CLARA: A Constrained Reinforcement Learning Based Resource Allocation Framework for Network Slicing

Yongshuai Liu*, Jiaxin Ding†, Zhi-Li Zhang‡, Xin Liu*
*Department of Computer Science, University of California, Davis
†John Hopcroft Center for Computer Science, Shanghai Jiao Tong University
‡Department of Computer Science and Engineering, University of Minnesota
Emails: yshliu@ucdavis.edu, jiaxinding@sjtu.edu.cn, zhzhang@cs.umn.edu, xinliu@ucdavis.edu

Abstract—As mobile networks proliferate, we are experiencing a strong diversification of services, which requires greater flexibility from the existing network. Network slicing is proposed as a promising solution for resource utilization in 5G and future networks to address this dire need. In network slicing, dynamic resource orchestration and network slice management are crucial for maximizing resource utilization. Unfortunately, this process is too complex for traditional approaches to be effective due to a lack of accurate models and dynamic hidden structures. We formulate the problem as a Constrained Markov Decision Process (CMDP) without knowing models and hidden structures. Additionally, we propose to solve the problem using CLARA, a Constrained reinforcement LeArning based Resource Allocation algorithm. In particular, we analyze cumulative and instantaneous constraints using adaptive interior-point policy optimization and projection layer, respectively. Evaluations show that CLARA clearly outperforms baselines in resource allocation with service demand guarantees.

Index Terms—Resource Allocation, Network Slicing, 5G, Constraints, Deep Reinforcement Learning

I. INTRODUCTION

As mobile networks proliferate, we face a broad variety of services, including autonomous driving, industry 4.0, virtual/augmented reality, and the Internet of Things (IoT), etc. These services are characterized by heterogeneous performance, functional, security and operational requirements [1]–[4], which demand the network to embed more flexibility. Because of the lack of flexibility in existing networks, independent initiatives of dedicated infrastructure solutions are deployed: 3GPP has developed an IoT specific MAC that can co-exist with general purpose MAC [5]; the industry has deployed proprietary architectures for extreme reliability [6]. Such monolithic vertical developments are clearly highly expensive and inefficient.

Network slicing [7] is a promising solution for flexible resource provisioning in 5G and future networks. It creates multiple virtual instances, named network slices, over physical infrastructures enabled by network function virtualization (NFV) and software-defined networking (SDN). Different slices can accommodate different service demands, such as

Jiaxin Ding is the corresponding author.

1) Mobile broadband (high throughput/low latency); 2) massive IoT communication (massive connection/bursty traffic); 3) video surveillance system (high edge processing requirement); 4) autonomous driving (ultra reliability/low latency); 5) VR/AR/real-time gaming (low latency/ultra high throughput); 6) industry 4.0 automation (ultra reliability/security). Network slicing can take place at different places on the network, including core cloud, edge cloud, radio resource management, RAN processing, spectrum, and radio frontend. [7]–[9]. Generally, it aims to manage and allocate different resources to meet various service demands.

Network slicing is essentially a generalized resource allocation problem that manages heterogeneous resources to meet service demands in accordance with the dynamics of the complex network. It is essential to orchestrate network slices dynamically to maximize resource efficiency [10]. Nevertheless, resource allocation is a highly complex issue in network slicing, which traditional approaches cannot resolve efficiently or effectively.

First, traditional optimization methods only incorporate models with precisely known parameters, which is often impossible in practice, especially as 5G and future networks become increasingly complex, large, and diverse in service. It is further complicated by the constraints from the physical system and service requirements, such as latency requirements and service level agreements. The network environment is affected by the location, geographical properties (e.g., tall buildings, hills, highways, etc), mobility, etc., all difficult to measure and model. The precise network traffic of a time slot is not available, when the network slicing decision is made at the beginning of the time slot. Because of these reasons, it is very difficult to obtain accurate model parameters in real networks, on which traditional optimization approaches rely.

Additionally, traditional methods do not accommodate epistemic uncertainty, manifested as hidden structures in networks, resulting from a lack of knowledge and subsequent ability to explore and learn from the network. For example, a user who experiences a poor quality of service may decide not to use the service again or at a reduced frequency. Such hidden structures can significantly affect user experience and network performance but are usually not directly observed/modeled.

Facing these challenges, learning-based approaches are beneficial for addressing these challenges because they have the ability to explore and learn from a network without assuming the prior knowledge of accurate models. The industry has recognized machine learning (ML) as a core technology for future telecommunication networks, including 5G and beyond [11]. A growing number of recent research studies have demonstrated significant performance improvement using learning-based networks, e.g., [12]–[18]. However, they either only consider one step of optimization (e.g. multi-armed bandit [15]) or do not analyze the resource allocation problem with constraints imposed by the service requirements, which is crucial for network slicing.

In this work, we propose CLARA, a Constrained reinforcement LeArning based Resource Allocation framework for network slicing. Thanks to NFV, we can focus our resource allocation decisions on the virtualized resources. We first model the problem as a Constrained Markov Decision Process (CMDP) without knowing prior knowledge. The objective is to maximize a reward in a long run, e.g. throughput over time. At the same time, it is subject to a number of constraints. The constraints abstracted from system capacity limits and service requirements are formulated as two types of constraints, cumulative and instantaneous constraints. A cumulative constraint requires that the sum of a quality is within a certain limit, e.g., outage probability or average throughput, etc, while an instantaneous constraint requires that the quality needs to satisfy a condition in each time slot, e.g. resource limits and service latency requirement, etc. Instantaneous constraints can be further divided into explicit and implicit instantaneous constraints. An explicit constraint has a closed-form expression that can be numerically checked, e.g., transmission power and spectrum available, etc. An implicit constraint does not have an accurate closed-form formulation due to the complexity of the system, e.g., latency and interference level, etc.

We develop efficient reinforcement learning algorithms for network slicing under both cumulative and instantaneous constraints with theoretical analysis on the performance bound. Specifically, to deal with cumulative constraints, we propose our adaptive constrained RL algorithm improved over Interior-point Policy Optimization (IPO) [19]. For instantaneous constraints, we project a resource allocation decision generated by the reinforcement learning algorithm to its nearest feasible decision at the end of policy neural network [20], [21].

II. Network Slicing

A. System Description

We briefly describe the network slicing architecture with CLARA for resource allocation management, as shown in Fig. 1a. It is developed from the ETSI reference model [22] colored in blue, and our contribution is highlighted with green.

From bottom to top, Fig. 1a first shows that Virtualized Infrastructure Management (VIM) is in charge of the Network Functions Virtualization Infrastructure (NFVI), which converts the physical resources to virtual resources, through the Virtualization layer. Thereafter, Virtual Network Function (VNF) consists of the virtual resources, which is monitored, configured, and controlled by the VNF manager. These VNFs providing different service functionalities finally make up the Network Slice (NS). Because of NFV, we are able to allocate resources according to virtual resources. The Network Slicing Manager (NSM) is responsible for the initialization, configuration, and managing the life cycles of the network slices. The Orchestrator automatizes the management of the elements on the lower levels, communicates and coordinates all those control managers to schedule tasks and update states of the system. The Orchestrator is linked to Operations Support System/Business Support System (OSS/BSS) which provides the services. Our CLARA, a policy neural network trained with constrained RL algorithms, is in charge of providing solutions for the resource allocation on network slices to the NSM and the Orchestrator, and receive feedbacks to improve the policy on further decisions.

