Zwei: A Self-play Reinforcement Learning Framework for Video Transmission Services

Tianchi Huang, *Student Member, IEEE,* Rui-Xiao Zhang, and Lifeng Sun, *Member, IEEE.*

***Abstract*—Video transmission services adopt adaptive algorithms to ensure users' demands. Existing techniques are often optimized and evaluated by a function that linearly combines several weighted metrics. Nevertheless, we observe that the given function often fails to describe the requirement accurately, resulting in the violation of generating the required methods.**

We propose *Zwei*, a self-play reinforcement learning framework that updates the policy by straightforwardly utilizing the actual requirement. Technically, Zwei effectively rolls out the trajectories from the same initial state, and instantly estimates the win rate w.r.t the competition outcome, where the outcome represents which trajectory is closer to the assigned requirement. We evaluate Zwei with different requirements on various video transmission tasks, including adaptive bitrate streaming, crowdsourced live streaming scheduling, and real-time communication. Results indicate that Zwei optimizes itself according to the assigned requirement faithfully, outperforming the state-of-the-art methods under all considered scenarios. Moreover, we further propose *Zwei*$^+$, which enables Zwei to learn the policies in the vanilla no-regret reinforcement learning scenario. We validate Zwei$^+$ in the adaptive bitrate streaming task and show the superiority of the proposed method over existing approaches.

***Index Terms*—Video transmission, Self-play Reinforcement Learning.**

I. Introduction

THANKS to the dynamic growth of video encoding technologies and essential Internet services [1], currently, humans are living with the generous help of video transmission services. Users often watch exciting videos and live streaming from different video content providers (e.g., YouTube and Kuaishou) or chat with each other via real-time video communication rather than a conventional phone call. In such scenarios, the videos are required to transmit with high bitrates and less rebuffering time or stalling ratio. However, due to the fluctuation and unpredictability of network conditions, blindly achieving high bitrates may heavily increase the probabilities of the rebuffering event. Hence, several rate adaptation methods have been proposed to consider these factors. Further, from the video content provider's perspective, they aim to provide video streaming services with less stalling ratio but lower costs, where it's also necessary to trade off the stalling ratio against the cost. Thus, the services are required to employ scheduling algorithms for balancing the two. In brief, such aforementioned observations are being left on the horns of a classic dilemma: in the video transmission tasks, both the quality of experience (QoE) and quality of service (QoS) are evaluated with contradicted metrics (§II-A). Hence, how to generalize a strategy to effectively trade-off those key factors?

Unfortunately, as much as the fundamental problem has already been published about *two decades* [2], current approaches, either heuristics or learning-based methods, fall short of achieving this goal. On the one hand, recent heuristics often use existing models [3], [4] or specific domain knowledge [5] as the basic working principle. However, such approaches sometimes require careful tuning and will backfire under the circumstance that violated with presumptions, which fails in achieving acceptable performance under all considered scenarios. On the other hand, learning-based methods [6], [7] leverage deep reinforcement learning (DRL) to train a neural network (NN) by interacting with the environments without any presumptions, aiming to obtain a higher score computed by the reward function, where the function is often defined as a linear-based equation with the combination of weighted sum metrics. Nevertheless, although the recent RL-based scheme the potential to outperform existing heuristics in most cases, in this study, we empirically illustrate that i) an inaccurate reward function may mislead the RL-based algorithm to generalize bad strategies, since ii) the actual requirement can hardly be presented by the linear-based reward function with fixed weights. Moreover, iii) considering the diversity of real-world network environments, we can hardly give a proper reward function that can perfectly fit *any* network conditions (§II-B). As a result, despite its ability to gain a higher numerical reward score, such training schemes may generalize a strategy that hardly meets the basic rules of the actual requirements.

Taking a look from another perspective, we observe that the aforementioned problem can be naturally written as a deterministic goal or requirement [8]. E.g., in most cases, the purpose of the adaptive bitrate (ABR) streaming algorithm is to achieve lower rebuffering time first, and next, reaching higher bitrate [9]. It is pretty straightforward that it can be easily understood and refined by others. To this end, we attempt to train the NN based on the assigned requirement without reward engineering. Unfortunately, off-the-shelf learning-based algorithms cannot optimize the policy like this since they cannot provide any gradient information to guide the

Manuscript received July 7, 2020; revised January 15, 2021; accepted February 17, 2021. This work was supported by the National Key R&D Program of China (No. 2018YFB1003703), NSFC under Grant 61936011, 61521002, and Beijing Key Lab of Networked Multimedia.

T. Huang, RX. Zhang and L.Sun are with Beijing Key Lab of Networked Multimedia, Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China. (e-mail: {htc19, zhangrx17}@mails.tsinghua.edu.cn, sunlf@tsinghua.edu.cn)

T. Huang and L.Sun are with BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing ,10084, China.

L.Sun is with Key Laboratory of Pervasive Computing (Tsinghua University), Ministry of Education, China

Lifeng Sun is the corresponding author. (e-mail: sunlf@tsinghua.edu.cn)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia

algorithm towards a better performance directly. Hence, we ask if self-play learning can help to tame the complexity of video transmission services with the actual requirement and primarily, without reward engineering.

Inspired by this opportunity, we envision a self-play reinforcement learning-based framework, known as Zwei [1], which can be viewed as a solution for tackling the video transmission dilemma (§III). The key idea of Zwei is to generalize a strategy that can always meet the actual requirement. Specifically, we rollout decisions in the process of video transmission services. In the MC search process, many simulated trajectories starting from the starting state are generated following the current policy, and the expected long-term win rate will be estimated by averaging the battle results from each of the trajectories. The battle result is determined by a fundamental question: given two strategies collected from the same environment, which is closer to the actual demand? Having estimated the long-term win rate, Zwei then updates the NN by increasing the winning sample's probabilities and reducing the possibilities of the failure sample. In this work, we use the state-of-the-art policy gradient method, namely Dual-clip Proximate Policy Optimization (Dual-PPO) [10] with adaptive entropy weight decay to optimize the NN (§III-C).

We evaluate Zwei's potential using trace-driven analyses of various representative video transmission scenarios. To achieve this, we build several faithful video transmission simulators that can accurately replicate the environment via a real-world network dataset. Specifically, we transfer our proposed framework to three different tasks (§II-A), including adaptive video streaming, crowd-sourced live streaming, and real-time communication service. For each task, we have to face several challenges, such as design adequate NN representation, determine its input and output, define various requirements, as well as decide its baselines and evaluation methodologies. As expected, evaluation results demonstrate the superiority of Zwei against existing state-of-the-art approaches on all tasks.

1) We evaluate Zwei according to four different requirements in the adaptive video streaming scenario. Results show that Zwei outperforms existing adaptive bitrate (ABR) algorithms under any form of requirement. Specifically, in the requirement of minimizing rebuffering time and maximizing the video bitrate, Zwei betters recent work with the improvements on Elo rating [11] of 32.24% - 36.38%.
2) Zwei performs well in crowd-sourced live streaming (CLS) scheduling task, reducing the overall costs by 22% and decreasing over 6.5% on the overall stalling ratio compared to prior study, namely LTS [12].
3) Zwei generalizes well in the real-time communication (RTC) scenario. Comparing the performance of Zwei and state-of-the-art heuristics WebRTC [13], we observe that Zwei effectively improves 11.34% on average receive rate, but reduces 20.49% on loss ratio and 62.95% on 95-percent round-trip-time (RTT).

In the rest of the paper, we investigate how to adapt Zwei to the traditional RL settings, i.e., no-regret reinforcement learning scenario. We propose Zwei$^+$, a novel self-play RL framework that is incrementally implemented based on the Zwei framework. Considering that the conventional no-regret reinforcement learning process, The key idea of Zwei$^+$ is to apply a new battle competition phase. For each collected trajectory, the phase identifies which trajectories, collected from other environments, are also useful for updating. To apply Zwei$^+$ in the ABR scenario, we implement and design a specific similarity function by using the mean and variance for each network trace. Experimental results show that, in the traditional RL settings, Zwei$^+$ can also converge into the optimal ABR policy, which decreases the total rebuffering time by 40%-89% and improves the average bitrate by 1% to 21% compared with off-the-shelf ABR algorithms.

[1]Zwei: means the number *two*, or *double* in German

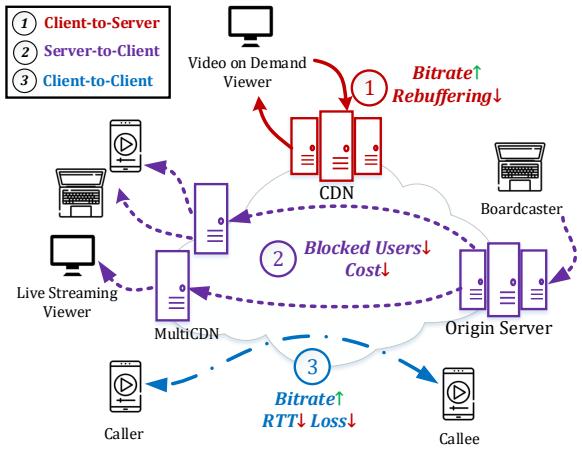


Fig. 1. A brief introduction of today's video transmission service. As shown, the service is mainly composed of adaptive streaming (client-to-server), crowd-sourced live streaming (server-to-client), and real-time communication (client-to-client) (§II-A).

Contributions: This paper makes three key contributions.

- We point out the shortcoming of learning-based schemes in video transmission tasks and employ Zwei, a self-play reinforcement learning framework, for tackling the problems.
- We implement Zwei into three representative video transmission scenarios, including rate adaptation, transmit scheduling, and rate control. We have to develop a new NN representation for each task, determine requirements, and construct a faithful offline simulator. Trace-driven experimental results illustrate that Zwei outperforms existing schemes on all considered scenarios.
- We further present Zwei$^+$, that extends Zwei to adapt the vanilla no-regret reinforcement learning scenario. Moreover, we implement Zwei$^+$ on the ABR task.

II. Background and Challenges

In this section, we first formally introduce the background of adaptive video streaming scenarios, including adaptive streaming (client-to-server), crowd-sourced live streaming (server-to-client), and real-time communication (client-to-client service), and then we briefly elaborate key challenges of each service.

A. Video Transmission Services

To better understand the limitations that conventional video streaming services suffering from, we plot the general service

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia

TABLE I
REQUIREMENTS FOR EACH VIDEO TRANSMISSION TASKS.

Transmission Type	Is ABR?	Requirement
Adaptive Streaming	✓	Rebuffering Time[1]↓, Bitrate[2]↑
Crowd-sourced Scheduling	×	Stalling Ratio[1]↓, Cost[2]↓
Real-time Communication	✓	Latency[1]↓, Loss[1]↓, Bitrate[2]↑

work-flow in Figure 1. As shown, the service commonly consists of three parts:

Adaptive Video Streaming. In the Client-to-Server scenario, users often adopt a video player to watch the video on demand. First, video contents are pre-encoded and pre-chunked as several bitrate ladders on the server. Then the video player, placed on the client-side, uses adaptive bitrate algorithms (ABR) to dynamically pick the proper bitrate for the next chunk to varying network conditions. Specifically, the bitrate decisions should achieve high bitrate and low rebuffering on the entire session [14]. We called it *adaptive bitrate streaming*.

Crowd-sourced Streaming. We now consider a typical Server-to-Client service, if we were the content provider and currently we had multiple content delivery networks (CDNs) with different costs and performance, how to schedule the users' requests to the proper CDN, aiming to provide live streaming services withing less stalling ratio and lower cost? In common, we call that *crowd-sourced live streaming(CLS)* [12].

