quality measures. *Journal of Electronic imaging* **2002**, *11*, 206–223.
3. Wang, Z.; Lu, L.; Bovik, A.C. Video quality assessment based on structural distortion measurement. *Signal processing: Image communication* **2004**, *19*, 121–132.
4. Kaul, S.; Gruteser, M.; Rai, V.; Kenney, J. Minimizing age of information in vehicular networks. In Proceedings of the 2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. IEEE, 2011, pp. 350–358.
5. Ruan, J.; Xie, D. A survey on QoE-oriented VR video streaming: Some research issues and challenges. *Electronics* **2021**, *10*, 2155.
6. Al-Abbasi, A.O.; Aggarwal, V.; Lan, T.; Xiang, Y.; Ra, M.R.; Chen, Y.F. Fasttrack: Minimizing stalls for cdn-based over-the-top video streaming systems. *IEEE Transactions on Cloud Computing* **2019**, *9*, 1453–1466.
7. Wang, Y.; Li, Y.; Lan, T.; Aggarwal, V. Deepchunk: Deep q-learning for chunk-based caching in wireless data processing networks. *IEEE Transactions on Cognitive Communications and Networking* **2019**, *5*, 1034–1045.

8. Georgopoulos, P.; Elkhatib, Y.; Broadbent, M.; Mu, M.; Race, N. Towards network-wide QoE fairness using openflow-assisted adaptive video streaming. In Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking, 2013, pp. 15–20.
9. Cherif, W.; Ksentini, A.; Négru, D.; Sidibé, M. A_PSQA: Efficient real-time video streaming QoE tool in a future media internet context. In Proceedings of the 2011 IEEE International Conference on Multimedia and Expo. IEEE, 2011, pp. 1–6.
10. Ye, D.; Zhang, M.; Yang, Y. A multi-agent framework for packet routing in wireless sensor networks. *sensors* **2015**, *15*, 10026–10047.
11. Liang, L.; Ye, H.; Li, G.Y. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE Journal on Selected Areas in Communications* **2019**, *37*, 2282–2292.
12. Wang, Y.; Li, Y.; Lan, T.; Choi, N. A reinforcement learning approach for online service tree placement in edge computing. In Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP). IEEE, 2019, pp. 1–6.
13. Lan, T.; Kao, D.T.H.; Chiang, M.; Sabharwal, A. An Axiomatic Theory of Fairness. *CoRR* **2009**, *abs/0906.0557*, [0906.0557].
14. Zhang, X.; Sen, S.; Kurniawan, D.; Gunawi, H.; Jiang, J. E2E: embracing user heterogeneity to improve quality of experience on the web. In Proceedings of the ACM Special Interest Group on Data Communication. ACM, 2019, pp. 289–302.
15. Agarwal, M.; Aggarwal, V.; Lan, T. Multi-Objective Reinforcement Learning with Non-Linear Scalarization. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, 2022, pp. 9–17.
16. Margolies, R.; Sridharan, A.; Aggarwal, V.; Jana, R.; Shankaranarayanan, N.; Vaishampayan, V.A.; Zussman, G. Exploiting mobility in proportional fair cellular scheduling: Measurements and algorithms. *IEEE/ACM Transactions on Networking* **2014**, *24*, 355–367.
17. Bu, L.; Babu, R.; De Schutter, B.; et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **2008**, *38*, 156–172.
18. Guestrin, C.; Lagoudakis, M.; Parr, R. Coordinated reinforcement learning. In Proceedings of the ICML. Citeseer, 2002, Vol. 2, pp. 227–234.
19. Kok, J.R.; Vlassis, N. Sparse cooperative Q-learning. In Proceedings of the twenty-first international conference on Machine learning, 2004, p. 61.
20. Kok, J.R.; Vlassis, N. Using the max-plus algorithm for multiagent decision making in coordination graphs. In Proceedings of the Robot Soccer World Cup. Springer, 2005, pp. 1–12.
21. Fitch, R.; Hengst, B.; Šuc, D.; Calbert, G.; Scholz, J. Structural abstraction experiments in reinforcement learning. In Proceedings of the Australasian Joint Conference on Artificial Intelligence. Springer, 2005, pp. 164–175.
22. Busoniu, L.; De Schutter, B.; Babuska, R. Multiagent Reinforcement Learning with Adaptive State Focus. In Proceedings of the BNAIC, 2005, pp. 35–42.
23. Watkins, C.J.; Dayan, P. Q-learning. *Machine learning* **1992**, *8*, 279–292.
24. Sutton, R.S.; McAllester, D.A.; Singh, S.P.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the Advances in neural information processing systems, 2000, pp. 1057–1063.
25. Iima, H.; Kuroe, Y.; Matsuda, S. Swarm reinforcement learning method based on ant colony optimization. In Proceedings of the 2010 IEEE international conference on systems, man and cybernetics. IEEE, 2010, pp. 1726–1733.
26. Sehgal, A.; La, H.; Louis, S.; Nguyen, H. Deep reinforcement learning using genetic algorithm for parameter optimization. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC). IEEE, 2019, pp. 596–601.
27. Liu, Y.; Cao, B.; Li, H. Improving ant colony optimization algorithm with epsilon greedy and Levy flight. *Complex & Intelligent Systems* **2021**, *7*, 1711–1722.
28. D'andreagiovanni, F.; Krolikowski, J.; Pulaj, J. A fast hybrid primal heuristic for multiband robust capacitated network design with multiple time periods. *Applied Soft Computing* **2015**, *26*, 497–507.
29. Foerster, J.; Assael, I.A.; De Freitas, N.; Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In Proceedings of the Advances in neural information processing systems, 2016, pp. 2137–2145.
30. Castellini, J.; Oliehoek, F.A.; Savani, R.; Whiteson, S. The representational capacity of action-value networks for multi-agent reinforcement learning. *arXiv preprint arXiv:1902.07497* **2019**.