Fig. 1b shows CLARA in details. The service providers submit the service requirements to CLARA. The Network Slice Manager monitors the current system state and sends it to CLARA. Based on the system state and the service requirements, CLARA proposes the resource allocation plan to the NSM. The NSM configure the network slices with the proposed plan. This plan is passed to the VNF manager and lower level components to further map the virtual resources to physical resources. For each configuration in a decision time slot, the NSM monitors the system and network slices to measure the rewards(constraints) collected and send the rewards(constraints) to CLARA for further policy improvement.

B. Radio Access Network Slicing

To be clear, we consider how to apply CLARA for radio access scenario with hidden dynamics. Specifically, give a list of slices $1, ..., N$ sharing bandwidth B. CLARA learns a policy to give a bandwidth allocation $(b_1, ..., b_N)$ to maximize the throughput over time while satisfying certain constraints.

We simulate a scenario containing a Base Station (BS) with three types of services (i.e., Video, VoLTE, URLLC). Each service has a random number (max 100 for each type) of users located surrounding the BS. The users arrive according to Poisson distributions (initial mean value 50, 50, and 10 respectively). User requests of VoIP and video services take the parameter settings of VoLTE and video streaming models, while URLLC service takes the parameter settings of FTP 2 model [23], the same setting seen in [24], similar settings also in [17]. The parameters are described in Table I based on their respective streaming model. This BS is fixed and given a certain bandwidth (100 Mbps). It limits the total available bandwidth. System operators slice the network by assigning bandwidth to each type of user (a slice).

Time slots are employed in the system (one second in each slot). At the start of each time slot, the BS determines the amount of bandwidth b_i will be allocated to i (i being the user type: Video, VoLTE, and URLLC) according to the number of active users in each slice. By taking t_i to represent the traffic demand for each slice, the throughput for each user type is

(a) The network slicing architecture with CLARA

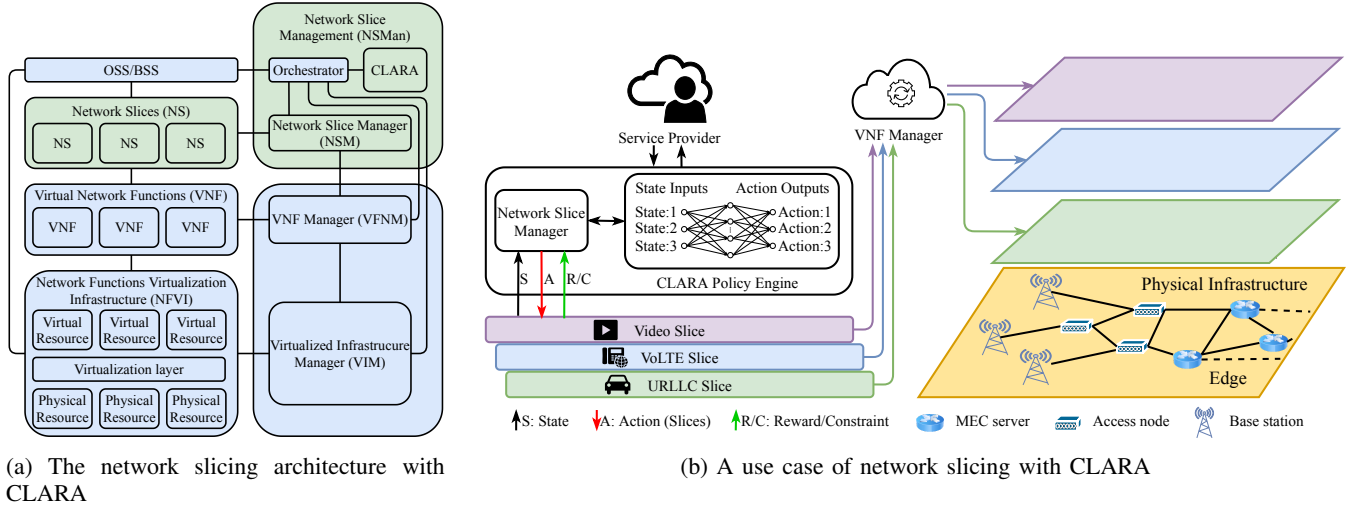(b) A use case of network slicing with CLARA

Fig. 1: Network slicing with CLARA

$\min(b_i, t_i)$. CLARA aims to maximize the total throughput over time.

Each type of user is assigned a dissatisfaction ratio indicating how dissatisfied they are with their service. We set it as $1 - \frac{\min(b_i, t_i)}{t_i}$ in our simulation. The dissatisfaction ratio is 0 if the bandwidth assigned to the users exceeds or equals their demands. Otherwise, it will be $1 - \frac{b_i}{t_i}$. The overall dissatisfaction ratio over time should under a constraint. BS determines the latency l_i of each user type using a queue maintained by the BS. Our system employs the first-come-first-serve principle. In queued traffic packets, those not processed during the current time slot are forwarded to the next period. Latency is the total time needed to transmit a traffic packet and it cannot be easily formulated mathematically. In each step, the latency should under a limitation.

In addition, dynamic hidden structures exist regarding how users engage in the corresponding service depending on the service quality. In general, users depart and arrive at the network according to Poisson distributions with mean of λ_i and μ_i. The arrival rate λ_i is adaptive with the satisfaction ratio from the last time slot. In each time slot, λ_i is updated with $\lambda_i = 0.99 * \lambda_i + 0.01 * \lambda_i * \frac{b_i}{t_i}$. Furthermore, users may depart early if the service quality is unsatisfactory. Specifically, we assume that if the dissatisfaction ratio is 0, no one departs early. Otherwise, users depart with probability $1 - \frac{b_i}{t_i}$. **During the slicing process, we assume that these traffic demands and mobility patterns are unknown. This gives learning-based approaches, which incorporate exploration, an advantage over traditional methods which rely only on observed states.**

III. PROBLEM FORMULATION

In this section, we present the problem formulation for CLARA. Resource allocation in the network slicing is an optimization problem in essence. From the system operator's view point, the objectives include total throughput, Quality of Service (QoS), Quality of Experience (QoE), churn rate,

as well as operational revenue and cost, etc. At the same time, network slicing is subject to a number of constraints, including resource constraints (e.g., spectrum, power, computation, storage), user service performance constraints (e.g., latency, average data rate, QoS, QoE), security requirements (e.g., isolation and firewall) and service level agreement (SLA) requirements (e.g., service availability and reliability), etc.

A. Constrained Markov Decision Process (CMDP)

Mathematically, we formulate the problem as a CMDP, which is represented with the tuple $(S, A, R, C, P, \mu, \gamma)$. The network observations constitute the state set S, e.g. current network slice allocation, network load (e.g., the number of users and traffic demand), network status (e.g. cell conditions, neighboring cell interference level), etc.

The resource allocation decisions constitute the action set A, which depending on what level CLARA runs, could be admission control, computation and spectrum allocation, as well as network configurations, etc.