Real-Time video Communication. Besides, in our daily life, there exists lots of Client-to-Client Services. For example, we usually chat with other users instantly via a video call, namely *Real-Time video Communication (RTC)*. The RTC system consists of a sender and a receiver. During the session, the sender adjusts the sending bitrate for the next period, aiming to achieve the high video bitrate, low round-trip time (RTT), and less loss ratio [7].

We list the representative requirement for each video transmission task in Table I. We label the underlying metric orderly since each metric has attentive priority (e.g., the priority of low rebuffering time is higher than that of high bitrate in the ABR scenario). The video transmission algorithm is often required to obtain better performance under various mutual metrics, making the critical principle of designing a traditional networking algorithm: best-effort delivery [2]. While learning-based approaches pursue best-performance delivery, aiming to maximize the performance over all considered networks. To this end, what's the Achilles' heel of the best-performance approach?

B. Challenges

Recent video transmission algorithms mainly consist of two types, i.e., heuristics and learning-based schemes. Heuristic methods utilize an existing fixed model or domain knowledge to construct the algorithm, while they inevitably fail to achieve optimal performance in all considered network conditions due to the inefficient parameter tuning (§VIII-A,[6]). Learning-based schemes train a NN model towards the higher reward score from scratch, where the reward function is often defined as a linear combination of several weighted metrics [12], [7],

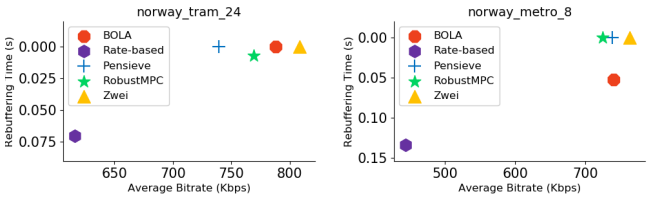


Fig. 2. We show average bitrate and rebuffering time for each ABR method. ABRs are performed over the HSDPA network traces.

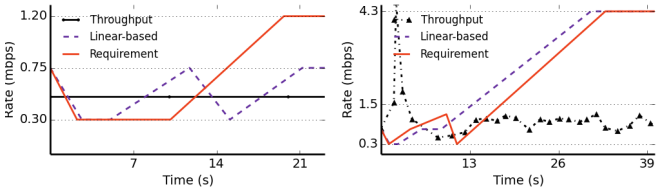


Fig. 3. The comparison of linear-based optimal and requirement-based optimal strategy. Results are evaluated on *fixed (0.5mbps)* and *HSDPA* [18] network traces under ABR scenario, and we can see the difference between the actual requirement and the optimal trajectory generated by the linear-based reward function.

[6]. Nevertheless, considering the characteristics of the video transmission tasks above, *we argue that the policy generated by the linear-based reward fails to always perform on the right track [8]*. In this study, we set up two experiments in the ABR scenario (IV) to prove this conjecture. We use this scenario to describe the challenges because the ABR task is the easiest to understand and closest to the user in the video transmission scenario [1].

Observation 1. *The best learning-based ABR algorithm Pensieve [6] is not always standing for the best scheme on every network traces.*

Given a deterministic QoE metric with linearly combining of several underlying metrics [15], [16], considering the ABR process as a Markov Decision Process (MDP), many approaches have been proposed to learn ABR algorithms via reinforcement learning (RL) method. E.g., RobustMPC [15] is a model-based RL approach that uses a solver and an offline ABR simulator to maximize QoE objectives by planning, as Pensieve [6] is a model-free RL approach that learns the system dynamic via interacting with the ABR environment. However, such a method heavily relies on the accuracy of the given QoE metric. Especially, *how to set a proper QoE parameter for each network condition* is indeed a critical challenge for training a good ABR algorithm. To verify whether QoE parameters have influenced the performance of ABR algorithms, we set up an experiment to report average bitrate and rebuffering time for each ABR method, including Rate-based, BOLA, Pensieve, RobustMPC, and Zwei ([17], [15], [6], §IV-A3). Results are evaluated over the HSDPA network traces. As shown in Figure 2, we can see that, despite the outstanding performances that Pensieve achieves, the best RL-based ABR algorithm does not always stand for the best scheme. By contrast, our proposed method Zwei directly learns the policy w.r.t the assigned requirements, which eventually consistently outperforms existing approaches.

Observation 2. *Existing linear-based weighted reward func-*

tion can hardly map the actual requirement for all the network traces of the given network condition (e.g., 4G and WiFi scenarios).

To better understand the effectiveness of weighted-sum-based reward functions, we compare the optimal linear-based strategy with requirement-based under two representative network traces. A linear-based optimal is the policy that obtains maximum reward. The requirement-based optimal stands for the closest approach in terms of the given requirement. Unsurprisingly, from Figure 3 we observe that linear-based optimal policy performs differently compared with the requirement-based optimal strategy. The reason is that the linear-based optimal policy heavily relies on the given weights, while the weighted parameters are not allowed to be adjusted dynamically according to current network status (e.g., throughput and jitter [19]), which finally leads to the failure of guiding the optimization process on the right track. Generally, we believe that the policy learned by reward engineering might fall into unexpected conditions.

Summary. In general, we observe that no matter how precisely and carefully the parameter of the linear-based reward function tunes, such tuned functions can hardly meet the requirement of any network conditions. E.g., the parameter of stable and unstable network conditions are not the same. Meanwhile, existing studies have also figured out this observation. [20], [21]. To that end, the traditional learning-based scheme, which has often been optimized via the assigned functions, will eventually fail to provide reliable performance on any network traces. We, therefore, attempt to learn the strategies from the original demands.

III. Zwei Design

In this section, we briefly introduce Zwei's details, including its key principle, the basic training algorithm, and the advanced improvements.

A. Self-play Method

As mentioned before, we attempt to generalize the strategy based on actual requirements instead of linear-based reward functions, in which the requirement is often defined by multi-objective goals. For example, as listed in Table I, the main requirement of adaptive streaming is to obtain lower rebuffering time while achieving higher video bitrates. However, as we have already proved (§II-B), it is quite challenging to design an adequate reward function w.r.t the requirement. Fortunately, given two different policies, we can easily identify which one performs closer to the assigned requirement. Following this step, we model the training process as self-play learning that enables the NN to explore better policies and suitable rewards via competing itself. Here suppose that, given any policies, we can compute the outcome that represents the probability of one beating the other in the same video transmission environment, such as adaptive streaming and RTC. We formulate the self-play process as follows. Note that we refer to a task with the same setting as a *game*.

Definition III.1. Suppose two agents, let θ be the weights of a neural net, agent 1 adopts trajectory $T_u \leftarrow \mathbb{T}(\pi_\theta, g_1, \texttt{Env})$, and agent 2 adopts trajectory T_v, where g_1 and g_2 are different random seeds. We can define a symmetric zero-sum functional-form game (FFG) [22] to be indicated by a function

$$\phi : T_u \times T_v \to \mathbb{R}. \tag{1}$$

Where ϕ represents the game rule, which is anti-symmetrical. $\phi > 0$ means agent 1 beats agent 2. The higher the win rate the better for agent 1.

Consider every FFG can be decomposed into two parts, i.e., transitive and cyclic game, where the transitive game means the rules of winning are transitive across different players, and we call a game is cyclic if wins against some agents are necessarily counterbalanced with losses against others, such as rock-paper-scissors or modern MOBA games [22]. In this work, one of the challenges is to define a proper rule for each video transmission task. The goal of the rule is to separate the requirement into two parts, i.e., the transitive part and the cyclic part. The transitive part enables us to determine which trajectory is better between two candidates, as the cyclic part aims to figure out the *best* responses among the set of trajectories on the Pareto Front. We present a hyper-parameter λ to eliminate the cyclic part. The FFG will transfer to a transitive game if λ is small enough. We discuss the usage of λ for each requirement on different video transmission tasks in § IV-§ VI.

Theorem III.1. *With $\phi_{T_v}(\cdot) := \phi(\cdot, T_v)$, if the current game is transitive, we can have the best response oracle defined by*

$$\mathbb{T}^* = \texttt{Oracle}(\pi, \phi_{T_v}(\cdot)), \tag{2}$$

$$\pi^*_\theta = \arg\max_\theta \mathbb{T}^*(\pi_\theta, g, \texttt{Env}) \tag{3}$$

$$s.t. \quad \phi_{T2}(\mathbb{T}^*) > \phi_{T2}(\mathbb{T}) + \phi \tag{4}$$

which can be implemented by an RL algorithm or evolutionary algorithm.

Proof. A task is transitive if there exists a *rating function* f that can measure the performance between two trajectories:

$$\phi(T_u, T_v) = f(T_u) - f(T_v) \tag{5}$$

We start training against a fixed opponent. Thus, assuming we fix agent 2, solving FFG with transitive game normally reduces to finding

$$T^*_u = \arg\max \phi_{T_v}(T_u) = \arg\max f(T_u), \tag{6}$$

as the choice of the fixed agent has no difference to the optimal trajectory. To this end, we can train against a fixed opponent *iteratively* till the best response generates. Just like self-play, which generates a sequence of opponents. Training against a sequence of opponents prevents gradients from vanishing due to large skill differentials [23]. ■

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia

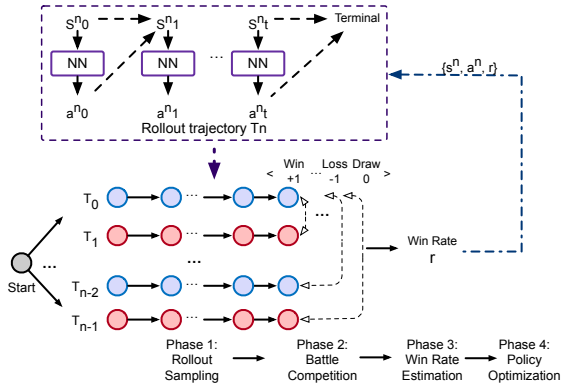


Fig. 4. Overview of Self-play Reinforcement Learning Framework (Zwei). The framework is mainly composed of *four* phases: rollout sampling, battle competition, win rate estimation, and policy optimization.

B. Zwei Overview

We propose Zwei, a self-play reinforcement learning framework for video transmission services. Zwei treats the learning task as a competition between distinct trajectories sampled by itself, where the competition outcome is determined by a set of rules, symbolizing which one is closer to the given requirement. Subsequently, Zwei updates the NN towards achieving a better outcome. Figure 4 presents the main phases in our framework. The pipeline can be summarized as follows:

Phase1: Rollout Sampling. First, we aim to sample N different trajectories $T_n = \{s_0^n, a_0^n, s_1^n, a_1^n, \ldots, a_t^n\}, n \in N$ according to the given policy $\pi(s)$ under the *same environments* ($a_t \sim \pi(s_t)$) and the starting point (i.e., the gray point in Figure 4). Here, we can select Monte Carlo Tree Search (MCTS) to implement the process. However, it's quite impractical to directly apply the MCTS method into the searching process since Zwei requires continuous state spaces rather than only the discrete ones, whereas vanilla MCTS methods diverge [24]. To that end, we reasonably propose a Deep Neural Network (DNN) to map the continuous state spaces to the discrete actions' probabilities. Specifically, Considering that the proposed algorithm needs high efficient yet stable sampling scheme, we then employ Gumbel-Softmax [25] trick for picking a sample from the categorical distribution. Such reparameterization trick draws an action a_t via Eq 7,

$$a_t = \arg\max_i \Big(g_i + \log \pi(s_t)_i\Big) \tag{7}$$

$$g = -\log(-\log \mu) \tag{8}$$

where $\pi(s_t)$ is a categorical distribution of the given state s_t, g is the Gumbel(0, 1) distribution that can be sampled using inverse transform sampling by drawing $\mu \sim \texttt{Uniform}(0, 1)$ [26]. Next, we record and analyze the underlying metric for the entire session. Finally, we store all the sample T_n into a collection D.

