A Constrained Reinforcement Learning Based Approach for Network Slicing

Yongshuai Liu*, Jiaxin Ding†, Xin Liu*
*Computer Science Department, University of California, Davis
{yshliu, xinliu}@ucdavis.edu
†John Hopcroft Center for Computer Science, Shanghai Jiao Tong University
jiaxinding@sjtu.edu.cn

***Abstract*—With the proliferation of mobile networks, we face strong diversification of services, demanding the current network to embed more flexibility. To satisfy this daring need, network slicing is embraced as a promising solution for resource utilization, in 5G and future networks. In network slicing, dynamic resource orchestration and network slice management are critical for resource efficiency. However, it is highly complicated such that the traditional approaches can not effectively perform resource orchestration due to the lack of accurate models and hidden problem structures. To address this challenge, we propose a constrained reinforcement learning based approach for network slicing. We formulate the resource allocation problem as a Constrained Markov Decision Process (CMDP) and solve it using constrained reinforcement learning algorithms. Specifically, we use the adaptive interior-point policy optimization and policy safety layer methods to deal with cumulative and instantaneous constraints. Our evaluations show that our method is effective in resource allocation with service demand guarantees and significantly outperforms baselines.**

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

I. Introduction

With the proliferation of mobile networks, we face strong diversification of services, such as autonomous driving, industry 4.0, virtual/augmented reality, Internet of Things (IoT), etc. These services are characterized by heterogeneous performance, functional, and operational requirements, and demand the network to embed more flexibility. In 5G and future networks, network slicing, enabled by network function virtualization (NFV) and software defined networking (SDN), is embraced as a promising solution for flexible resource provisioning - it creates multiple virtual network instances, named network slices, with service isolation and guarantees on top of a common physical infrastructure. Network slicing can be performed at different parts of the network, including core cloud, edge cloud, radio resource management, RAN processing, spectrum, and radio frontend to manage and allocate different resources to satisfy the different service demands.

Network slicing is a generalized resource allocation problem over heterogeneous resources to meet the service demands, in compliance with the complex network dynamics, in the long run. Such dynamic orchestration of network slices is essential for resource efficiency. However, the resource allocation in network slicing is a highly complicated problem, which the existing traditional approaches can not solve effectively and efficiently. First, traditional optimization approaches require accurate mathematical models with parameters known, which is often difficult to achieve in practice, especially with the increasing complexity, scale and service diversity of the 5G and future networks. Constraints from the physical systems and service demands are prevalent and complex, such as latency requirement, service level agreement and safety demand [1]–[3], which further adds to the difficulty, let alone obtaining a closed-form expression. Second, traditional methods do not adapt to epistemic uncertainty, exhibited as hidden structures in networks, due to a lack of knowledge and subsequent ability to explore and learn from the studied system.

Faced with these challenges, learning-based approaches are beneficial because they explore and learn from the environment without assuming the knowledge of accurate models. Recently, there have been growing learning-based network research works showing significant performance improvement, e.g., [4], [5], However, few previous works have analyzed the resource allocation problem with the constraints imposed by the service requirements, which is the crucial for network slicing.

In this work, we propose a constrained reinforcement learning based approach 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). We develop efficient reinforcement learning algorithms for network slicing under both cumulative and instantaneous constraints. To the best of our knowledge, we are the first one to apply constrained RL for network slicing. Specifically, to deal with cumulative constraints, we propose our adaptive constrained reinforcement learning algorithm based on Interior-point Policy Optimization (IPO) [6]. 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 [7], [8].

Y. Liu and X. Liu would like to acknowledge supports from NSF CNS-1718901, IIS-1838207, and CNS 1901218. J. Ding would like to acknowledge supports from Shanghai Sailing Program 20YF1421300.

Jiaxin Ding is the corresponding author.

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

II. Problem Formulation

In this section, we present the problem formulation for resource allocation of network slicing as Constrained Markov Decision Process.