The reward of taking action a under state s is defined as the reward function $R : S \times A \times S \mapsto \mathbb{R}$, e.g., throughput and revenue, etc. The objective is to maximize the cumulative reward expectation. Similarly, the costs of taking action a under state s is defined as the cost functions, $C_i : S \times A \times S \mapsto \mathbb{R}$. There are m cost functions and each is under a constraint, e.g. service level agreement (SLA) and latency, etc. $P : S \times A \times S \mapsto [0, 1]$ is the unknown transition probability function, where $P(s'|s, a)$ is the transition probability from state s to s' taking action a. Last, $\mu : S \mapsto [0, 1]$ is the initial state distribution and γ is the discount factor, which can be different for reward and costs.

The actions are constrained by two types of constraints. A cumulative constraint requires that the cumulatively sum of a cost is within a certain limit, e.g., outage probability or average throughput, etc, while an instantaneous constraint requires that a cost needs to satisfy a condition in each time slot, e.g. resource limits and service latency requirement, etc.

Distribution	Initial number of users	Inter-Arrival Time	Packet Size
Video	Poisson [Mean=50]	Pareto [Exponential Para = 1.2, Mean= 6 ms, Max = 12.5 ms]	Truncated Pareto [Exponential Para = 1.2, Mean= 100 Byte, Max = 250Byte]
VoLTE	Poisson [Mean=50]	Uniform [Min = 0, Max =160ms]	Constant (40 Byte)
URLLC	Poisson [Mean=10]	Truncated Exponential [Mean = 180ms]	Truncated Lognormal [Mean = 2 MB, Standard Deviation = 0.722 MB, Maximum =5 MB]

TABLE I: Parameter for different types of user

CMDP	Radio Resource Slicing
State	Number of active users in each type: $(n_{Video}, n_{VoLTE}, n_{URLLC})$
Action	Sliced bandwidth to each type of user: $(b_{Video}, b_{VoLTE}, b_{URLLC})$
Reward function	Total throughput: $\min(b_{Video}, t_{Video}) + \min(b_{VoLTE}, t_{VoLTE}) + \min(b_{URLLC}, t_{URLLC})$
Explicit instantaneous constraint	Sum of sliced bandwidth: $b_{Video} + b_{VoLTE} + b_{URLLC}$
Implicit instantaneous constraint	Average of latency
Cost function	Dissatisfaction ratio: $1 - \frac{b_i}{t_i}$, (i =Video, VoLTE and URLLC)

TABLE II: Mapping from Radio Resource Slicing to CMDP

Instantaneous constraints can be further divided into explicit and implicit instantaneous constraints. An explicit constraint has a closed-form expression that can be numerically checked, e.g., transmission power and spectrum available, etc. An implicit constraint does not have an accurate closed-form formulation due to the complexity of the system, e.g., latency and interference level, etc. Such constraints have to be modeled or learned using existing data and/or during exploration. We define action a as feasible, if $a \in A$ satisfies all the constraints including both cumulative and instantaneous.

B. Mapping Network Slicing to CMDP

Mapping to the radio access scenario, as summarized in Table II, the state $s = (n_{Video}, n_{VoLTE}, n_{URLLC})$ is the number of users observed at the beginning of each time slot. We do not know the exact traffic demand generated in this time slot. The action $a = (b_{Video}, b_{VoLTE}, b_{URLLC})$ is the bandwidth allocation for each type of users. The reward $R(s,a)$ at each time slot is the total throughput $\min(b_{Video}, t_{Video}) + \min(b_{VoLTE}, t_{VoLTE}) + \min(b_{URLLC}, t_{URLLC})$.

Moreover, each type of users has a cumulative constraint, which is the expectation of cumulative dissatisfaction ratio. The corresponding cost function $C_i(s,a)$ in each step is the dissatisfaction ratio $1 - \frac{\min(b_i, t_i)}{t_{i_i}}$ for each type of user.

Last, the actions need to satisfy both the explicit and implicit instantaneous constraints. The explicit instantaneous constraint is the sum of allocated bandwidth, $b_{Video} + b_{VoLTE} + b_{URLLC}$, which must be less or equal to the total bandwidth (100 Mbps). The implicit instantaneous constraint is on the average latency of each type of user, which we cannot get a closed-form solution and needs to be learned. The average latency should be less or equal to a predefined limitation.

CLARA learns a policy π takes states as input and output actions. We denote a policy as π_θ to emphasize its dependence on the parameter θ (e.g., a neural network policy) and $\tau = (s_0, a_0, s_1, a_1...)$ is a trajectory, where $\tau \sim \pi_\theta$. The objective is to select a policy π_θ, which maximizes the discounted cumulative reward

$$J_R^{\pi_\theta} = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})], \quad (1)$$

while satisfying discounted cumulative constraints

$$J_{C_i}^{\pi_\theta} = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t C_i(s_t, a_t, s_{t+1})] \quad (2)$$

and instantaneous constraints.

Formally, the optimization problem is defined as

$$\underset{\theta}{\text{maximize}} \quad \max_{\theta} J_R^{\pi_\theta} \quad (3)$$

$$\text{subject to} \quad J_{C_i}^{\pi_\theta} \leq \omega_i, \text{for each i,} \quad (4)$$

$$C_j(s_t, a_t) \leq \epsilon_j, \text{for each j and t,} \quad (5)$$

The formulation and following CLARA algorithm can be applied to general network slicing problems, including radio access, edge, cloud, and core networks. Here, we explain CLARA on a radio resource slicing scenario derived from the work [24] and it can be extended to more general cases easily.

IV. PRELIMINARIES

To solve the above problem, we briefly review the preliminaries from previous work in this section.

A. Definition

For a state-action trajectory starting from state s, the value function of state s is

$$V_R^{\pi_\theta}(s) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|s_0 = s].$$

The action-value function of state s and action a is

$$Q_R^{\pi_\theta}(s,a) = \mathbb{E}_{\tau \sim \pi_\theta}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})|s_0 = s, a_0 = a],$$

and the advantage function is

$$A_R^{\pi_\theta}(s,a) = Q_R^{\pi_\theta}(s,a) - V_R^{\pi_\theta}(s). \quad (6)$$

In the CMDP, the corresponding values for the cumulative constraint cost functions are calculated in the same way. $V_{C_i}^{\pi_\theta}(s), Q_{C_i}^{\pi_\theta}(s,a), A_{C_i}^{\pi_\theta}(s,a)$, for each constraint cost function C_i, are defined by replacing reward function R above with C_i. In the following sections, to simplify the notation, we omit the subscripts of R and C_i if there is no ambiguity.

Let $\rho_{\pi_\theta}(s)$ be the discounted visitation frequencies

$$\rho_{\pi_\theta}(s) = \sum_{t=0}^{\infty} \gamma^t P(S_t = s),$$

where the actions are chosen according to π_θ.

The Kullback-Leibler (KL) divergence [25] of distribution Γ from Δ is defined as

$$D_{KL}(\Gamma, \Delta) = \sum_{x \in \chi} \Gamma(x) \log\left(\frac{\Gamma(x)}{\Delta(x)}\right).$$

We denote $D_{KL}^{max}(\pi, \pi') = \max_s D_{KL}(\pi(\cdot|s), \pi'(\cdot|s))$.