Phase2: Battle Competition (BC). To better estimate how the current policy performs, Zwei requires a module to label all trajectories from D: given two *different* trajectories, T_i and T_j which are all collected from the *same environment settings* ($T_i, Tj \in D$)), we attempt to identify which trajectory is positive for NN updating, and which trajectory is generated by the worse policy. Thus, a predefined requirement called ϕ is given to determine the *better* trajectory between the given two candidates, in which *better* means which trajectory is closer to the requirement. At the end of the session, the terminal position s_t is scored w.r.t the rules of the requirement for computing the game outcome o: -1 for a loss, 0 for a draw game, and $+1$ for a win. The equation is listed in Eq. 9.

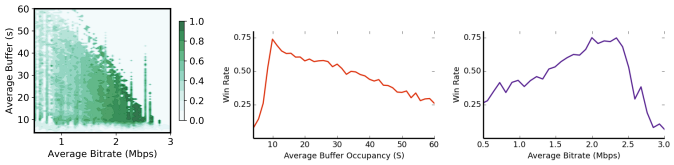


Fig. 5. We plot the underlying win rate for different average bitrate and average buffer occupancy.

$$o_i^j = \phi(T_i, T_j). \tag{9}$$

$$s.t. \quad o_i^j = \{-1, 0, 1\}, T_i, T_j \in D. \tag{10}$$

Phase3: Win Rate Estimation. Next, having computed the competition outcome o_i for any two trajectories, we then attempt to estimate the average win rate r_i for each trajectory T_i in D. The equation is listed in Eq. 11. Notice that the accuracy of the win rate estimation heavily depends on the number N of trajectories.

$$w_i = \mathbb{E}[\phi(T_i,)] = \frac{1}{N}\sum_u^N o_i^u. \tag{11}$$

Phase4: Policy Optimization. In this part, given a batch of collected samples and their win rate, our key idea is to update the policy via *elevating* the probabilities of the winning sample from the collected trajectories, and *diminishing* the possibilities of the failure sample from the worse trajectories. In other words, the improved policy π at state s_t is required to pick the action a_t which produced the best estimated win rate w_t, i.e., $a_t = \arg\max_a E[w_t(s_t, a)]$. Hence, We use Dual-PPO (Dual-clip Proxy Policy Optimization [10]), a state-of-the-art reinforcement learning method that is incrementally implemented based on Proxy Policy Optimization (PPO) [27].

C. Optimizing with Relative Win Rate

As listed before, Zwei's basic training algorithm is derived from a fundamental presumption: the win rate of each state s_t will be converged to zero, that means, *a draw game*. However, we argue that the distribution of the win rate on each state s is different. Thus, we set up an experiment in the ABR scenario: we select a short video clip encoded with six bitrate levels and five chunks and list all the possible trajectories under the same network trace. Then we compute the underlying win rate for each state s_t via ϕ (minimizing rebuffering time and maximizing average bitrate), where s_t is represented by *Average Bitrate* and *Average Buffer*. This makes sense since recent work has proved that these two metrics are critical features for the ABR task [16]. Experimental results

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia

on Figure 5 illustrate that *different states map different win rates*. E.g., the win rate will degrade to 25% if the average bitrate is lower than 1 Mbps and the average buffer is higher than 40 seconds. At the same time, the win rate will achieve the highest score under the conditions that the average bitrate is about 2 Mbps and the average buffer is in the range of 10 seconds to 20 seconds. What's more, we also consider the correlation between the average buffer and win rate, as well as the relationship between average bitrate and win rate, respectively. Results also indicate the influence on the win rate by different values of current states.

To this end, we further add the relative win rate to avoid the bias, which is caused by the high variance in each situation. In detail, Zwei with Baselines consists of a policy network and a value network. The Dual-PPO algorithm adopts the dual-clip method to restrict the step size of the policy iteration and update the NN by minimizing the following *clipped surrogate objective*. If $\hat{A}_t > 0$, Dual-PPO will work equal to the original PPO algorithm [27]. Otherwise, Dual-PPO will clip the ratio $p_t(\theta)$ with a lower bound of the value $\hat{A}_t$. Zwei's NN consists of a policy network and a value network. The loss function of the policy network are computed as Eq. 13,

$$\mathcal{L}^{PPO} = \min\Big(p_t(\theta)\hat{A}_t, \text{clip}\big(p_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t\Big). \quad (12)$$

$$\mathcal{L}^{Policy} = \begin{cases} \hat{\mathbb{E}}_t[\max(\mathcal{L}^{PPO}, c\hat{A}_t)] & \hat{A}_t < 0 \\ \hat{\mathbb{E}}_t[\mathcal{L}^{PPO}] & \hat{A}_t \geq 0 \end{cases} \quad (13)$$

where $\hat{A}_t$ is the *advantage* function which represents the relative value between the future win rate and the current win rate: $\hat{A}_t = w_{t+1} - w_t$, $p_t(\theta)$ denotes the probability ratio between the policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$: $p_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ ϵ and c are hyper-parameters that control how to clip the gradient. We set $\epsilon = 0.2$, $c = 3$ as consistent with the original paper [10].

In practice, considering that Zwei uses basic TensorFlow operations (< 2.0) to set up a static computational graph, while the static graph is tough to be changed during the training, we modify $\mathcal{L}^{Policy}$ (Eq. 13) into an easier implementation version, described in Eq 14.

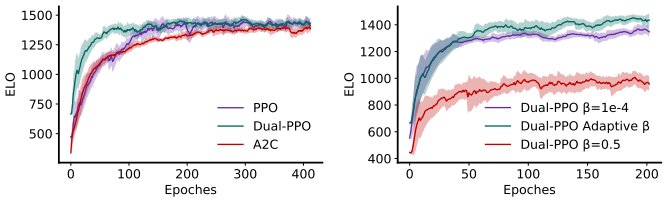
$$\mathcal{L}^{Policy} = (\hat{A}_t < 0)\max(\mathcal{L}^{PPO}, c\hat{A}_t) + (\hat{A}_t \geq 0)\mathcal{L}^{PPO} \quad (14)$$

The value network V_{θ_p} is updated via minimizing the error of $\hat{A}_t$: $\mathcal{L}^{Value} = \frac{1}{2}\hat{\mathbb{E}}_t\left[\hat{A}_t\right]^2$. Furthermore, we add the entropy of the policy $H(s_t;\theta)$ into the loss function to encourage exploration feedback. Here $H^{\pi_\theta}(s_t) = -\sum_{i\in A}\pi_\theta(a_i|s_t)\log\pi_\theta(a_i|s_t)$., β is the entropy weight. To sum up, we summarize the loss function $\mathcal{L}^{Zwei}$ in Eq. 15.

$$\nabla\mathcal{L}^{Zwei} = -\nabla_\theta\mathcal{L}^{Policy} + \nabla_{\theta_p}\mathcal{L}^{Value} + \nabla_\theta\beta H^{\pi_\theta}(s_t). \quad (15)$$

D. Adaptive Entropy Weight Decay

The performance of on-policy reinforcement learning methods is sensitive to the entropy weight β. If the value of the weight is too small, the overall training will be converged quickly but fail to achieve optimal performance. For instance, the default settings in most environments are 0.01 [28]. On the contrary, if the weight is too big, the value network will fail to estimate the accurate baselines due to the high variance of collected trajectories. For example, in practice, the starting weight of Pensieve [6] is 5. As a result, the learned policy eventually lacks stability compared with the optimal strategy.



(a) PPO, A2C vs. Dual-PPO (b) w/wo Adaptive Entropy Weight

Fig. 6. The comparison of typical reinforcement learning algorithm and Dual-PPO. What's more, we also report the effectiveness of adaptive entropy weight.

To alleviate this issue, we propose a novel training trick called adaptive entropy weight decay to dynamically adjust the entropy weight β_t during the training process. In detail, we derive the collected dataset into two parts, i.e., a training set and a validation set. Zwei is trained over the training set and validated on the validation set every 100 epochs. In the beginning, the entropy weight is initialized at β_0. Then the weight will be decreased if the overall performance over the validation set has no longer been improved 10 times. At step t, the entropy weight is updated as $\beta_t = \lambda\beta_{t-1}$, in which λ means the decay rate. In this work, we set $\beta_0 = 1$, $\lambda = 0.8$.

In general, we apply Dual-PPO with adaptive entropy weight to optimize Zwei. It's not gilding the lily since Figure 6 illustrates that Dual-PPO's performance is higher than A2C [28] and Vanilla-PPO [27]. Moreover, we also prove the effectiveness of utilizing the adaptive entropy weight. We observe that Dual-PPO with fixed entropy weight can hardly achieve good results.

E. Training Methodology

The training procedure of Zwei is summarized in Algorithm 1. As shown, Zwei's workflow mainly consists of two parts, i.e., exploration and exploitation. Zwei uses a multi-agent training method, which employs 12 forward propagation agents and one central agent.

Sample Complexity The main difference between Zwei and the conventional RL method is the sample efficiency. Zwei requires N samples per-step, as the conventional reinforcement learning method only requires 1 sample. To this end, for Zwei, given the number of the action A, for S states, each iteration needs $O(NAS^2)$ steps, or slightly faster if the transition function is sparse.

Basic NN Implementation. We use TFlearn to implement the NN and leverage TensorFlow to construct Zwei [2]. Zwei consists of two NNs, including a policy network and a value network. The policy network takes an n-dims vector with *Softmax* active function as the output. The value network outputs a value with *Tanh* function scaled in $(-1, 1)$. We apply

[2]https://github.com/thu-media/Zwei

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia

Algorithm 1 Zwei Training Procedure

Require: Environment Env; Rule ϕ.

Initialize parameters θ with random weights;
repeat
  $D = \{\}$;
  Env ←Sample environment settings, e.g., network trace T, video V, or CDN configuration c;
  for $i \in \{1, 2, \ldots, N\}$ **do** ▷ Rollout Sampling;
    Get state s_t from Env.
    $T_i = \{\}$;
    while not *Terminal* **do**
      Perform a_t according to policy $\pi_\theta(a_t; s_t)$.
      $T_i \leftarrow T_i \bigcup \{s_t, a_t\}$.
      Receive new state s_{t+1}
    $D \leftarrow D \bigcup T_i$
  for $pairs(T_u, T_v) \in D$ **do** ▷ Battle competition;
    Determine win or loss: $o_u^v = \phi(T_u, T_v)$;
  for $T_u \in D$ **do** ▷ Win rate estimation;
    Estimate win rate: $w_u = \frac{\sum_i^N o_u^i}{N}$.
    for $s_t, a_t \in T_u$ **do** ▷ Policy Optimization;
      Update θ and θ_p with $\mathcal{L}^{Zwei}$ (Eq. 15) using collected samples and win rate (s_t, a_t, w_u);
until Converged

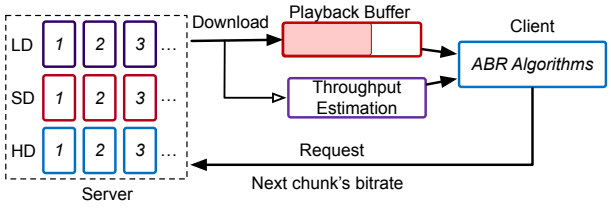


Fig. 7. System architecture of typical adaptive video streaming. The ABR algorithm is placed on the client-side.

the same set of hyper-parameters for training the NN, i.e., sample number $N = 16$, learning rate $\alpha = 10^{-4}$, and entropy weight decay $\lambda = 0.8$. Furthermore, considering the characteristics of video transmission tasks, we construct different NN representations for implementing NN architectures.