A. Constrained Markov Decision Process

The Constrained Markov Decision Process (CMDP) is defined with the tuple $(S, A, P, R, C, \mu, \gamma)$. S is the set of states, including system information such as 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. A is the set of actions of the operator, which, depending on what level our method runs, could include the admission control, spectrum/power allocation, computation and storage allocation, priority assignment, network routing decision, network configurations, etc. The action space is constrained by the instantaneous constraints, e.g., resource limits, fairness, isolation and other network constraints. The instantaneous constraints can be further be catergorized into two types, explicit and implicit instantaneous constraints. The explicit instantaneous constraints can be accurately (and relatively easily) evaluated for each action, for example, spectrum band available, the number of antennas, and transmission time, etc. The implicit instantaneous constraints are the outcome of actions that we do not have an accurate closed-form formulation. Examples include latency (e.g., average latency or tail latency), job competition time, interference, etc. Such constraints have to be modeled or learned using existing data and/or during exploration. We define a is feasible, if $a \in A$ satisfies all the constraints including both explicit and implicit. Let $P : S \times A \times S \mapsto [0, 1]$ be the transition probability function, where $P(s'|s, a)$ is the transition probability from state s to state s' taking action a, $R : S \times A \times S \mapsto \mathbb{R}$ is the reward, which can be a weighted sum of the objectives, including the overhead of network slice reconfiguration as a negative reward. There are m cost functions for constraints; each function $C_i : S \times A \times S \mapsto \mathbb{R}$ cumulatively is under a constraint, such as service level agreement (SLA), outage probability and average data rate, etc. $\mu : S \mapsto [0, 1]$ is the initial state distribution. γ is the discount factor, which can be different for reward and constraints.

We write a policy π as π_θ to emphasize its dependence on the parameter θ (e.g., a neural network policy with parameter θ). Our objective is to select a policy π_θ, which maximizes the discounted cumulative reward $J_R^{\pi_\theta}$ while satisfying discounted cumulative constraints $J_{C_i}^{\pi_\theta}$ and instantaneous constraints, defined as

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

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

where $\tau = (s_0, a_0, s_1, a_1...)$ is a trajectory, and $\tau \sim \pi_\theta$.

Formally, the optimization problem is defined as

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

$$\text{subject to} \quad \text{each } a_t \text{ is feasible}, \tag{4}$$

$$J_{C_i}^{\pi_\theta} \leq \omega_i, \text{for each } C_i, \tag{5}$$

B. Case Study: Radio Access Network Slicing

The formulation can be applied to general network slicing problems, including radio access, edge, cloud, and core networks. In this work, we use a radio resource slicing scenario derived from the work [9] as a case study.

Consider a scenario where there is one single Base station (BS) providing three types of services (i.e., Video, VoLTE, URLLC). Each type of user arrives following a Poisson distribution. User requests are generated by the distribution described in Table I based on their respective streaming model same as the setting in [9], similar settings also seen in [4]. The total bandwidth of the BS is fixed and given (100 Mbps). The network slicing problem is for the system operator to allocate bandwidth to each type of users (a slice). Users in the same slice are assigned bandwidth equally.

The system is time slotted. At the beginning of each time slot, the BS decides the bandwidth allocation b_i (i is user type: Video, VoLTE and URLLC) for the three network slices based on the number of active users in each slice. Let t_i be the actual traffic demand, then the throughput for each type of user is $\min(b_i, t_i)$.

For each type of users, there is a dissatisfaction ratio representing their dissatisfaction with respect to the service received. In simulation, we define it as $1 - \frac{\min(b_i,t_i)}{t_i}$. If the bandwidth assigned to the users is greater or equal to their demand then the dissatisfaction ratio is 0, otherwise, it's $1 - \frac{b_i}{t_i}$. The latency l_i for each type of user, which does not yield to easy mathematical formulation, is decided based on a queue maintained at the BS. We assume the first-come-first-serve service principle. Traffic that cannot be serviced in the current time slot is queued to the next time slot. Each type of users arrive and depart the network following a Poisson distribution with mean λ_i and μ_i. The arrival rate λ_i adapts based on the satisfaction ratio of last time slot. In the simulation, we assume the λ_i is updated in each time slot with $\lambda_i = 0.99 * \lambda_i + 0.01 * \lambda_i * \frac{b_i}{t_i}$. Furthermore, users may depart with the probability of the dissatisfaction ratio. We assume these user behaviors are unknown to the slicing algorithms. This is one of the reasons why learning-based approaches, which incorporate exploration, perform better than the traditional methods based only on observed states (i.e., the number of active users in each type).