B. Policy Gradient Methods

Policy gradient [26] method is applied to find an optimal policy of an unconstrained Markov Decision Process (MDP) problem. It calculates the gradient of the objective Eq. (1),

$$\bigtriangledown J^{\pi_\theta} = \mathbb{E}_t[\bigtriangledown_\theta log\pi_\theta(a_t|s_t)A_t]$$

where π_θ is the current policy under parameter θ and A_t is the advantage function Eq. (6) at time step t. Thereafter, θ is updated as

$$\theta = \theta + \eta \bigtriangledown J^{\pi_\theta},$$

where η is the learning rate.

Trust Region Policy Optimization (TRPO) [27] is proposed to achieve monotonic improvement of the new policy based on the results of the previous policy. The objective is approximated with a surrogate function combined with the Kullback Leibler (KL) divergence shown as follows. We denote a local approximation $L_{\pi_\theta}(\pi_{\theta'})$ for $J^{\pi_{\theta'}}$ with π_θ as

$$L_{\pi_\theta}(\pi_{\theta'}) = J^{\pi_\theta} + \sum_s \rho_{\pi_\theta}(s) \sum_a \pi_{\theta'}(a|s)A^{\pi_\theta}(s,a). \quad (7)$$

The objective of TRPO is to maximize

$$L^{TRPO}(\theta) = L_{\pi_{\theta_o}}(\pi_\theta) - \frac{4\varepsilon^{\pi_{\theta_o}}\gamma}{(1-\gamma)^2}D_{KL}^{max}(\pi_{\theta_o}, \pi_\theta), \quad (8)$$

where π_{θ_o} is the old policy to improve, $\varepsilon^{\pi_{\theta_o}} = \max_{s,a} |A^{\pi_{\theta_o}}(s,a)|$. The reward improvement of the new policy obtained by solving the optimization is guaranteed by the following theorem.

Theorem 1. *[27]*

$$J^{\pi_\theta} \geq L_{\pi_{\theta_o}}(\pi_\theta) - \frac{4\varepsilon^{\pi_{\theta_o}}\gamma}{(1-\gamma)^2}D_{KL}^{\max}(\pi_{\theta_o}, \pi_\theta).$$

The following theorem provides a tighter bound than Theorem 1 to measure the difference between any two policies.

Theorem 2. *[28] For any policies π', π, the following bound holds:*

$$J(\pi') - J(\pi) \geq$$
$$\underset{\substack{s \sim \rho_{\pi_\theta} \\ a \sim \pi_{\theta'}}}{\mathbb{E}} [A^\pi(s,a)] - \frac{\gamma\varepsilon^{\pi'}}{(1-\gamma)^2}\sqrt{2D_{KL}^{\max}(\pi', \pi)},$$

where $\varepsilon^{\pi'} = \max_s |\mathbb{E}_{a \sim \pi'}[A^\pi(s,a)]|$.

Proximal Policy Optimization (PPO) [29] is a heuristic algorithm with the same intuition as TRPO to formulate the problem with a first-order surrogate optimization to reduce the complexity, maximizing

$$L^{CLIP}(\theta) =$$
$$\underset{\substack{s \sim \rho_{\pi_{\theta_o}} \\ a \sim \pi_{\theta_o}}}{\mathbb{E}} [\min\{r(\theta), \text{clip}(r(\theta), 1 - \varepsilon, 1 + \varepsilon)\} A^{\pi_{\theta_o}}(s,a)], \quad (9)$$

where $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_o}(a|s)}$, $\text{clip}(\cdot)$ is the clip function and $r_t(\theta)$ is clipped between $[1 - \epsilon, 1 + \epsilon]$.

V. Constrained Reinforcement Learning

A. Cumulative Constraints

To deal with the cumulative constraints in the CMDP, we develop our method built upon our previous work Interior-point Policy Optimization (IPO) [19]. IPO augments the objective function of PPO (Eq. (9)) with logarithmic barrier functions without theoretical guarantee for the policy improvement. However, in this work, we incorporate the TRPO objective with logarithmic barrier functions and provide theoretical policy improvement guarantees. Moreover, IPO uses a fixed hyperparameter t for the logarithmic barrier function, while we further propose practical algorithms in an adaptive manner.

The way that we deal with constraints is inspired by the interior-point method [30]. For completeness, we briefly review the related background. Intuitively, the constrained optimization problem is reduced to an unconstrained one by adding indicator functions as penalty such that 1) if a constraint is satisfied, zero penalty is added to the objective function, and 2) if the constraint is violated, the penalty is negative infinity, ideally.

A logarithmic barrier function is a differentiable approximation of the indicator function, defined as

$$\phi(\widehat{J}_{C_i}^{\pi_\theta}) = \frac{\log(-\widehat{J}_{C_i}^{\pi_\theta})}{t},$$

where t is a hyperparameter, and $\widehat{J}_{C_i}^{\pi_\theta} = J_{C_i}^{\pi_\theta} - \omega_i$, to simplify the notation. We get a better approximation for the indicator function with a higher t.

Now we take $L^{TRPO}(\theta)$ in Eq. (8) as our objective with cumulative constraints, that is,

$$\max_\theta \ L^{TRPO}(\theta),$$
$$s.t. \quad \widehat{J}_{C_i}^{\pi_\theta} \leq 0. \quad (10)$$

We reduce the above optimization problem to an unconstrained optimization by augmenting the objective with the logarithmic barrier functions for constraints. Our objective becomes

$$\max_\theta \ L^{TRPO}(\theta) + \sum_{i=1}^m \phi(\widehat{J}_{C_i}^{\pi_\theta}). \quad (11)$$

In the following subsection, we show the theoretical guarantee of the policy performance with this objective function.

B. Policy Performance Bounds

Theorem 3. *The maximum gap between the optimal value of Eq. (10) and the optimal of Eq. (11) is bounded by $\frac{m}{t}$, where m is the number of constraints and t is the hyperparameter of logarithmic barrier function.*

This theorem provides a lower bound for the performance guarantee for incorporating the logarithmic barrier function into the TRPO objective. The proof is similar as the proof of the guarantee for the PPO objective in Theorem 1 of [19]. The theorem shed lights upon the effects from the hyperparameter t, since a larger t provides a better approximation of the original objective, which is also validated in the empirical experiments in IPO [19]. In practice, a larger t can result in a large fluctuation near the boundary. Hence, there is a tradeoff in choosing this hyperparameter t.

Theorem 4. *Assume π_θ is obtained by solving Eq. (11) based on a previous feasible policy π_{θ_o}. The following bounds holds*

$$J_R^{\pi_\theta} - J_R^{\pi_{\theta_o}} \geq$$
$$- \sum_{i \in \Phi} \frac{1}{t} \log \left(1 - \underset{\substack{s \sim \rho_{\pi_{\theta_o}} \\ a \sim \pi_\theta}}{\mathbb{E}} \left[\frac{1}{\psi^{\pi_{\theta_o}}} A_{C_i}^{\pi_{\theta_o}}(s,a) \right] + \frac{\sqrt{2}\gamma\varepsilon_{C_i}^{\pi_\theta}}{\psi^{\pi_{\theta_o}}(1-\gamma)^2} \sqrt{D_{KL}^{\max}(\pi_\theta, \pi_{\theta_o})} \right),$$

where $\Phi = \{i : J_{C_i}(\pi_\theta) < J_{C_i}(\pi_{\theta_o})\}$,

$$\varepsilon_{C_i}^{\pi_\theta} = \max_s \left| \mathbb{E}_{a \sim \pi_\theta}[A_{C_i}^{\pi_{\theta_o}}(s,a)] \right|,$$