IV. CASE I: Adaptive Video Streaming

We first understand how Zwei works in the traditional adaptive video streaming scenario. Technically, we train an NN-based adaptive bitrate (ABR) algorithm w.r.t our proposed method. Detailing the working process, including testbed setup, baseline introduction, and Zwei vs. Existing ABR schemes.

Adaptive Video Streaming Overview As demonstrated in Figure 7, the traditional video streaming architecture consists of a video player client with a constrained buffer length and an HTTP-Server or Content Delivery Network (CDN). The video player client decodes and renders video frames from the playback buffer. Once the streaming service starts, the client fetches the video chunk from the HTTP Server or CDN orderly by an ABR algorithm, and, in the meanwhile, the ABR algorithm, implemented on the client-side, determines the next chunk and next chunk video quality via throughput estimation and current buffer utilization.After finished playing the video, several metrics, such as total bitrate, total rebuffering time, and total bitrate switch will be summarized as a QoE metric to evaluate the performance.

A. Detailed Implementation

1) NN Architecture: **Inputs.** NN takes past t chunks' network status vector $C_k = \{c_{k-t-1}, \ldots, c_k\}$ into NN, where c_i represents the throughput measured for video chunk i. Specifically, c_i is computed by $c_i = n_{r,i}/d_i$, in which $n_{r,i}$ is the downloaded video size of chunk i with selected bitrates r, and d_i means download time for video chunk $n_{r,i}$. Besides, we also consider adding video content features into NN's inputs for improving its abilities on detecting the diversity of video contents. In details, the learning agent leverages $M_k = \{N_{k+1}\}$ to represent video content features. Here N_{k+1} is a vector that reflects the video size for each bitrate of the next chunk $k+1$. The last essential feature for describing the ABR's state is the current video playback status. The status is represented as $F_k = \{q_{k-1}, B_k, D_k, m_k\}$, where q_{k-1} is the video bitrate for the past video chunk selected, B_k, D_k are vectors which stand for past t chunks' buffer occupancy and download time, and m_k means the normalized video chunk remaining. We takes past chunk $k = 8$.

Outputs. The output is an n-dim vector indicating the probability of the bitrate being selected under the current ABR state S_k. In this work, we set the bitrate level as 6, which is widely used in existing ABR papers [6], [19].

NN Representation. We now describe Zwei's NN representation. First, it leverages two Conv-1D layers (filter number=128, kernel size=4) to extract features from the throughput and delay sequence. Meanwhile, Zwei adopts several fully-connected layers (neuron number=128) for obtaining implicit features. Then all the features will be merged by the concentrated layer. Next, the network will be split into two sides, i.e., the policy network and the win rate network. On the one side, the output of the NN's policy network is a *6-dims* vector, which represents the selected probabilities for each bitrate. We utilize the *ReLU* active function for each feature extraction layer and leverage *Softmax* operation for the last layer. On the other size, NN's win rate network outputs a scalar, which uses *tanh* function as the active function.

2) Requirements for ABR tasks.: Recall that the requirement for ABR tasks is diverse, such as selecting bitrates with high bitrate and less rebuffering time [9], as well as picking bitrates with higher video quality and lower rebuffering ratio. To that end, in this case, we implement several representative requirements for evaluating Zwei, which includes:

- ***Minimizing rebuffering and maximizing video bitrates (rebuffer, bitrate).*** Prior research has often added additional smoothness metrics to control the bitrate change of the entire session. While in practice, the following metric is neglectable for the ABR algorithm [9]. Hence we first propose a requirement that constructs the ABR algorithm with minimizing rebuffering time and maximizing video bitrates. The detailed rule of the requirement is described

Algorithm 2 Requirement for the ABR task.

Require: Trajectory T_u, T_v;

1: Average bitrate $\bar{R}_u, \bar{R}_v$, average rebuffering time $\bar{\mathbf{E}}_u, \bar{\mathbf{E}}_v$ from the given trajectories T_u, T_v.
2: Initialize Return $s = \{-1, -1\}$;
3: **if** $\frac{|\bar{\mathbf{E}}_u - \bar{\mathbf{E}}_v|}{\min(b_u, b_v)} < \eta$ or $\frac{|\bar{R}_u - \bar{R}_v|}{\min(r_u, r_v)} < \eta$ **then**
4: $s = \{0, 0\}$; ▷ A draw game.
5: **else if** $\frac{|\bar{\mathbf{E}}_u - \bar{\mathbf{E}}_v|}{\min(\bar{\mathbf{E}}_u, \bar{\mathbf{E}}_v)} < \eta$ **then**
6: $s_i \leftarrow 1, i = \arg\max_{i \in \{u,v\}} \bar{R}$;
7: **else**
8: $s_i \leftarrow 1, i = \arg\min_{i \in \{u,v\}} \bar{\mathbf{E}}$;
9: return s;

in Alg. 2, where we set the threshold $\eta = 0.01$. As listed in line 4, we can see that a draw game will be determined if the absolute percentage of the two bitrates and rebuffering time is lower than η. It can be defined as a cyclic game. At the same time, line 5 indicates a transitive game. We can easily output the winner according to the given rules (line 6 - line 8).

- ***Less rebuffering time and higher perceptual video qualities (rebuffer, quality).*** In recent years, several attempts have also been proposed to pick the video chunk with higher video quality rather than video bitrate [7], [29]. We, therefore, ask if Zwei can handle the requirement that allows the algorithm to achieve higher video qualities as well as lower rebuffering time. In this work, the perceptual quality is measured by Video Multi-Method Assessment Fusion (VMAF) [30], which stands for the state-of-the-art video quality assessment metric. We take past selected video quality, buffer occupancy, throughput, download time, quality, and video size for each bitrate of the next chunk, and chunk remaining as the input. Such settings are different from other requirements.
- ***Achieving better QoE scores (Better QoE).*** To better demonstrate that Zwei can handle various type of requirements, we propose a vanilla requirement that aims to maximize standard QoE objective function QoE_{lin} [15], [6]. Eq 16 lists the details of QoE_{lin}, in which N is the total number of chunks during the session, R_n represents the each chunk's video bitrate, $\mathbf{E}_n$ reflects the rebuffering time for each chunk n. We set the rebuffering penalty as 4.3, as suggested by [6].

$$\mathtt{QoE} = \sum_{n=1}^{N} R_n - 4.3 \sum_{n=1}^{N} \mathbf{E}_n - \sum_{n=0}^{N-1} |R_{n+1} - R_n|, \quad (16)$$

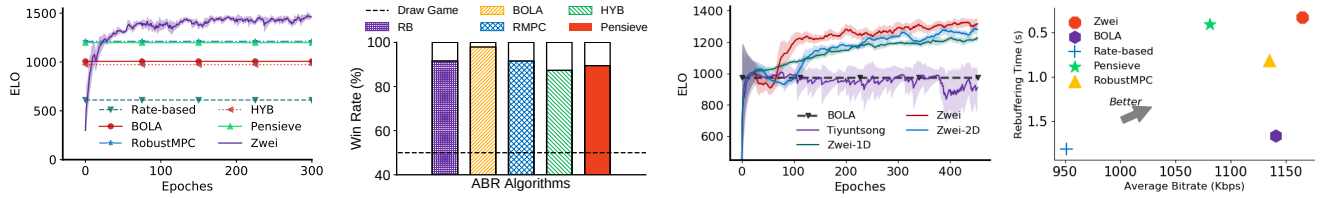
- ***Less rebuffering time, lower smoothness, and higher video bitrates (rebuffer, smoothness, quality).*** Besides, we believe that Zwei can also solve the overall optimization problem with smoothness metric. Thus, we apply a 3-stage specific requirement, i.e., 1) reducing rebuffering time, 2) decreasing smoothness, and eventually, 3) increasing video bitrates. Such requirements are quite challenging since Zwei has to balance three mutual objectives, as the searching space seems larger than that of achieving a proper equilibrium between two objectives.

3) ABR Baselines: In this part, we select several representational ABR algorithms from various type of fundamental principles:

1) ***Rate-based* (RB) [31]:** the basic baseline for ABR problems. Firstly, it leverages *harmonic mean* of past five throughput measured as future bandwidth. Next it picks the next chunks' bitrate with closest and lower than the predicted bandwidth.
2) ***Buffer-based Approach* (BBA) [3]:** A buffer-based approach that dynamically chooses next chunks' bitrate according to the current buffer occupancy.
3) ***BOLA* [17]:** the most popular buffer-based ABR scheme in practice. BOLA turns the ABR problem into a utility maximization problem and solves it by using the Lyapunov function. We use BOLA provided by the authors [32].
4) ***RobustMPC* (RMPC) [15]:** a state-of-the-art heuristic method that maximizes the objectives by jointly considered the buffer occupancy and throughput predictions. We implement *RobustMPC* by ourselves.
5) ***HYB* [19]:** A hybrid algorithm that considers both the predicted throughput and current buffer occupancy. For each chunk, HYB picks the highest bitrate that can avoid rebuffering time by only considering one step ahead.
6) ***Pensieve* [6]:** the state-of-the-art learning-based ABR scheme, which utilizes deep reinforcement learning to select bitrate for next video chunks. We use the pre-trained model provided by the authors.
7) ***Tiyuntsong* [8]:** the first study of multi-objective optimization ABR approach. Tiyuntsong uses the actor-critic method to update the NN via the competition with two agents under the same network condition. Note that Tiyuntsong's learning agent uses DRL to update gradients w.r.t the sample generated by the agent independently, which not only sample inefficient but also reaching the optimal policy.
8) ***Comyco* [29]:** Comyco is a video quality-aware ABR approach that leverages imitation learning methods for training the policy via imitating expert trajectories, enormously improving the learning-based methods by tackling the two fundamental issues, i.e., low sample efficiency and lack of awareness of the video quality information. We adopt the pre-trained model provided by the authors [29].

4) Evaluation Metrics: The Elo rating [33] is a traditional method for calculating the relative performance of players in zero-sum games. Specifically, a player's Elo rating is represented by a number that may change w.r.t the outcome of games played. After every game, the winning player takes points from the losing one, where the difference between the ratings of the winner and loser determines the total number of points gained or lost after a game. Thus, the Elo rating system is suitable to compare different schemes via win rate information only. We set the initial Elo rating as 1000 [8]. For more information about the Elo rating, please refer to the original paper [33].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia



(a) Zwei vs. ABR schemes (b) Overall Win Rate (c) Zwei vs. Tiyuntsong (d) ABR Scheme Details

Fig. 8. This group of figures show the comparison results of Zwei and other ABR approaches, where the goal of Zwei is **to minimize rebuffering while maximizing video bitrates**. Results are evaluated in the typical ABR system with HD Videos (1080P, minimum bitrate=0.3mbps, maximum bitrate=4.3Mbps).

5) Detailed Implementation: We use the standard ABR simulation environment [6].