C. Mapping Network Slicing to CMDP

Next, we address how to map the radio access network slicing problem to CMDP. The state is the number of users in each type $n = (n_{Video}, n_{VoLTE}, n_{URLLC})$, observed at the beginning of each time slot. We do not know the exact overall traffic demand generated in this time slot. The action

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

is $b = (b_{Video}, b_{VoLTE}, b_{URLLC})$, bandwidth allocation for each type of users. The reward function at a time slot $R(n, b)$ is defined as the total throughput $\min(b_{Video}, t_{Video}) + \min(b_{VoLTE}, t_{VoLTE}) + \min(b_{URLLC}, t_{URLLC})$. The objective is to maximize the long-term expected reward. The actions need to satisfy both the explicit and implicit instantaneous constraints. The explicit instantaneous constraint is the sum of sliced bandwidth, is bounded by the total bandwidth (100 Mbps). The implicit instantaneous constraint is that of the average latency of each type of users. Last, the cumulative constraints of users is the expected cumulative dissatisfaction ratio of users.

III. Preliminaries

In this section, we briefly review the preliminaries from previous works to solve our problem.

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

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

$V_{C_i}^{\pi_\theta}(s)$, $Q_{C_i}^{\pi_\theta}(s, a)$, $A_{C_i}^{\pi_\theta}(s, a)$, 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 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. Trusted Region Methods

Trust Region Policy Optimization (TRPO) [10] 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

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

The objective of TRPO is to maximize

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

where π_{θ_o} is the old policy to improve, $\varepsilon^{\pi_{\theta_o}} = \max_{s,a}|A^{\pi_{\theta_o}}(s, a)|$.

Proximal Policy Optimization (PPO) [11] 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

$$\begin{aligned} &L^{CLIP}(\theta) \\ &= \mathop{\mathbb{E}}_{\substack{s\sim\rho_{\pi_{\theta_o}} \\ a\sim\pi_{\theta_o}}}[\min\{r(\theta), \text{clip}(r(\theta), 1-\varepsilon, 1+\varepsilon)\}A^{\pi_{\theta_o}}(s, a)], \end{aligned} \tag{9}$$

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

IV. Constrained Reinforcement Learning

A. Cumulative Constraints

To deal with the cumulative constraints in the CMDP, we propose our method based on Interior-point Policy Optimization (IPO) [6]. IPO augments the objective function of PPO (Eq. (9)) with logarithmic barrier functions. However, IPO is conducted with fixed hyperparameter for the logrithmic barrier function, while we propose algorithms in an adaptive manner.

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 better approximation for the indicator function with a higher t.

Now we take $L^{CLIP}(\theta)$ in Eq. (9) as our objective with cumulative constraints, that is,

$$\begin{aligned} &\max_\theta \ L^{CLIP}(\theta), \\ &s.t. \quad \widehat{J}_{C_i}^{\pi_\theta} \le 0. \end{aligned} \tag{10}$$

We assume that the above optimization problem is strictly feasible and reduce it to an unconstrained optimization by augmenting the objective with the logarithmic barrier functions for constraints. Our objective becomes

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

B. Policy Performance Bound

Theorem 1: [6] 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.

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 [6]. In practice, a larger t can result a large fluctuation near the boundary. Hence, there is a tradeoff in choosing this hyperparameter t.

C. Practical Implementation

In this work, we improve IPO in an adaptive manner, to change the hyperparameter t adaptively in the tradeoff of approximation accuracy and algorithm performance, compared with the fix t in original IPO. We present our adaptive IPO in Algorithm 1. In phase I, the cost functions are recurrently optimized to find a feasible policy. The algorithm is initialized with the objective of maximizing

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

to decrease the cumulative cost on C_1 until the constraint is satisfied. Thereafter, we update the objective $L^C(\theta)$ for the next cost function, by replacing $-L^{CLIP}_{C_i}(\theta)$ with $\phi(\widehat{J}^{\pi_\theta}_{C_i})$ and adding $-L^{CLIP}_{C_{i+1}}(\theta)$,

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

The above process is repeated until we find a feasible policy for all the constraints.

In Phase II, the algorithm is initialized with a feasible policy in Phase I. As we discussed in Section IV-B, a larger t results in better accuracy of the policy estimation and meanwhile greater fluctuation near the boundary of the feasible region. In light of this, we start with a moderate small t and adaptively increase it with a factor $\mu > 1$. In each iteration, we update policy parameters by maximizing the objective of IPO $L^{IPO}(\theta)$. The adaptive IPO acts as the main component of the policy training in our method.

D. Instantaneous Constraints

We further extend the policy neural network obtained above with new layers to deal with instantaneous constraints.