$$\psi^{\pi_{\theta_o}} = \min_i (\omega_i - J_{C_i}^{\pi_{\theta_o}}).$$

Proof. Since π_θ is obtained by maximizing the objective in Eq. (11) and π_{θ_o} is a feasible policy, we have

$$J_R(\pi_{\theta_o}) + \frac{1}{t} \sum_i \log\left(\omega_i - J_{c_i}(\pi_{\theta_o})\right)$$
$$= L_{\pi_{\theta_o}}(\pi_{\theta_o}) - \frac{4\varepsilon^{\pi_{\theta_o}}\gamma}{(1-\gamma)^2} D_{KL}^{max}(\pi_{\theta_o}, \pi_{\theta_o}) + \frac{1}{t}\sum_i \log\left(\omega_i - J_{c_i}(\pi_{\theta_o})\right)$$
$$\leq L_{\pi_{\theta_o}}(\pi_\theta) - \frac{4\varepsilon^{\pi_{\theta_o}}\gamma}{(1-\gamma)^2} D_{KL}^{max}(\pi_{\theta_o}, \pi_\theta) + \frac{1}{t}\sum_i \log\left(\omega_i - J_{c_i}(\pi_\theta)\right)$$
$$\leq J_R(\pi_\theta) + \frac{1}{t}\sum_i \log\left(\omega_i - J_{c_i}(\pi_\theta)\right). \tag{12}$$

Theorem 1 is applied in Step (12).

$$J_R^{\pi_\theta} - J_R^{\pi_{\theta_o}}$$
$$\geq - \sum_i \frac{1}{t} \log \left(1 + \frac{J_{C_i}^{\pi_\theta} - J_{C_i}^{\pi_{\theta_o}}}{\omega_i - J_{C_i}^{\pi_{\theta_o}}} \right)$$
$$\geq - \sum_{i \in \Phi} \frac{1}{t} \log \left(1 + \frac{J_{C_i}^{\pi_\theta} - J_{C_i}^{\pi_{\theta_o}}}{\omega_i - J_{C_i}^{\pi_{\theta_o}}} \right) \tag{13}$$
$$\geq - \sum_{i \in \Phi} \frac{1}{t} \log \Bigg(1 - \underset{\substack{s \sim \rho_{\pi_{\theta_o}} \\ a \sim \pi_\theta}}{\mathbb{E}} \left[\frac{1}{\psi^{\pi_{\theta_o}}} A_{C_i}^{\pi_{\theta_o}}(s,a) \right]$$
$$+ \frac{\sqrt{2}\gamma\varepsilon_{C_i}^{\pi_\theta}}{\psi^{\pi_{\theta_o}}(1-\gamma)^2} \sqrt{D_{KL}^{\max}(\pi_\theta, \pi_{\theta_o})} \Bigg), \tag{14}$$

In Step (13), we neglect all the positive terms with $J_{C_i}(\pi_\theta) \geq J_{C_i}(\pi_{\theta_o})$. Step (14) is obtained by applying Theorem 2. $\square$

In the above theorem, we provide new results for the lower bound of our policy improvement. The worst case performance guarantee is decided by the advantage functions of the previous policy and the KL divergence between it and the new policy. A lower the KL divergence and higher advantage function values can lead to a better policy improvement.

C. Adaptive IPO

Besides the theoretical improvements, we propose our adaptive IPO algorithm, improved over the original IPO in [19]. As demonstrated in Theorem 3, there is a trade-off between approximation accuracy and algorithm performance depending on the hyperparameter t. In the original IPO, the hyperparameter t is fixed. It is time-consuming to tune the hyperparameter and is not adaptive to different system dynamics. In our work, we adjust the hyperparameter in an adaptive manner: we start with a small t to have more stable policy updates, and gradually increase t to achieve better policies on convergence.

In practice, we face the challenge of complex computation of TRPO in large-scale problems. The heuristic algorithm, PPO, achieves better performance than TRPO empirically, which is employed in IPO. In the implementation, we also replace the objective function of TRPO with that of PPO in Eq. (9), making the objective as maximizing

$$L^{IPO}(\theta) = L_R^{CLIP}(\theta) + \sum_{i=1}^m \phi(\widehat{J}_{C_i}^{\pi_\theta}), \tag{15}$$

where we denote $L_R^{CLIP}(\theta)$ and $L_{C_i}^{CLIP}(\theta)$ to be the formulations applying Eq. (9) to reward and cost functions.

We present our adaptive IPO in Algorithm 1. In Phase I, the cost functions are successively optimized to obtain a feasible policy. The algorithm is initialized with the objective of maximizing

$$L^C(\theta) = -L_{C_1}^{CLIP}(\theta),$$

to decrease the cumulative cost C_1 until the constraint is satisfied. Thereafter, at the end of iteration i, we update the objective $L^C(\theta)$ for next iteration,

$$L^C(\theta) = -L_{C_{i+1}}^{CLIP}(\theta) + \sum_{j=1}^i \phi(\widehat{J}_{C_j}^{\pi_\theta}),$$

where we replace $-L_{C_i}^{CLIP}(\theta)$ with its logarithmic barrier function $\phi(\widehat{J}_{C_i}^{\pi_\theta})$ to guarantee the constraint for C_i is satisfied in the future updates, and add $-L_{C_{i+1}}^{CLIP}(\theta)$ for the next cost function C_{i+1}. The above process is repeated until obtaining a feasible policy for all the constraints.

In Phase II, the algorithm is initialized with the feasible policy in Phase I. In light of the trade-off of t, we start with a moderate small t and adaptively increase it with a factor $\mu > 1$ when policy convergence criteria are satisfied. In each iteration, we update the policy by maximizing the objective $L^{IPO}(\theta)$ in Eq. (15).

Algorithm 1 The procedure of adaptive IPO

Input: Initialize with a random policy π_{θ_0}. Set the hyperparameter ε for PPO, $t = t_0$, $\mu > 1$ for logarithmic barrier function, and iteration number k = 0
Output: The policy parameter θ

Phase I:
1: Initialize $L^C(\theta) = -L^{CLIP}_{C_1}(\theta)$
2: **for** i=1,2,...,m **do**
3:    **while** $\widehat{J}^{\pi_\theta}_{C_i} > 0$ **do**
4:      Sample N trajectories $\tau_1, ..., \tau_N$ including observations, actions and costs with the current policy π_{θ_k}
5:      Calculate advantages, constraint values, etc
6:      Update θ_{k+1} by maximizing $L^C(\theta)$ with stochastic gradient descent methods
7:      Iteration k = k+1
8:    **end while**
9:    Update the objective as maximizing $L^C(\theta) = -L^{CLIP}_{C_{i+1}}(\theta) + \sum_{j=1}^{i} \phi(\widehat{J}^{\pi_\theta}_{C_j})$
10: **end for**

Phase II:
1: **for** iteration k **do**
2:    Sample N trajectories $\tau_1, ..., \tau_N$ including observations, actions, rewards and costs with the current policy π_{θ_k}
3:    Calculate advantages, constraint values, etc
4:    Update θ_{k+1} by maximizing $L^{IPO}(\theta)$ with stochastic gradient descent methods
5:    **if** the policy converges **then**
6:      $t = \mu * t$
7:    **end if**
8:    Iteration k = k+1
9: **end for**
10: **return** the policy parameter $\theta = \theta_{k+1}$