Network and Video Dataset We train and validate Zwei on various network datasets, including HSDPA [18], FCC [34] and Oboe [19]. Meanwhile, we adopt *EnvivioDash3*, a video that commonly used in recent work [19], [6], [29], to validate Zwei, where the video chunks are encoded as $\{0.3, 0.75, 1.2, 1.8, 2.8, 4.3\}$ Mbps.

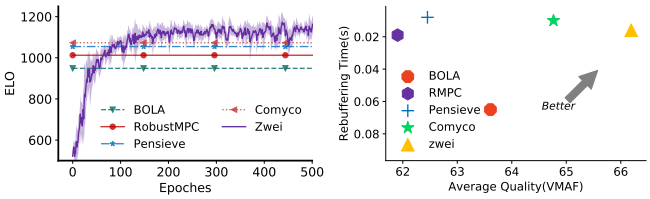


Fig. 9. Comparing Zwei with existing ABR approaches with the goal of less rebuffering and higher video quality.

B. Results Analysis

Zwei vs. Existing schemes. We compare Zwei with existing ABR schemes over the HSDPA dataset. Results are computed as Elo-score [33] and reported in Figure 8(a). Through the experiments, we can see that Zwei outperforms recent ABR approaches. In particular, Zwei improves Elo-score by 36.38% compared with state-of-the-art learning-based method Pensieve and increases 31.11% in terms of state-of-the-art heuristic method RobustMPC. Furthermore, Zwei outperforms recent heuristics on Elo-rating of 158.52% (RB), 62.08% (HYB), and 56.76% (BOLA), respectively. Figure 8(b) demonstrates that Zwei wins against proposed schemes, with the high winning ratings of 91.55% (RB), 97.89% (BOLA), 91.55% (RMPC), 87.32% (HYB), 89.44% (Pensieve). Besides, we illustrate the detailed results of the proposed methods on average bitrate and average rebuffering in Figure 8(c). Results show that Zwei can not only achieve the highest bitrate but also obtain the lowest rebuffering under all network traces. Recall that the main requirement is to obtain lower rebuffering time and higher video bitrate.

Zwei vs. Tiyuntsong. Moreover, to better understand the superiority of our proposed method, we compare Zwei with the previously proposed self-learning scheme *Tiyuntsong* in the same experimental settings. During the training process, we report the Elo-curve on the same validation set in Figure 8(b). As expected, Zwei, using different NN architectures, outperforms Tiyuntsong on average Elo-score of 35%. It's worth noting that, Tiyuntsong is also a self-play learning method that uses the conventional actor-critic method to update the NN for obtaining a higher requirement. We now explain the key difference between our proposed method Zwei and Tiyuntsong.

1) Tiyuntsong treats the ABR training task as a static game with incomplete information, as the training process is done by the competitive results generated by two independent agents. On the contrary, Zwei rolls-out several trajectories by sampling the same policy. In this way, the high variance of win rate results, which is caused by different policies, will be effectively eliminated.
2) Tiyuntsong aims to update the policy via a long-term cumulative win rate W_t, in which $W_t = w_t + \gamma W_{t+1}, \gamma = 0.99$. For each step, Tiytuntsong receives an instant win rate, scaled in $\in (0, 1)$. By contrast, Zwei employs Monte Carlo sampling to compute the current win rate. Thus, as much as Zwei's training algorithm is equal to original actor-critic methods with $w_t = 0$ and n-step discounted factor $\gamma = 1$ in mathematics, we believe that the fundamental principle of that two approaches is different.

Zwei with Different NN Architectures. This experiment evaluates Zwei with all considered NN architectures and compares their performance under the same network setting. Zwei-1D is the standard ABR NN architecture proposed by Pensieve [6], Zwei-2D stands for the advanced ABR NN architecture proposed by QARC [7], and Zwei only leverages three fully-connected layers sized $\{128, 64, 64\}$. Experimental results are calculated with Elo-score and shown in Figure 8(b). Unsurprisingly, when Zwei trains with some complicated NN architecture (Zwei-1D and Zwei-2D), it generalizes poorly and performs worse than the fully connected NN scheme. This makes sense since ABR is indeed a light-weighted task that can be solved in a practical and uncomplicated manner instead of a NN incorporating some *deep* yet *wide* layers.

1) Rebuffering & Video quality: What's more, Zwei can generalize outstanding quality-aware ABR algorithms w.r.t the assigned requirements, i.e., minimizing rebuffering and maximizing video quality. Results are reported in Figure 9. We observe that Zwei surpasses existing ABR algorithms, with the improvements on Elo ratings of 12.58%-24.73%. At the same time, detailed results on average quality and rebuffering time demonstrate that Zwei reaches the best average video quality (66.19) among all the ABR candidates and rivals recent ABRs on the average rebuffering time (0.016). In particular, comparing the average quality and rebuffering

TABLE II
WE REPORT THE AVERAGE BITRATE, REBUFFERING TIME, SMOOTHNESS AS WELL AS QOE FOR EACH ABR SCHEME. *In this part, we attempt to maximize linear-based QoE objective function.*

ABR	Bitrate (Mbps)	Rebuffering (s)	Smoothness (Mbps)	QoE
RB	0.951±0.554	0.038±0.067	0.078±0.051	0.71±0.612
BBA	1.141±0.58	0.035±0.062	0.352±0.106	0.639±0.649
RMPC	1.142±0.612	0.019±0.052	0.139±0.063	0.921±0.622
HYB	1.081±0.578	0.003±0.029	0.207±0.086	0.86±0.534
Pensieve	1.081±0.553	0.008± 0.037	0.12±0.051	0.925±0.57
Zwei	1.126±0.557	0.015±0.08	0.087±0.047	**0.975±0.632**

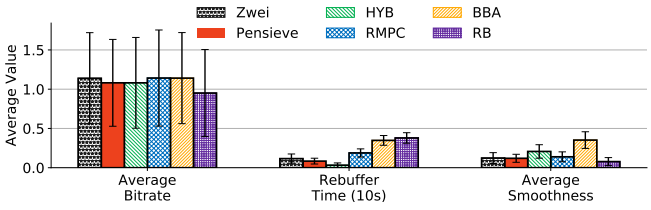
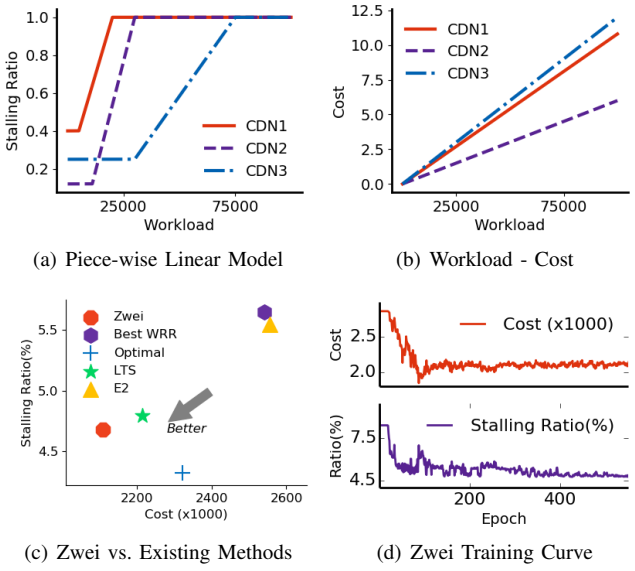


Fig. 10. Comparing Zwei with existing ABR approaches under the HSDPA network traces. In this part, we minimize the rebuffering and smoothness, and maximize the video bitrate.



(a) Piece-wise Linear Model (b) Workload - Cost

(c) Zwei vs. Existing Methods (d) Zwei Training Curve

Fig. 11. Results of Zwei on the CLS environment. We collect Zwei's training curve during the training process and observe that there also exists *two stages* on the entire process.

time of Zwei with the state-of-the-art quality-aware ABR algorithm Comyco, we find that Zwei betters Comyco on the average quality of 2.21% at the cost of increasing the average rebuffering time of only 6 milliseconds. Here, Zwei's rebuffering time is still acceptable, as it is similar to that of RMPC performs.

2) Better QoE: Can Zwei solve the requirement of achieving a better linear-based QoE score? To answer the question, we train and validate Zwei w.r.t the better QoE requirement. Table II details each metric, including average bitrate, rebuffering, and smoothness. Zwei outperforms recent ABRs, with the improvements on average QoE of 5.52% - 59.84% across the HSDPA dataset. Moreover, we also find that Zwei wins against the learning-based ABR approach Pensieve for almost 80% of the sessions. Specifically, Zwei slightly increases the average rebuffering time compared with Pensieve (from 8 to 15ms), but heavily improves the average video bitrates (from 1.081 to 1.126Mbps), and in the meanwhile, reducing average smoothness (from 0.139 to 0.087Mbps) in comparison of RobustMPC.

3) Rebuffering, Smoothness, & Bitrate: In this requirement, we attempt to minimize the rebuffering and smoothness, and in turn, maximizing the video bitrate. Figure 10 shows the comparison of several underlying metrics, including average bitrate, rebuffering time, and smoothness. As expected, Zwei (in gray) can avoid rebuffering and bitrate changes, as it also obtains the highest bitrate. An interesting observation is that as much as Zwei doesn't provide the lowest rebuffering time and bitrate changes among all ABR schemes, e.g., Zwei performs slightly higher than Pensieve and HYB in terms of rebuffering time and requires more bitrate changes compared with RB, the learned policy wins most of the sessions, i.e., 76.06% on RB, 83.8% on BOLA, 52.82% on RMPC, 86.62% on HYB, and 69.72% on Pensieve.

V. CASE II: CROWD-SOURCED LIVE STREAMING

In this part, we evaluate Zwei in the Crowd-sourced Live Streaming (CLS) scheduling task and compare its performance with several state-of-the-art scheduling algorithms. Results illustrate that Zwei can leverage historical CDN information for providing the live video streaming service with less number of blocked (or stalling) users and lower CDN costs.

CLS System Overview. Upon receiving viewers' requests, the CLS platform will first aggregate all stream data to the source server, and then deliver the them to viewers through CDN providers according to a certain scheduling strategy. We treat the ratio of CDN i at time step t as x_{i_t}. At time step t, x_{A_t} of users are redirected to CDN$_A$, while x_{B_t} and x_{C_t} users are redirected to CDN$_B$ and CDN$_C$, respectively. Note that such ratios should satisfy: $x_{A_t} + x_{B_t} + x_{C_t} = 1$.

A. Detailed Implementation

Testbed Setup. As suggested by previous work [12], our experiments are conducted on the real-world CLS dataset provided by Kuaishou, spanning 1 week (6 days for training and 1 day for testing). At each time, we select 3 candidates from 4 different CDN providers, and we fit a separate simulator for each of them. Moreover, we change the decision every 1 minute, i.e., the decision duration=1. The CLS task lasts 1440 steps, i.e., 1-day (1440 minutes), to achieve the terminal state. Such a task with long-term feedback rewards brings out a great challenge to Zwei. Thanks to the high sample efficiency feature of the Dual-PPO algorithm [10], we train Zwei for only 2 hours to converge in a stable result, which is 2× faster than the vanilla PPO algorithm [27].