31. Fu, H.; Tang, H.; Hao, J.; Lei, Z.; Chen, Y.; Fan, C. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces. *arXiv preprint arXiv:1903.04959* **2019**.
32. Wang, Y.; Zhang, Z. Experience Selection in Multi-agent Deep Reinforcement Learning. In Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2019, pp. 864–870.
33. Al-Abbasi, A.O.; Ghosh, A.; Aggarwal, V. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* **2019**, *20*, 4714–4727.
34. Haliem, M.; Mani, G.; Aggarwal, V.; Bhargava, B. A distributed model-free ride-sharing approach for joint matching, pricing, and dispatching using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* **2021**, *22*, 7931–7942.
35. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354.
36. Elgabli, A.; Aggarwal, V. FastScan: Robust Low-Complexity Rate Adaptation Algorithm for Video Streaming Over HTTP. *IEEE Transactions on Circuits and Systems for Video Technology* **2020**, *30*, 2240–2249.
37. Huang, T.Y.; Johari, R.; McKeown, N.; Trunnell, M.; Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In Proceedings of the 2014 ACM conference on SIGCOMM, 2014, pp. 187–198.
38. Spiteri, K.; Urgaonkar, R.; Sitaraman, R.K. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* **2020**.
39. Yin, X.; Jindal, A.; Sekar, V.; Sinopoli, B. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, 2015, pp. 325–338.
40. Elgabli, A.; Aggarwal, V.; Hao, S.; Qian, F.; Sen, S. LBP: Robust rate adaptation algorithm for SVC video streaming. *IEEE/ACM Transactions on Networking* **2018**, *26*, 1633–1645.
41. Mao, H.; Netravali, R.; Alizadeh, M. Neural adaptive video streaming with pensieve. In Proceedings of the Conference of the ACM Special Interest Group on Data Communication, 2017, pp. 197–210.
42. Friedlander, E.; Aggarwal, V. Generalization of LRU cache replacement policy with applications to video streaming. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)* **2019**, *4*, 1–22.
43. Kimura, T.; Yokota, M.; Matsumoto, A.; Takeshita, K.; Kawano, T.; Sato, K.; Yamamoto, H.; Hayashi, T.; Shiomoto, K.; Miyazaki, K. QUVE: QoE maximizing framework for video-streaming. *IEEE Journal of Selected Topics in Signal Processing* **2016**, *11*, 138–153.
44. Li, C.; Toni, L.; Zou, J.; Xiong, H.; Frossard, P. QoE-driven mobile edge caching placement for adaptive video streaming. *IEEE Transactions on Multimedia* **2017**, *20*, 965–984.
45. Bentaleb, A.; Begen, A.C.; Zimmermann, R. SDNDASH: Improving QoE of HTTP adaptive streaming using software defined networking. In Proceedings of the 24th ACM international conference on Multimedia, 2016, pp. 1296–1305.
46. Qian, L.; Cheng, Z.; Fang, Z.; Ding, L.; Yang, F.; Huang, W. A QoE-driven encoder adaptation scheme for multi-user video streaming in wireless networks. *IEEE Transactions on Broadcasting* **2016**, *63*, 20–31.
47. Miller, K.; Al-Tamimi, A.K.; Wolisz, A. QoE-based low-delay live streaming using throughput predictions. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* **2016**, *13*, 1–24.
48. Bhattacharyya, R.; Bura, A.; Rengarajan, D.; Rumuly, M.; Shakkottai, S.; Kalathil, D.; Mok, R.K.; Dhamdhere, A. 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, 2019, pp. 251–260.
49. Zinner, T.; Hohlfeld, O.; Abboud, O.; Hoßfeld, T. Impact of frame rate and resolution on objective QoE metrics. In Proceedings of the 2010 second international workshop on quality of multimedia experience (QoMEX). IEEE, 2010, pp. 29–34.
50. Balachandran, A.; Aggarwal, V.; Halepovic, E.; Pang, J.; Seshan, S.; Venkataraman, S.; Yan, H. Modeling web quality-of-experience on cellular networks. In Proceedings of the 20th annual international conference on Mobile computing and networking, 2014, pp. 213–224.
51. Alreshoodi, M.; Woods, J. Survey on QoE\QoS correlation models for multimedia services. *arXiv preprint arXiv:1306.0221* **2013**.

52. Sutton, R.S.; Barto, A.G. *Reinforcement learning: An introduction*; MIT press, 2018.
53. Rummery, G.A.; Niranjan, M. *On-line Q-learning using connectionist systems*; Vol. 37, University of Cambridge, Department of Engineering Cambridge, UK, 1994.
54. Sutton, R.S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Proceedings of the Advances in neural information processing systems, 1996, pp. 1038–1044.
55. Van Seijen, H.; Van Hasselt, H.; Whiteson, S.; Wiering, M. A theoretical and empirical analysis of Expected Sarsa. In Proceedings of the 2009 ieee symposium on adaptive dynamic programming and reinforcement learning. IEEE, 2009, pp. 177–184.
56. Lan, T.; Kao, D.; Chiang, M.; Sabharwal, A. An Axiomatic Theory of Fairness in Network Resource Allocation. In *2010 Proceedings IEEE INFOCOM*.
57. Wang, W.; Li, B.; Liang, B. Dominant resource fairness in cloud computing systems with heterogeneous servers. In Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications. IEEE, 2014, pp. 583–591.
58. Bertsekas, D.P.; Tsitsiklis, J.N. *Neuro-dynamic programming*; Athena Scientific, 1996.
59. YouTube help – Recommended upload encoding settings. https://support.google.com/youtube/answer/1722171.