D. Instantaneous Constraints

We further extend the policy neural network with new layers to handle instantaneous constraints. Our policy network π_θ takes a state s as input and outputs an action $a = \pi_\theta(s)$. To satisfy instantaneous constraints, one way is to project the infeasible action generated by π_θ to the feasible space [20]. To achieve this, one can introduce another additional last layer to π_θ, whose role is to solve

$$\min_a \frac{1}{2} \|a - \pi_\theta(s)\|^2 \qquad (16)$$
$$s.t. \ C_j(s_t, a_t) \leq \epsilon_j$$

where $C_j(s_t, a_t)$ is the instantaneous cost under new action, and ϵ_j is the predefined constraint. In other words, we project the action from the policy $\pi_\theta(s)$ to the ℓ_2 nearest feasible action a that satisfies the instantaneous constraint. The projection idea can apply to both explicit and implicit constraints. One challenge for implicit constraints is that the function $C_j(\cdot, \cdot)$ is unknown. To address the problem, we take

advantage of another neural network to learn the value of $C_j(s_t, a_t)$ simultaneously, as in [20].

Global Sum constraints are a special case of explicit instantaneous constraints that can be easily satisfied with a softmax projection layer. Assume that the action is a vector with k entries, denoted as $a = [a_1, a_2, ..., a_k]$, and we deal with the specific explicit constraints that require $\sum_k a_k = 1$. A softmax layer is added right after the output layer of policy network π_θ to project the arbitrary action a to a feasible action $a' = [a'_1, a'_2, ..., a'_k]$, where

$$a'_i = \frac{e^{a_i}}{\sum_{j=1}^{k} e^{a_j}}. \qquad (17)$$

In the radio access scenario II-B, we handle our explicit constraints (bandwidth allocation) with the softmax projection layer and handle the implicit constraints (latency) with general ℓ_2 projection layer.

VI. EXPERIMENTS

In the experiments, we demonstrate that CLARA outperforms the baselines including traditional methods and RL-based methods; CLARA can handle multiple constraints simultaneously and CLARA is able to speed up converge by incorporating domain knowledge; at last, we discuss the reason why CLARA works well through our observations.

A. Settings

We evaluate the performance of CLARA on the radio resource slicing scenario, as described in Section II-B, and compare it with traditional benchmarks and RL-based methods. The network scenario is numerically simulated. As hidden structures and system dynamics exist, accurate models of the system cannot obtain in practice. However, we can apply traditional methods based on observed states, which result in the following baselines, as suggested in [24]:

- One-third equal allocation: the total bandwidth is equally sliced into three pieces.
- User-number-based allocation: the total bandwidth is sliced weighted by the number of users of each type.
- Packet-number-based allocation: the total bandwidth is sliced weighted by the number of packets of each type.
- Traffic-demand-based allocation: the total bandwidth is sliced weighted by the current traffic demand.

The RL-based methods include:

- IPO: the original IPO [19] with fixed hyperparameter t.
- PPO: the state-of-the-art unconstrained RL algorithm, which usually outperforms DQN [24] and TRPO [27].
- PPO+Safelayer: the PPO method with Safelayer.

We also compare with only adaptive IPO without Safelayer, to demonstrate the performance of Safelayer. In the experiment, all policy neural-networks consist of two fully connected layers, with 64 and 32 nodes, respectively.

(a) Reward under Video cumulative constraint

(b) Video cumulative constraint (log-scaled y-axis)

(c) Instantaneous constraint under Video cumulative constraint (log-scaled y-axis)

(d) Reward under VoLTE cumulative constraint

(e) VoLTE cumulative constraint (non-linear y-axis)

(f) Instantaneous constraint under VoLTE cumulative constraint (log-scaled y-axis)

(g) Reward under URLLC cumulative constraint

(h) URLLC cumulative constraint

(i) Instantaneous constraint under URLLC cumulative constraint (log-scaled y-axis)

(j) Reward under Video and VoLTE cumulative constraints

(k) Video cumulative constraint (log-scaled y-axis)

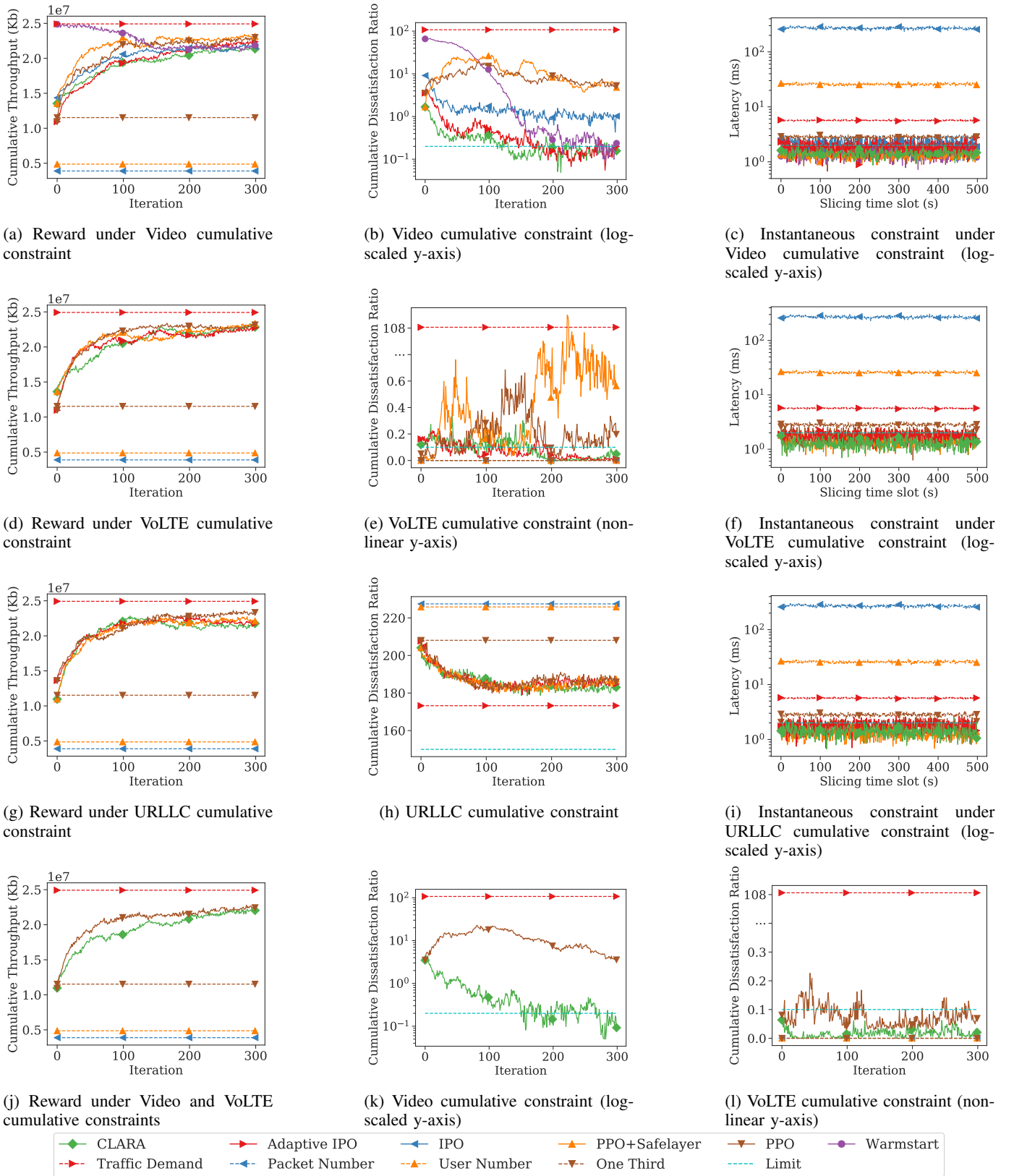(l) VoLTE cumulative constraint (non-linear y-axis)