CDN Configuration. The CDN configuration is demonstrated in Figure 11. Consistent with previous work [12], we

use a piece-wise linear model to characterize this relationship between workload and block ratio (Figure 11(a)). Note that the following features are extracted by the real CDN dataset. What's more, we define the CDN pricing model w.r.t various CDN providers in the industry, such as Amazon E2 and Tencent CDN, and plot the correlation between the workload and cost in Figure 11(b). Specifically, given three heterogeneous CDNs, i.e., CDN_A, CDN_B, CDN_C, we set the unit price P_i for each CDN as 0.72, 0.4, 0.8 respectively, where the price means actual cost (CNY, Chinese Yuan) per Gigabytes. Thus, we can compute the estimated intermediate cost c_i^j for CDN_i at time j: $c_i^j = \bar{b} * w_i^j * P_i$. Here we assume that each person watches the live video streaming with the bitrate $\bar{b}$ of 1.2Mbps, which is the default setting in the industry. w_i^j is the instant workload of CDN_i at time j. To this end, the accumulative cost C_u for trajectory T_u will be calculated as $C_u = \sum_i \sum_j c_i^j$.

NN Representation. We implement Zwei for the CLS task as suggested by LTS [12]. More precisely speaking, for each CDN provider CDN_i, Zwei passes past 20 normalized workloads and stalling ratio to a Conv-1D layer with filter=64, size=4, and stride=1. Then several output layers are merged into a hidden layer that uses 64 units to apply the *Softmax* activation function. We model the action space as a heuristic way: each CDN provider has 3 choices instead: increase its configuration ratio by 1%, 5%, and 10% based on the previous stalling ratio. And Zwei will reduce the ratio for CDN that obtained the worst previous ratio.

Requirements for CLS tasks. In this case, Zwei is proposed to achieve a less stalling ratio with lower costs. Same as Alg. 1, we use the threshold η for separating the cyclic game and the transitive game.

Baselines. Like other scenarios, we also compare Zwei with the following state-of-art scheduling baselines:

1) ***Weighted round-robin* [35].** The idea of weighted round-robin (WRR) is: the requests will be redirected to different CDN providers w.r.t a constant ratio. We adopt the algorithm with the *best* parameters.
2) ***E2* [5].** Exploitation and Exploration (E2) algorithm utilizes harmonic mean for estimating CDN providers' performance and select with the highest upper confidence bound of reward. We use the E2 algorithm provided by the authors [5].
3) **Learn To Schedule (*LTS*) [12].** The state-of-the-art CLS algorithm which uses deep reinforcement learning to train the NN towards a lower stalling ratio. However, it ignores the trade-off between cost and performance. We use the trained LTS model provided by Zhang et al.

B. Zwei vs. State-of-the-art Scheduling Methods

As shown in Figure 11(c), we find that Zwei stands for the best scheme among the candidates. For example, comparing with recent heuristics, such as the weighted round-robin and E2 method, Zwei not only reduces the stalling ratio by 15.54%-17.14% but also decreases the overall costs by 16.97%-17.42%. Specifically, Zwei reduces the overall costs by 22% compared with state-of-the-art learning-based method LTS and decreases over 6.5% in terms of the overall stalling ratio. The reason is that the vanilla LTS algorithm takes the weighted-sum-based combination function as the reward, while the function can hardly give clearer guidance for the optimized algorithm. Moreover, we offline computed the optimal policy w.r.t the original reward function provided by the LTS paper. Comparing the performance of Zwei with the optimal strategy, we observe that both optimal policy and Zwei are on the Pareto front. Zwei consumes less pricing costs than the optimal policy since the requirement is to *minimize the cost first*. Moreover, we present the training process in Figure 11(d). As shown, Zwei converges in less than 600 epochs, which needs about 2 hours. It's worth noting that Zwei also experienced two stages on the CLS task. The first stage ranges from 0 to 100 epochs, and we can see the goal of Zwei is to minimize the cost without considering the number of stalling ratios. For the rest of the stage, we find that Zwei attempts to reduce the number of stalling ratios, and at the same time, the cost converges to a steady-state. Such observation also proves that Zwei learns the strategies by following the given requirements.

VI. Case III: Real-Time Communication

To better understand how Zwei performs in the RTC task, we develop a faithful packet-based network emulation testbed, train Zwei via various real-world network traces, and validate Zwei with scale. As expected, results also prove the superiority of Zwei against existing methods.

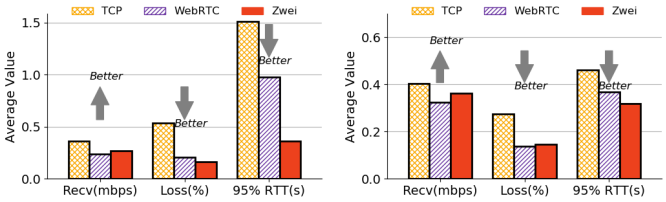
RTC Overview. The RTC system contains a sender and a receiver, and its transport protocol mainly consists of two channels: the streaming channel and the feedback message channel. In the beginning, the sender sends the video packets, denoted as a packet train to the receiver. The receiver then feeds the network status observed back to the sender. Based on this information, the sender will select a proper bitrate for the next period. We place Zwei as the RTC control module on the sender-side.

A. Detailed Implementation

Network Simulator Setup. We implement a faithful network simulator inspired by the queuing theory and several recent state-of-the-art congestion control platforms [36]. During training, we first randomly pick one network trace from the network trace dataset and uniformly set the basic RTT, loss ratio, and queue length for the selected trace. Then we adopt the network simulator to compute some metrics at the packet level, such as delay gradient, deliver rate, receive rate. Finally, we integrate the collected metrics every 100 milliseconds.

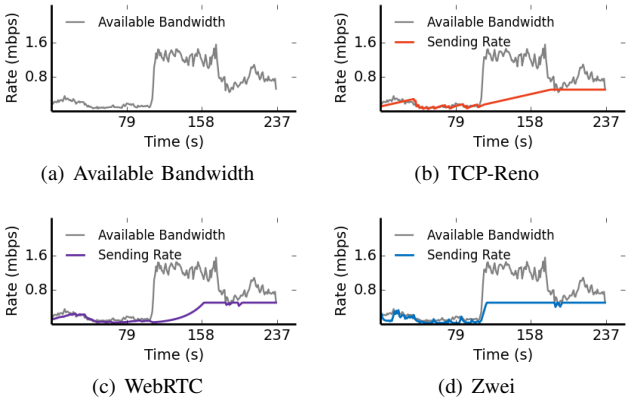
NN Architecture Overview. We take past sending rate, past receive rate, delay gradient, round-trip time (RTT), and loss ratio as the input. Besides, we output the NN as 11-dims vector to control the sending rate, which represents {-0.4, -0.3, -0.2, -0.1, -0.05, 0.0, 0.05, 0.1, 0.2, 0.3, 0.4}. Inspired by the additive-increase, multiplicative-decrease (AIMD) algorithm, we take different logic to increase or decrease the sending rate. Zwei increases the sending rate r_t as $r_t = r_t + r_{max} \times a_t$, and reduces the sending rate via $r_t = r_t \times (1 - a_t)$. For the NN representation, Zwei leverages Conv-1D layers to extract

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia



(a) HSDPA (b) WiFi

Fig. 12. The comparison of Zwei and WebRTC as well as TCP on both HSDPA and WiFi network datasets.



(a) Available Bandwidth (b) TCP-Reno

(c) WebRTC (d) Zwei

Fig. 13. The comparison of Zwei and existing methods over the HSDPA network trace.

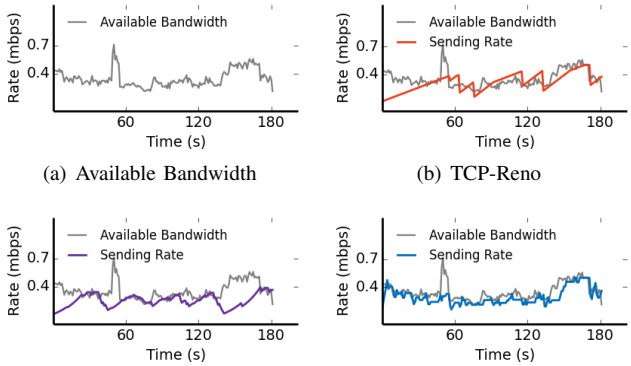
features for each metric, and employ a FC-layer to output two networks. Here we set past sequences k as 10.

We bound the sending rate to 100kbps-4mbps. We apply the combination of 1D-CNN and Fully connected layers to extract the feature and then merge it into a vector. At first, we use 1D-CNN with 64 filters and filter size=4 for extracting sequence features. Then the features will be merged by a combination layer. Subsequently, we leverage a 64-dim, fully connected layer to down-sample the collected features. Finally, the NN has two outputs i) the policy network, representing as the 11-dim vector with the Softmax active function, and ii) the value network with the Tanh active function.

Requirements for RTC tasks. In this RTC task, we aim to minimize the one-way delay and loss ratio and maximize the sending rate. Moreover, considering that measuring the one-way delay metric is required to modify the transport protocol, while the basic algorithm TCP-Reno fails to measure it in practice, we further leverage the RTT metric to take the place of the one-way-delay metric. Inspired by previous work [36], we adopt 95 percentile RTT that must be experienced between a sender and receiver.

RTC Baselines. We compare Zwei with several proposed heuristic methods, including:

1) ***TCP-Reno* [4].** The most popular congestion control algorithm on the Internet. The key insight of the proposed algorithm is the additive-increase, multiplicative-decrease (AIMD) method, which means a linear increase of the congestion window with an exponential reduction when congestion is detected. We adopt TCP-Reno with *fast recovery*.



(a) Available Bandwidth (b) TCP-Reno

(c) WebRTC (d) Zwei

Fig. 14. We also compare Zwei with existing methods over the trace collected from the real-world WiFi scenario.

2) ***WebRTC* [13].** State-of-art RTC scheme on both academics and industry. More precisely speaking, WebRTC leverages delay-based and loss-based modules to jointly control the rate adaption problem in the real-time scenario, where its parameters have been carefully tuned for almost one decade. In this work, we adopt WebRTC from its open-sourced repository.

B. Zwei vs. RTC Baselines

In this experiment, we compare Zwei with existing proposed heuristic methods (§VI-A) on various network conditions such as 3G and WiFi. We collect average receiving bitrate, 95 percent round-trip-time (RTT) as well as average loss ratio every 1 second. Results are reported in Figure 12. We demonstrate that Zwei improves 11.34% on average receiving bitrate, reduces 20.49% in terms of the loss ratio, and decreases 62.95% on 95-percent RTT compared with WebRTC under HSDPA dataset. Other results on the WiFi network conditions show that Zwei also outperforms WebRTC, with the improvements on sending rate of 12.08% and decreasing in 95% RTT of 13.59%. The comparison of Zwei and TCP-Reno illustrates that Zwei can effectively detect the congestion signal and dynamically adjust the sending rate to avoid the high loss ratio and RTT.

To better understand the performance of the selected methods, we report two representative results from the validation dataset and compare the sending rate curve of Zwei with TCP and WebRTC. As demonstrated in Figure 13 and Figure 14, we observe that Zwei can accurately estimate the bottleneck of the session since it always sends video streaming with high bitrates while also avoids congestion events. For instance, Figure 13 illustrates how the proposed algorithms perform over the HSDPA network conditions. The HSDPA network trace is quite challenging, since the low up-link capacity and the high variance of available bandwidth. We can see that the typical TCP congestion control algorithm TCP-Reno always obtains a higher receiving rate, while its exploration process uses fixed rules or strategies, which thereby often exceeds the rated transmission rate. On the contrary, the policy of WebRTC can better adapt to the 3G's network fluctuation. However, due to the strategy dependence on the estimated delay gradients, its overall performance is slightly conservative as well as fails to achieve a higher receiving rate. While Zwei's strategy is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia

different from previous methods: On the one hand, in the non-stationary network condition, it dynamically switches the sending rate for avoiding congestion events. On the other hand, in the stationary network environments, it quickly detects the maximum sending rate and maintains the transmission rate until the congestion event happens. Figure 14 shows the same observation in comparison of the performance over HSDPA network traces.