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:

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

Phase II:

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

a) Explicit instantaneous constraints.: Our policy π_θ takes a state $s \in S$ as input and outputs a action $a = \pi_\theta(s) \in A$. 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 instantaneous constraint that requires $\sum_k a_k = 1$. Then 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 $a' = [a'_1, a'_2, ..., a'_k]$, such that

$$a'_i = \frac{e^{a_i}}{\sum_{j=1}^{k} e^{a_j}}. \tag{12}$$

b) Implicit instantaneous constraints.: To satisfy implicit instantaneous constraints, one way is to project the action generated by policy network π_θ to the feasible space [7]. To achieve this goal, one can introduce another additional last layer to π_θ, whose role is to solve

$$\begin{aligned} &\min_{a} \frac{1}{2} \left\| a - \pi_\theta(s) \right\|^2 \\ &s.t.\ C_i(s, a) \le \omega_i \end{aligned} \tag{13}$$

where $C_i(s, a)$ is the implicit instantaneous constraint value under action a given state s, and ω_i is the constraint limit. In other words, we project the action of the policy, $\pi_\theta(s)$, to the ℓ_2 nearest feasible action a that satisfies the implicit instantaneous constraint. One challenge is that the function $C_i(\cdot, \cdot)$ is unknown. To address the problem, we take advantage of another neural network to learn the value of $C_i(s, a)$ simultaneously, as is in [7].

V. Experiments

In the experiments, we demonstrate that our method outperforms the baselines including traditional methods and RL-based methods, with higher long term reward, as well as satisfying all the constraints; at last, we discuss the reason why our method works well through our observations.

A. Settings

As details described in Section II, we allocate the radio bandwidth resource of a base station to three types of users: Video, VoLTE and URLLC. The input states of our method is the number of users of each type observed at the beginning of each time slot; the action is the bandwidth allocation to the three types. We compare the performance of our method to four traditional benchmarks and an unconstrained RL-based method for the three types of users , as is [9]. The traditional baselines include:

- One-third equal allocation: the total bandwidth is equally sliced into three.
- 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 real traffic demand.

We also choose the most commonly applied RL algorithm, PPO [11] as a baseline. In the experiment, all policy neural networks consist of two fully connected layers, with 64 and 32 nodes, respectively.

B. Evaluation Results

First, we demonstrate the evaluations of resource allocation among three network slices, with one single cumulative constraint which is selected from cumulative dissatisfaction ratio of Video, VoLTE and URLLC separately, in Fig. 1 (a-h). Fig. 1a, 1b, 1d, 1e, 1g, 1h 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 our method and PPO, the rewards and cumulative cost values are updated during the training process, while the traditional baselines does not adapt to the changes in the environment. Even though PPO can get a little higher reward, its cost significantly violates the constraints.

We collect the policy after training and demonstrate the performance on the implicit instantaneous constraints (latency) in 500 time slots, shown in Fig. 1c, 1f, where the figure for the URLLC instantaneous constraint is omitted for the same pattern. Our algorithm satisfies the latency requirements best. Remark that in Fig. 1g and 1h, 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. Moreover, our method can satisfy the latency constraint while the traffic demand allocation can not. All above, the final policy learned by our method outperforms all the baselines in either reward or constraint cost, if not both.

	Video	VoLTE	URLLC
Number of user	37.38	47.71	1.93
Number of packet	4436.37	580.57	10.70
Traffic Demand (kb)	5220.75	181.43	295171.64

TABLE II: Statistics of three types of users

(kb)	Video	VoLTE	URLLC
One third	34133.33	34133.33	34133.33
User number	43989.44	56137.48	2273.07
Packet number	90357.18	11824.80	218.01
Traffic Demand	1778.61	61.81	100559.57
PPO	17459.34	6417.78	78522.87
Our method	23641.98	9453.28	69304.72