Fig. 2: Average performance under different cumulative constraints: Fig. 2a, 2b and 2c are under single Video cumulative constraint; Fig. 2d, 2e and 2f are under single VoLTE cumulative constraint; Fig. 2g, 2h and 2i are under single URLLC cumulative constraint; Fig. 2j, 2k and 2l are under both Video and VoLTE cumulative constraints; the dash lines are limits for different constraints. In Fig. 2b, 2k, the cumulative values for Packet Number, User Number, and One Third are 0.

B. Evaluation Results

First, we demonstrate the evaluations with one single cumulative constraint selected from cumulative dissatisfaction ratio of Video, VoLTE and URLLC separately, in Fig. 2 (a-i). Fig. 2a, 2b, 2d, 2e, 2g, 2h show the results of long-term reward (throughput) and cumulative constraints (dissatisfaction ratio) with respect to the iterations of policy updates. Both the rewards and constraints are cumulative values in 500 time slots. For RL-based methods, the rewards and cumulative cost values update during the training process, while the traditional baselines do not improve or adapt to the changes in the environment. Even though PPO and PPO+Safelayer can achieve slightly higher reward than CLARA, its cost value significantly violate the constraint. The original IPO violate the constraint as well. CLARA achieves fair high reward and satisfies cumulative constraints.

We also collect the policy after training and demonstrate the performance on the implicit instantaneous constraints (latency) in 500 time slots, compared with baselines, shown in Fig. 2c, 2f, 2i. CLARA and PPO+Safelayer satisfy the latency requirements best of all. Adaptive IPO without the Safelayer are not guaranteed to satisfy the instantaneous constraint. Remark that in Fig. 2g and 2h, we have tested an extreme case, where the traffic demand of URLLC is larger than the total bandwidth. The traffic demand based allocation benefits from knowing the extra information of the total traffic demand volume, however, our algorithm still performs almost the same. Our method can satisfy the latency constraint while the traffic demand allocation cannot. Moreover, since explicit instantaneous constraints (bandwidth allocation) can be numerically checked. CLARA can always make sure that they are satisfied.

In the setting of multiple cumulative constraints, we consider the cumulative constraints on the dissatisfaction ratio of Video and VoLTE users together. The results in Fig. 2 (j-l) show that CLARA achieves the equivalent best reward as well as the satisfaction of the two cumulative constraints. All above, the final policy learned by CLARA outperforms all the baselines in either reward or constraint cost, if not both.

C. Convergence speed

In Fig. 2, we have seen that in most cases CLARA converges within 100 iterations. We can further speed up the convergence, by utilizing domain knowledge and traditional baselines to obtain an initial policy other than a purely random one to warm start. As shown by "Warmstart" in Fig. 2 (a-c), we use the traffic demand based allocation as our initial policy, which has a high reward although a high cost violating the constraint; after a few iterations, the reward converges to the reward achieved starting with a random policy and the constraint is satisfied. The convergence in RL is an essential and challenging issue. A lot of potential future work need to be explored to speed up the convergence, e.g., better exploration techniques and model-based approaches, etc.

TABLE III: Statistics of three types of users

	Video	VoLTE	URLLC
Number of user	50.15	49.44	1.92
Number of packet	5950.56	601.50	10.70
Traffic Demand (kb)	7002.35	187.97	295290.37

TABLE IV: Average bandwidth allocation with algorithms

(kb)	Video	VoLTE	URLLC
One third	34133.33	34133.33	34133.33
User number	50589.71	49871.62	1938.67
Packet number	92847.71	9385.29	167.01
Traffic Demand	2370.53	63.63	99965.83
IPO	25641.12	8933.74	67842.22
Adaptive IPO	22152.22	5320.23	74927.54
PPO	22141.71	11958.00	68300.28
PPO+Safelayer	19563.07	7711.20	75125.81
CLARA	27104.34	7372.04	67923.62

D. Discussions

We show the statistics of three types of users to illustrate the diverse characteristics of the three network slices, in Table III as suggested by [24]. We also show the average resource allocation with all methods in Table IV. Traffic demands from URLLC network slice users dominate over 90% of the total traffic demands, while the number of users and packets of URLLC is far less. Given such heterogeneous user demands, the one third equally allocation method, user number and packet number based method allocate much more than enough bandwidth to Video and VoLTE users, resulting in losing the high volume traffic from the URLLC users. As for the traffic demand based method, it focuses on the demands of URLLC network slices and works well in reward maximization with the extra information of the real user demands. However, this allocation results in the dissatisfaction and leaving of the users of the other two network slices. In the extreme case as in Fig. 2h, the resource is not enough to satisfy the URLLC users, and therefore, the URLLC users are also unsatisfied and leave. In this case, the allocation with traffic demand based method loses all the users, while our CLARA can adapt to this situation, and allocate more to the users from Video and VoLTE network slices with low latency requirement, to keep a high user participation level, achieving a better overall performance. The other RL-based methods can achieve a good reward performance, but violate the cumulative or instantaneous constraints.

VII. RELATED WORK

A. Network Slicing

Rost et al. in [31] propose that network slicing, which enables the multiplexing of virtualized and independent logical networks on the same physical network infrastructure, is an efficient solution that addresses the diverse requirements of 5G mobile networks, through analyzing the realization options of a flexible radio access network slicing. Different service models and architectures are suggested, such as those in [32]. The key of network slicing is efficient resource allocation. The dynamic resource allocation in network slicing can improve

the resource efficiency [33]. However, resource allocation in network slicing is proved to be NP-hard [10], [34] and heuristic algorithms are proposed, while the performances are not guaranteed.

B. Reinforcement Learning in Network Slicing

RL is employed to allocate resources in network slicing. In [24], the authors study the setting of demand-aware network slicing. Network bandwidth is allocated to three types of slices: Video, VoLTE, and URLLC; the state is the traffic load in each slice; the action is the bandwidth allocation; the reward is the weighted sum of spectrum efficiency (SE) and QoE of the slices. Bega et al. [35] study admission control of two types of network slice requests with different price and service requirements. The state is the number of current network slice users in the system; the action is the admission decision of the provider; and the reward is the revenue of the service provider. In [36], the system allocates the RAN and mobile-edge computing (MEC) slices to provide computation offloading services to mobile users; the state consists of task queue, energy queue, and channel qualities; the action is the decision to execute the offloading computation; the reward is the total utility value of all users. Similar works can be found in [37], [38]. These works perform efficiently, however, they do not consider the crucial constraints in network slicing usually require knowing the traffic demand or user mobility which is not realistic in most cases. A very recent work [39] considers constraints in network slicing. However, the Lagrangian Relaxation method they are using to handle the constraints is not as efficient as our IPO [19] where CLARA relies on. An earlier short version of our work appears at a workshop [40] and another short version appeared as a poster [41]. In comparison, in this paper, we expand the system architecture and description, we prove the policy improvement in Theorem 4, we show CLARA can handle multiple cumulative constraints that are more common in real-world scenarios, we demonstrate the feasibility of warm-start to speed up convergence, and we expand the evaluation to compare with more baselines.