VII. Beyond Zwei: Practical Analysis

Considering that Zwei is impractical in the real-world since Zwei has to roll out from the same starting point, in this section, we further investigate the probability to make the proposed scheme more practical, i.e., training Zwei in the vanilla no-regret reinforcement learning settings. Such settings only allow the environment with the same setting to appear once, which are quite common in the real world.

A. Methodology

As we mentioned before, Zwei's *battle competition* process uses a Rule to estimate the win rate by the trajectories which are collected from the same environment settings. More detailed explanation is listed in Eq 17, where π_θ is the current policy, g_i is the Gumbel(0,1) distribution sampled by the agent i, and $\texttt{Env}$ represents the given environment. We can immediately see that the two trajectories T_i and T_j must be sampled from the environment $\texttt{Env}$ with the same settings. However, in the real world, the same environment is unlikely to appear more than once. In other words, real-world reinforcement learning is often considered as a no-regret exploration-exploitation in episodic Markov decision processes rather than regret learning in simulated and repeated environments.

$$o_i^j = \phi[T(\underbrace{\pi_\theta, g_i}_{Action}, \texttt{Env}), T(\underbrace{\pi_\theta, g_j}_{Action}, \texttt{Env})]. \tag{17}$$

Followed by this step, we modify Eq. 17 to Eq. 18 for varying traditional RL settings, in which $\texttt{Env}_i$ and $\texttt{Env}_j$ are different environments initialized by agent i and j. For example, in the ABR scenario, the two environments are represented as two different video descriptions and network traces. Thus, as described in Eq. 19, one intuitive idea is to optimize Zwei robustly is to use a constraint on the similarity $Dist$ between the $\texttt{Env}_i$ and the $\texttt{Env}_j$, where $Dist(\cdot)$ denotes a function to measure the dissimilarity.

$$o_i^j = \phi[T(\underbrace{\pi_\theta, g_i}_{Action}, \mathbf{Env}_i), T(\underbrace{\pi_\theta, g_j}_{Action}, \mathbf{Env}_j)]. \tag{18}$$

$$s.t. Dist(\mathbf{Env}_i || \mathbf{Env}_j) \leq \zeta. \tag{19}$$

Here ζ is a hyper-parameter that controls whether the current comparison is acceptable or not. The higher ζ means low variance but inaccurate win rate estimation, as the lower ζ yields high variance but accurate win rate estimation. Thus, how to select a proper ζ for each task is also a non-trivial problem for training Zwei better. We will discuss the choice of ζ for the ABR task in §IV.

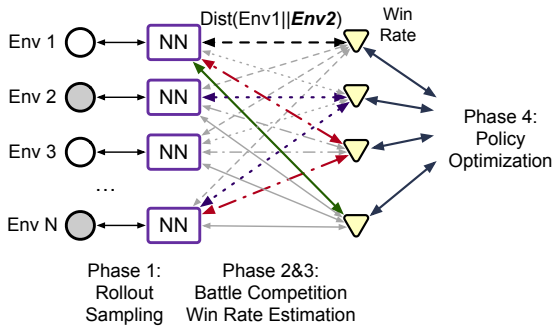


Fig. 15. Zwei$^+$ Overview. The framework is mainly composed of *four* phases: rollout sampling, battle competition, win rate estimation, and policy optimization.

B. Framework Overview

Inspired by the key ideas of deploying Zwei in the conventional no-regret RL setting, we propose a novel real-world self-play RL framework called Zwei$^+$. The system overview of Zwei$^+$ is shown in Figure 15, which also mainly consists of 4 phases, i.e., rollout sampling, battle competition, win rate estimation, as well as policy optimization. The overall training process is shown as follows.

Phase 1: Rollout Sampling. Followed by the traditional RL parallel training process, we rollout N different trajectories T_n for each agent respectively. For each episode, the Zwei$^+$'s learning agent meets various environments independently. Here please note that the policy π_θ of each agent is the same.

Phase 2: Battle Competition. Given two trajectories T_i and T_j collected by different agents i and j, we have to compute the similarity between that two trajectories for avoiding the inaccurate win rate estimation (§VII-A). The result of battle competition on T_i and T_j will be stored if the computed similarity lower than the threshold ζ. Otherwise, it will be dropped if the computed similarity higher than threshold ζ.

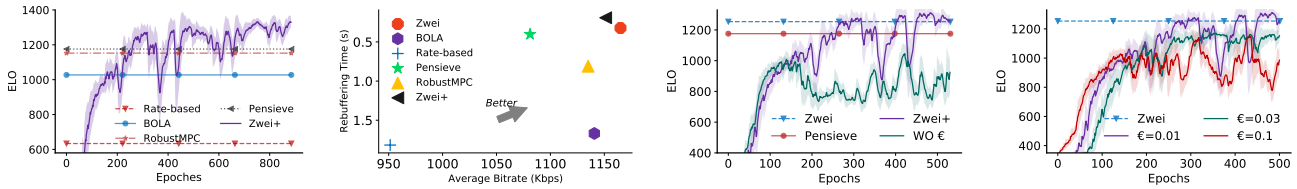
In the rest of the phases, we use the original win rate estimation process for estimating Zwei$^+$'s win rate and employ Dual-PPO [10] to optimize the NN since the goals of that two-phase are the same.

C. Evaluation in ABR Scenario

In this part, we consider applying Zwei$^+$ in the ABR scenario for evaluating the performance. The key reason is the ABR task is more popular and easy to follow compared with the RTC task, and in the meanwhile, its environment is also more complicated compared with the CLS task. We now explain the detailed evaluation methodology, including similarity function design, experimental settings as well as evaluation results.

Similarity Function Design. One of the important challenges in deploying Zwei$^+$ on ABR tasks is to determine how to measure the similarity between environments. In the ABR scenario, the environment is often constructed by a set of network traces and videos. Recent studies indicated that the throughput traces can be characterized by a specific mean (μ) and variance (σ) [37]. Hence, we extract the mean and variance for each trace, where the trace is collected by the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2021.3063620, IEEE Transactions on Multimedia



(a) Zwei$^+$ vs. ABRs (b) Zwei$^+$ Performance Details (c) Zwei$^+$ with/without ζ (d) Zwei$^+$ with different ζ

Fig. 16. Comparing Zwei$^+$ with state-of-the-art ABR approaches, including RB, BOLA, etc.

ABR player itself. In detail, we compute the similarity with the following steps:

1) In common, the network trace is represented as $\{C_k, D_k\}$, in which C_k is a vector that stored the throughput measured by chunk $[k-N+1, k]$, and D_k means the download time for each chunk $[k-N+1,k]$, and N=8 is the sequence length. We employ the weighted mean $\bar{C}$ and variance $Var(C)$ of the weighted mean for extracting mean and variance features.
2) Having estimated the weighted mean and variance of throughput traces, we then leverage rooted mean square error (RMSE) to compute the distance between the two $\{\mu, \sigma^2\}$ pairs. Here please note that we can use another type of formulation to evaluate the distance.
3) Finally, we use a threshold ζ to *filter* the proper network trace into comparison. The competition result will be stored if only the calculated $Dist(.)$ is lower than ζ. In this work, we set ζ as 0.01. We further implement an experiment to investigate the influence of ζ on the proposed method.

Zwei$^+$ Training Details. Zwei$^+$ also adopts multi-agent parallel training technologies, as it leverages 12 agents for collecting trajectories from various network traces and utilizes 1 central agent to compute win rate w.r.t the collected samples. Here we apply OpenMP [38] for accelerating the win rate estimation on the central agent. We train Zwei$^+$ to minimize rebuffering time while maximizing video bitrates (rebuffer, bitrate). The training time lasts over 24 hours, which is 6-times slower than the vanilla Zwei. The key reason is Zwei$^+$ has to collect more samples than the original Zwei for finding the trajectories collected from a similar environment.

Zwei$^+$ vs. Baselines. We compare Zwei$^+$ with various existing methods, including RB [31], BOLA [17], RMPC [15], Pensieve [6] and Zwei. The evaluation results are reported in Figure 16, where the ABR's performance are computed by Elo-ratings [11]. Two key observations can be drawn from Figure 16(a). First, although Zwei$^+$ lacks time efficiency compared with the vanilla Zwei method, it has also converged within 300 steps, which rivals the previously proposed approach. Second, we also observe that the performance gain in Elo rating between Zwei$^+$ and existing ABR baselines is approximately 127%, 40%, 25%, and 23% for Rate-based, BOLA, RMPC, and Pensieve, respectively. Such observation confirms that Zwei$^+$ has the abilities to learn the strategies from the real-world ABR environments, aka no-regret RL settings since our proposed similarity function design (§VII-C) enables Zwei$^+$ for better merging the samples collected from the similar environment. Moreover, we report the average bitrate and total rebuffering time for each ABR algorithm in Figure 16(b). Zwei$^+$ reduces the total rebuffering time of 89.48%, 88.55%, 76.41%, 52.47%, 40.67% (Rate-based, BOLA, RMPC, Pensieve, and Zwei), as well as increases the average bitrate within each stream of 21.03%, 0.96%, 1.41%, 6.48% (Rate-based, BOLA, RMPC, Pensieve), except with respect to the vanilla Zwei (-1.2%). Here it's notable that both Zwei and Zwei$^+$ have achieved Pareto front performance, while Zwei$^+$ performs better according to the requirement (See Alg. 2).

Zwei$^+$ with different ζ. As we have mentioned before, it's critical to take a proper threshold ζ for measuring the similarity between environments. We make two experiments in this part. First, we set up several experiments to discuss if it's necessary to apply the similarity function for Zwei$^+$. As shown in Figure 16(c), we compare Zwei$^+$ with that of using the function and without using that function. At the same time, we add two baselines, i.e., Zwei and Pensieve. Results illustrate that, without using any similarity function to separate the collected trajectories, the trained policy can hardly converge to optimal ABR decisions (see the green curve). By contrast, Zwei$^+$ leverages the function that enables the selection of only optimal decisions during the ABR process, and especially, rivals or performs recent state-of-the-art ABR algorithms. Thus, we find that a good similarity function is non-trivial for optimizing Zwei in the no-regret RL environment. Second, Figure 16(d) demonstrates the Elo-ratings of Zwei$^+$ with different ζ, including the original Zwei, $\zeta = 0.1$, $\zeta = 0.03$, and $\zeta = 0.01$. We see that with the decreasing of ζ, the overall Elo-rating of Zwei$^+$ will achieve the higher score. In particular, Zwei$^+$ with $\zeta = 0.01$ can finally obtain the highest Elo-ratings, which even rivals the vanilla Zwei. Hence, we believe that $\zeta = 0.01$ represents the best parameters for this ABR work.

VIII. Related Work

A. Heuristic Methods

Adaptive Bitrate Video Streaming. Heuristic-based ABR methods adopts throughput prediction (E.g., FESTIVE [31]) or buffer occupancy control (E.g., BOLA [17]) to choose the proper bitrate for the ABR task. However, such approaches suffer from either inaccurate bandwidth estimation or long-term bandwidth fluctuation problems. Then, MPC [15] picks next chunks' bitrate by jointly considering throughput and buffer occupancy. Nevertheless, MPC is sensitive to its parameters since the it relies on well-understanding different network conditions. Oboe [19] even proposes an auto-tuning

method to tune the traditional heuristic methods to achieve better performance in different network settings. Moreover, Lu et al. [39] investigates how video-quality information can be exploited by HTTP-based adaptive streaming clients in their rate adaptation schemes.