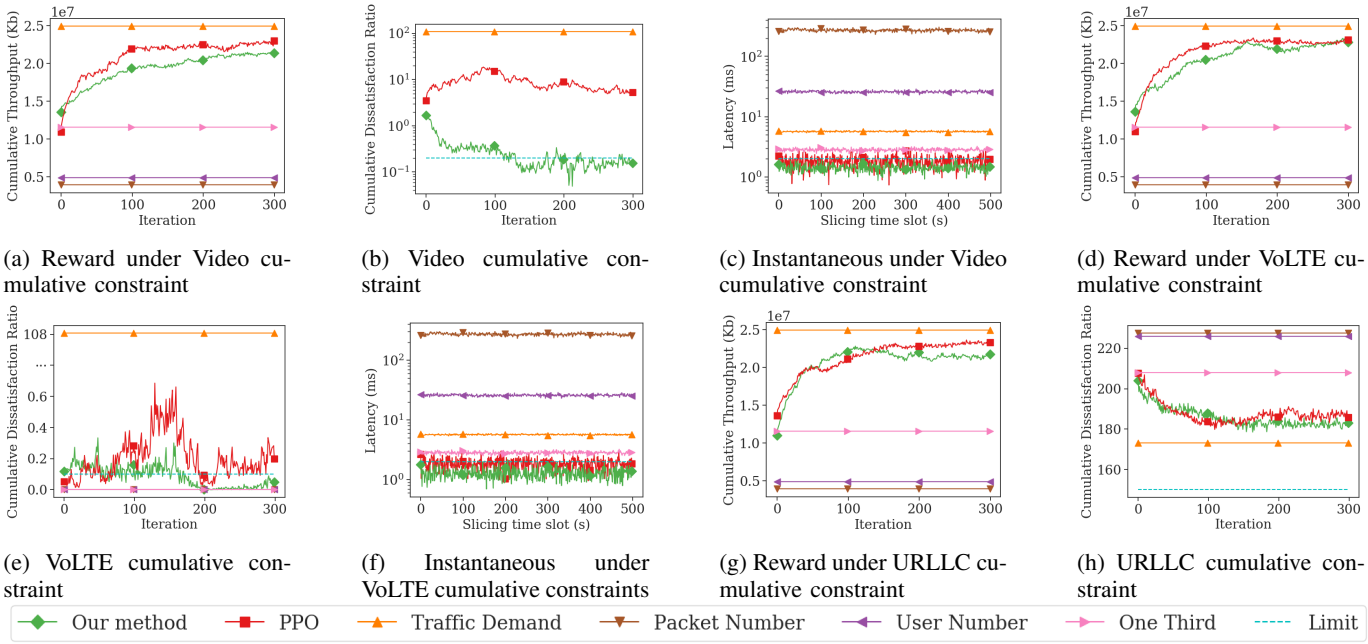
TABLE III: Average bandwidth allocation with our algorithm and traditional baselines

C. Discussions

We show the statistics of three types of users to illustrate the diversified characteristics of the three network slices, in Table II, and the average resource allocation with all methods in Table III. Traffic demands from URLLC users contribute over 90% of the total traffic demands, while the numbers of users and packets are far less. Given such heterogeneous demands, the one third equal 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 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. 1h, 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 our method 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 performance. PPO works in an unconstrained manner, therefore, it can achieve good rewards with constraints violated.

VI. Related work

Reinforcement Learning is employed to allocate resources in network slicing. In [9], the authors study the setting of



(a) Reward under Video cumulative constraint

(b) Video cumulative constraint

(c) Instantaneous under Video cumulative constraint

(d) Reward under VoLTE cumulative constraint

(e) VoLTE cumulative constraint

(f) Instantaneous under VoLTE cumulative constraints

(g) Reward under URLLC cumulative constraint

(h) URLLC cumulative constraint

Fig. 1: Average performance under different single cumulative constraints: Fig. 1a, 1b and 1c are under Video cumulative constraint; Fig. 1d, 1e and 1f are under VoLTE cumulative constraint; Fig. 1g, 1h and are under URLLC cumulative constraint

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 and QoE of the slices. Bega et al. [12] 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 [13], 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. To the best of our knowledge, there is no effort on constrained RL to solve network slicing problems with constraints.

VII. Conclusion

This work focuses on a constrained reinforcement learning based approach for network slicing. We formulate the network slicing problem as a Constrained Markov Decision Process (CMDP) and solve it with constrained reinforcement learning. Our evaluation results show that our method can solve network slicing problems effectively. Much future work exists, including stronger theoretical bounds, improved sample efficiency and testbed, as well as real world evaluations.

References

[1] L. Zanzi and V. Sciancalepore, "On guaranteeing end-to-end network slice latency constraints in 5g networks," in *2018 15th International Symposium on Wireless Communication Systems*. IEEE, 2018, pp. 1–6.

[2] 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*, 2018.

[3] 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.

[4] 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.

[5] 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.

[6] Y. Liu, J. Ding, and X. Liu, "IPO: Interior-point policy optimization under constraints," *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[7] 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.

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

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.