C. Constrained Reinforcement Learning

Although much progress has been made in RL, while the work on constrained RL is limited [42], [43]. The most common approach is to use Lagrangian relaxation [44], [45]. It reduces the constrained optimization problem to an unconstrained one by adding constraint as a weighted penalty to the objective function, however, Lagrangian relaxation methods are sensitive to the initialization of the Lagrange multipliers and the learning rate and constraint satisfaction is only guaranteed upon convergence. Constrained Policy Optimization (CPO) [28] employs TRPO to approximate the complex constrained optimization problem with a quadratic optimization. However, CPO can result in the high computation training cost in large scale problems and Interior-point Policy Optimization (IPO) [19] takes advantage of the logarithmic barrier function

to handle cumulative constraints and achieves good performance, while IPO does not have theoretical guarantee for the policy improvement and the hyperparameters are not adaptive tuned.

As for instantaneous constraints, a natural approach is to project the solutions to the feasible space. In [20], [21], the authors propose approaches based on Deep Deterministic Policy Gradient (DDPG) and project the output of the infeasible actors neural network to the feasible space by adding a projection safety layer.,

VIII. Conclusion

This work focuses on network slicing with constraints. Because of the service diversity, complexity, and hidden structures, prior approaches fail to satisfy the cumulative and instantaneous service constraints. To address this challenge, we formulate the network slicing problem as a Constrained Markov Decision Process (CMDP) and solve it with CLARA, which is a reinforcement learning based resource allocation framework for network slicing with constraints. We evaluate CLARA on the radio resource allocation scenario to illustrate the details of the proposed approach and its advantage. Our evaluation results show that constrained reinforcement learning can solve network slicing problems effectively. Much future work exists, including stronger theoretical bounds, improved sample efficiency, testbed development, as well as more dynamic real-world evaluations.

References

[1] L. Lu and Y. Liu, "Safeguard: User reauthentication on smartphones via behavioral biometrics," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 3, pp. 53–64, 2015.

[2] H. Qu, X. Xie, Y. Liu, M. Zhang, and L. Lu, "Improved perception-based spiking neuron learning rule for real-time user authentication," *Neurocomputing*, vol. 151, pp. 310–318, 2015.

[3] L. Lu, L. Liu, M. J. Hussain, and Y. Liu, "I sense you by breath: Speaker recognition via breath biometrics," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 2, pp. 306–319, 2017.

[4] Y. Liu, J. Chen, and H. Chen, "Less is more: Culling the training set to improve robustness of deep neural networks," in *International Conference on Decision and Game Theory for Security*. Springer, 2018, pp. 102–114.

[5] 3rd Generation Partnership Projec (3GPP), "Cellular system support for ultra low complexity and low throughput Internet of Things," *3GPP Technical Report (TR) 45.820*, 2015.

[6] X. Li, D. Li, J. Wan, A. V. Vasilakos, C.-F. Lai, and S. Wang, "A review of industrial wireless networks in the context of industry 4.0," *Wireless networks*, vol. 23, no. 1, pp. 23–41, 2017.

[7] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[8] N. Alliance, "5G white paper," *Next generation mobile networks, white paper*, vol. 1, 2015.

[9] N. Nikaein, E. Schiller, R. Favraud, K. Katsalis, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun, and T. Korakis, "Network store: Exploring slicing in future 5G networks," in *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*. ACM, 2015, pp. 8–13.

[10] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. Leung, "Network slicing based 5G and future mobile networks: mobility, resource management, and challenges," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[11] "ITU workshop on Machine Learning for 5G and beyond," 2018.

[12] 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*. ACM, 2017, pp. 197–210.

[13] T. Uzakgider, C. Cetinkaya, and M. Sayit, "Learning-based approach for layered adaptive video streaming over sdn," *Computer Networks*, vol. 92, pp. 357–368, 2015.

[14] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 50–56.

[15] J. Chuai, Z. Chen, G. Liu, X. Guo, X. Wang, X. Liu, C. Zhu, and F. Shen, "A collaborative learning based approach for parameter configuration of cellular networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1396–1404.

[16] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *IEEE International Conferences on Big Data (IEEE BigData 2016)*, Dec 2016.

[17] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung, J. Tucker, and A. Swami, "Macs: Deep reinforcement learning based sdn controller synchronization policy design," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–11.

[18] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De, "Hotdash: Hotspot aware adaptive video streaming using deep reinforcement learning," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 165–175.

[19] Y. Liu, J. Ding, and X. Liu, "Ipo: Interior-point policy optimization under constraints," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4940–4947.

[20] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," *arXiv preprint arXiv:1801.08757*, 2018.

[21] A. Bhatia, P. Varakantham, and A. Kumar, "Resource constrained deep reinforcement learning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, no. 1, 2019, pp. 610–620.

[22] N. ETSI, "Network functions virtualization (nfv) infrastructure overview," *NFV-INF*, vol. 1, p. V1, 2015.

[23] NGMN, "Ngmn radio access performance evaluation methodology," 2007. [Online]. Available: https://www.ngmn.org/publications/ngmn-radio-access-performance-evaluation-methodology.html

[24] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[25] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[26] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[27] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[28] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 22–31.

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

[30] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[31] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[32] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[33] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How should I slice my network? a multi-service empirical evaluation of resource sharing efficiency," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 191–206.

[34] S. D'Oro, F. Restuccia, A. Talamonti, and T. Melodia, "The slice is served: Enforcing radio access network slicing in virtualized 5G systems," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 442–450.

[35] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, and X. Costa-Perez, "Optimising 5G infrastructure markets: The business of network slicing," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[36] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, 2018.

[37] V. Sciancalepore, X. Costa-Perez, and A. Banchs, "Rl-nsb: Reinforcement learning-based 5g network slice broker," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1543–1557, 2019.

[38] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "Gan-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 334–349, 2019.

[39] Y. Xu, Z. Zhao, P. Cheng, Z. Chen, M. Ding, B. Vucetic, and Y. Li, "Constrained reinforcement learning for resource allocation in network slicing," *IEEE Communications Letters*, vol. 25, no. 5, pp. 1554–1558, 2021.

[40] Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning based approach for network slicing," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–6.

[41] ——, "Resource allocation method for network slicing using constrained reinforcement learning," in *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, 2021, pp. 1–3.

[42] Y. Liu, A. Halev, and X. Liu, "Policy learning with constraints in model-free reinforcement learning: A survey," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021.

[43] Y. Liu and X. Liu, "Cts2: Time series smoothing with constrained reinforcement learning," in *Proceedings of the thirteenth Asian Conference on Machine Learning*, 2021.

[44] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, 2017.

[45] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *International Conference on Learning Representations*, 2019.