Crowd-sourced Live Streaming. The details of how a particular live-streaming platform chooses CDNs or schedules users to different CDNs are uncertain. However, through preliminary measurements, it is widely accepted that the strategies are largely statically configured (e.g., [35]). In recent years, dynamic scheduling across different CDNs has received more attention. [40] are model-based methods which update their model through real time measurement and predict the performance of CDN candidates for the next-time. However, they are significantly affected by the accuracy of modeling. [5] uses E2 method to replace traditional model-based methods, which is the state-of-the-art algorithm. Though E2 does not use any pre-modeling and only make decisions in real-time way, it underutilizes the states and time sequence information.

RTC Rate Adaption. Rate adaption methods are mainly composed of loss-based approach [4], attempting to increase bitrate till packet loss occurs, and delay-based approach [41], which attempts to adjust sending rate to control the transmission delay. However, recent methods are sensitive to network conditions. Thus, model-based approach, such as WebRTC [13], have been proposed to control the bitrate based on previous status observed, including past sending rate, receiving rate, delay gradients, as well as loss ratio.

In short, heuristic-based methods require careful tuning their parameters. Moreover, such methods are often myopic, one-shot, and only consider instant rewards.

B. Learning-based Schemes

In the adaptive video streaming scenario, Mao et al. [6] develop an ABR algorithm called Pensieve. Pensieve leverage DRL method to generate a strategy towards higher reward feedback, where the reward function is represented as the simple weighted sum of several critical factors. Moreover, several schemes combine learning-based and model-based approaches. For example, Bentaleb et al. propose AMP that encompasses techniques for bandwidth prediction and model auto-selection specifically designed for low-latency live steaming with chunked transfer encoding [37]. Stick is a harmonious fusion approach which fuse the learning-based and the conventional buffer-based approach [42]. While these studies are often optimized towards a specific linear-based objective function. The only study that is similar to our proposed framework is Tiyuntsong [8]. Tiyuntsong optimizes itself towards a rule or a specific reward via the competition with two agents under the same network condition. However, Tiyuntsong lacks training stability that makes it difficult to be deployed in the real-world.

In the crowd-sourced live streaming scheduling field, LTS [12] is the first DRL-based scheduling approach that outperforms previously proposed CLS approaches. Furthermore, the authors propose FAST-LTS [43] to balance the learning efficiency and computational cost. Specifically, they employ end-to-end learning-based methods to schedule viewer. However, since these methods all use QoS as the optimization objective, they seldom consider the correlation between the stalling ratio and the additional cost affected by the workload.

Furthermore, in the real-time video transmission services, QARC [7] is a DRL-based approach that optimizes a NN towards higher perceptual video quality and less stalling events. Meanwhile, OnRL trains the NN online for avoiding the simulation-to-reality gap [44]. Nevertheless, we also find that the performance of QARC and OnRL is quite sensitive to parameter settings of the reward function.

IX. Conclusion

Off-the-shelf learning-based video transmission techniques optimize themselves towards the linear-combination of several weighted objectives with mutual restriction rather than deterministic requirements, which might finally generalize a strategy that violates the original demand. We propose Zwei, a self-play reinforcement learning framework that utilizes the comparison outcome between several samples to enhance itself towards a better *win rate*. We implement and evaluate Zwei over various video transmission tasks. Results show that Zwei outperforms baselines with the improvements of more than 22% in terms of three fundamental video transmission tasks (32.24% on Elo score over the adaptive streaming, 22% on stalling ratio over the crowd-sourced live streaming, and 11.34% on average receiving rate over real-time communication). Furthermore, leveraging a specific battle competition phase, we extend Zwei to perform well in the reinforcement learning with no-regret settings.

Acknowledgement. We thank the TMM reviewers for the valuable feedback. This paper extends [45] by adding video transmission scenarios and proposes novel method which improves Zwei to work in the no-regret learning task.

References

[1] Cisco, "Cisco visual networking index: Forecast and methodology, 2016-2021," 2017. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[2] C. Lefelhocz, B. Lyles, S. Shenker, and L. Zhang, "Congestion control for best-effort service: why we need a new paradigm," *IEEE Network*, vol. 10, no. 1, pp. 10–19, 1996.

[3] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *SIGCOMM'14*, vol. 44, no. 4, 2015.

[4] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of tcp reno and vegas," in *INFOCOM'99*, vol. 3, 1999.

[5] J. Jiang, S. Sun, V. Sekar, and H. Zhang, "Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation." in *NSDI*, vol. 1, 2017.

[6] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.

[7] T. Huang, R.-X. Zhang, C. Zhou, and L. Sun, "Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 1208–1216.

[8] T. Huang, X. Yao, C. Wu, R.-X. Zhang, and L. Sun, "Tiyuntsong: A self-play reinforcement learning approach for abr video streaming," *arXiv preprint arXiv:1811.06166*, 2018.

[9] T.-Y. Huang, C. Ekanadham, A. J. Berglund, and Z. Li, "Hindsight: Evaluate video bitrate adaptation at scale," in *MMSys'19*, ser. MMSys '19. New York, NY, USA: ACM, 2019, pp. 86–97. [Online]. Available: http://doi.acm.org/10.1145/3304109.3306219

[10] D. Ye, Z. Liu, M. Sun, B. Shi, P. Zhao, H. Wu, H. Yu, S. Yang, X. Wu, Q. Guo *et al.*, "Mastering complex control in moba games with deep reinforcement learning," *arXiv preprint arXiv:1912.09729*, 2019.

[11] A. Elo, *The rating of chessplayers, past and present*. Arco Pub., 1978. [Online]. Available: https://books.google.com/books?id=8pMnAQAAMAAJ

[12] R.-X. Zhang, T. Huang, M. Ma, H. Pang, X. Yao, C. Wu, and L. Sun, "Enhancing the crowdsourced live streaming: a deep reinforcement learning approach," in *NOSSDAV'19*, 2019.

[13] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (webrtc)," in *MMSys'16*, 2016.

[14] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over http," *IEEE Communications Surveys & Tutorials*, 2018.

[15] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *SIGCOMM'15*, 2015.

[16] Y. Sun and et al., "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *SIGCOMM 2016*, 2016.

[17] K. Spiteri *et al.*, "Bola: Near-optimal bitrate adaptation for online videos," in *INFOCOM'16*, 2016.

[18] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, 2013.

[19] Z. Akhtar, P. Adria, M. Mirza *et al.*, "Oboe: auto-tuning video abr algorithms to network conditions," in *SIGCOMM 2018*, 2018.

[20] X. Zhang, Y. Ou, S. Sen, and J. Jiang, "Sensei: Aligning video streaming quality with dynamic user sensitivity," *arXiv preprint arXiv:2008.04687*, 2020.

[21] J. Jiang and S. Sen, "A new abstraction for internet qoe optimization," *arXiv preprint arXiv:2008.04128*, 2020.

[22] D. Balduzzi, M. Garnelo, Y. Bachrach, W. M. Czarnecki, J. Perolat, M. Jaderberg, and T. Graepel, "Open-ended learning in symmetric zero-sum games," *arXiv preprint arXiv:1901.08106*, 2019.

[23] D. Silver, A. Huang, C. J. Maddison *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, 2016.

[24] B. Kim, K. Lee, S. Lim, L. P. Kaelbling, and T. Lozano-Pérez, "Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds." in *AAAI*, 2020, pp. 9916–9924.

[25] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[26] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*. US Government Printing Office, 1948, vol. 33.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML'17*, 2016.

[29] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Comyco: Quality-aware adaptive video streaming via imitation learning," *arXiv preprint arXiv:1908.02270*, 2019.

[30] R. Rassool, "Vmaf reproducibility: Validating a perceptual practical video quality metric," in *Broadband Multimedia Systems and Broadcasting (BMSB), 2017 IEEE International Symposium on*, 2017.

[31] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *TON*, vol. 22, no. 1, 2014.

[32] K. Spiteri, R. Sitaraman, and D. Sparacio, "From theory to practice: improving bitrate adaptation in the dash reference player," in *Proceedings of the 9th MMSys*, 2018.

[33] R. Coulom, "Whole-history rating: A bayesian rating system for players of time-varying strength," in *International Conference on Computers and Games*. Springer, 2008, pp. 113–124.

[34] M. F. B. Report, "Raw data measuring broadband america 2016," https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016, 2016, [Online; accessed 19-July-2016].

[35] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang, "Unreeling netflix: Understanding and improving multi-cdn movie delivery," in *INFOCOM'12*, 2012.

[36] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate record-and-replay for {HTTP}," in *ATC'15)*, 2015.

[37] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann, "Data-driven bandwidth prediction models and automated model selection for low latency," *IEEE Transactions on Multimedia*, 2020.

[38] R. Chandra, L. Dagum, D. Kohr, R. Menon, D. Maydan, and J. McDonald, *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[39] Z. Lu, S. Ramakrishnan, and X. Zhu, "Exploiting video quality information with lightweight network coordination for http-based adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 7, pp. 1848–1863, 2017.

[40] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, "Cfa: A practical prediction system for video qoe optimization." in *NSDI*, 2016, pp. 137–150.

[41] D. Rossi *et al.*, "Ledbat: The new bittorrent congestion control protocol." in *ICCCN*, 2010.

[42] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1967–1976.

[43] R.-X. Zhang, M. Ma, T. Huang, H. Pang, X. Yao, C. Wu, and L. Sun, "A practical learning-based approach for viewer scheduling in the crowd-sourced live streaming," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2s, pp. 1–22, 2020.

[44] H. Zhang, A. Zhou, J. Lu, R. Ma, Y. Hu, C. Li, X. Zhang, H. Ma, and X. Chen, "Onrl: improving mobile video telephony via online reinforcement learning," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, 2020, pp. 1–14.

[45] T. Huang, R.-X. Zhang, and L. Sun, "Self-play reinforcement learning for video transmission," in *NOSSDAV 2020*, 2020, pp. 7–13.

Tianchi Huang received his M.E degree in the Department of Computer Science and Technology in Guizhou University in 2018. Currently he is a Ph.D student in the Department of Computer Science and Technology at Tsinghua University, advised by Prof. Lifeng Sun. His research work focuses on the multimedia network streaming, including transmitting streams, and edge-assisted content delivery. He received Best Student Paper Awards presented by ACM Multimedia System 2019 Workshop.

Rui-Xiao Zhang received his B.E degree in Electronic Engineering Department in Tsinghua University in 2017. Currently, he is pursuing his Ph.D candidate in Department of Computer Science and Technology, Tsinghua University, China. His research interests lie in the area of content delivery networks, the optimization of multimedia streaming and reinforcement learning. He received Best Student Paper Awards presented by ACM Multimedia System 2019 Workshop.

Lifeng Sun received the B.S and Ph.D degrees in system engineering from National University of Defense Technology, Changsha, Hunan, China, in 1995 and 2000, respectively. He joined Tsinghua University since 2001. He is currently a Professor with the Computer Science and Technology Department of Tsinghua University, Beijing. Prof. Sun's research interests include the area of networked multimedia, video streaming, 3D/multiview video coding, multimedia cloud computing, and social media.