Generalizable Resource Scaling of 5G Slices using Constrained Reinforcement Learning

Muhammad Sulaiman*, Mahdieh Ahmadi*, Mohammad A. Salahuddin*, Raouf Boutaba*, and Aladdin Saleh[†]

*David R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada

{m4sulaim, mahdieh.ahmadi, mohammad.salahuddin, rboutaba}@uwaterloo.ca

[†]Rogers Communications Inc., Ontario, Canada

{aladdin.saleh@rci.rogers.com}

Abstract—**Network slicing is a key enabler for 5G to support various applications. Slices requested by service providers (SPs) have heterogeneous quality of service (QoS) requirements, such as latency, throughput, and jitter. It is imperative that the 5G infrastructure provider (InP) allocates the right amount of resources depending on the slice's traffic, such that the specified QoS levels are maintained during the slice's lifetime while maximizing resource efficiency. However, there is a non-trivial relationship between the QoS and resource allocation. In this paper, this relationship is learned using a regression-based model. We also leverage a risk-constrained reinforcement learning agent that is trained offline using this model and domain randomization for dynamically scaling slice resources while maintaining the desired QoS level. Our novel approach reduces the effects of network modeling errors since it is model-free and does not require QoS metrics to be mathematically formulated in terms of traffic. In addition, it provides robustness against uncertain network conditions, generalizes to different real-world traffic patterns, and caters to various QoS metrics. The results show that the state-of-the-art approaches can lead to QoS degradation as high as 44.5% when tested on previously unseen traffic. On the other hand, our approach maintains the QoS degradation below a preset 10% threshold on such traffic, while minimizing the allocated resources. Additionally, we demonstrate that the proposed approach is robust against varying network conditions and inaccurate traffic predictions.**

Index Terms—**5G, Network Slicing, Resource Scaling, Constrained Reinforcement Learning, QoS**

I. Introduction

With 5G, mobile networks are moving away from one-size-fits-all towards a more programmable network architecture. The adoption of Software Defined Networking (SDN) and Network Function Virtualization (NFV) allows an infrastructure provider (InP) to virtualize its physical network resources, and use them to create virtual isolated networks on top of a shared physical network infrastructure. These on-demand virtual isolated networks are also referred to as network slices. Network slicing enables 5G mobile networks to host applications or services with diverse quality of service (QoS) requirements. For example, enhanced mobile broadband (eMBB) slices can be used for applications that require high throughput but lenient latency constraints such as 4K video streaming. On the other hand, ultra-reliable low-latency communication (URLLC) slices can be used for applications that require high reliability and very low latency such as remote surgery. Whenever a service provider (SP) requests a slice from

an infrastructure provider (InP), it includes its peak traffic and its required minimum quality of service (QoS) in the service level agreement (SLA). The required resources for maintaining a slice's QoS depend on the slice type and its traffic, which varies with time. The InP can guarantee the QoS by allocating isolated resources to the slice based on its peak traffic. However, this can lead to over-provisioning since the actual traffic of a slice rarely reaches its peak [1]. In this case, the majority of the allocated resources remain unused or under-utilized. On the other hand, the InP can improve its resource efficiency (RE) by predicting the future traffic of a slice, and preemptively scaling its resources accordingly. However, under-provisioning the resources, based on inaccurate traffic prediction or imprecise modeling of the relationship between allocated resources and QoS, can lead to a deterioration in the QoS of the slice. As a result, a certain level of QoS degradation is typically incorporated into SLAs, and the goal of InP is to dynamically scale resources to maximize resource efficiency while keeping QoS degradation under the specified limit. We refer to this as dynamic resource scaling.

Several challenges need to be addressed to achieve effective dynamic resource scaling. Mobile networks consist of multiple domains including the radio access network (RAN), transport and the core network, and the QoS obtained can be dependent on the relative level of resources allocated in these domains. The QoS achieved at any resource allocation also depends on the state of the network at that time, *e.g.*, the level of interference by other slices or the state of the queues in the network. Additionally, the QoS may be defined heterogeneously for different slices serving different kinds of applications. For example, the QoS may be defined in terms of the throughput for an eMBB slice, and in terms of the latency for a URLLC slice. Finally, even though some proposed solutions in the literature require a dataset to be trained (e.g., [2, 3]), the traffic that a slice experiences may be unknown during training time. Therefore, given the uncertainty of network conditions and future traffic, and the complex modeling of the end-to-end network, it is challenging to design an algorithm that can dynamically scale the resources of the slices while keeping their QoS degradation under the agreed-upon threshold.

Several works in the literature model the mobile network mathematically [4, 5]. Since it is quite challenging to accurately model end-to-end network dynamics [6], these works are

based on simplifying assumptions which makes them inapplicable to real-world networks. Another method commonly used in the literature is to model the network as a queue and then use simulation to calculate the metrics of interest [2, 3]. This method can be too simple to model a multi-domain mobile network, and complex user-level QoS metrics [6, 7]. Given a network model, for resource scaling under QoS degradation constraints, a number of works in the literature propose using traditional deep reinforcement learning (DRL) algorithms such as Deep Q-Learning [8, 3]. However, the reward function of these methods needs to be carefully engineered to achieve the desired tradeoff between QoS degradation level and resource efficiency, which is non-trivial. To circumvent these problems, authors in [9] use constrained deep reinforcement learning (CDRL) and online learning to dynamically scale the resources. But online learning can be quite slow, and even infeasible for slices with short lifespans.

In this paper, we propose using a regression-based model to capture the behavior of an end-to-end network under different conditions. This model is trained offline using a dataset gathered by measuring the performance of an isolated slice in the real network under diverse network conditions and different amounts of allocated resources. Regression-based models, such as neural networks, can learn complex relationships between predictor and response variables, without requiring an exact network model. For dynamically scaling the resources allocated to the slice while satisfying QoS requirements, we propose using CDRL with offline training. Although offline training addresses the slow training problem of online training, it must be generalizable to online traffic patterns not seen during offline training. For this purpose, we utilize a risk-constrained DRL algorithm coupled with domain randomization (DR). Risk-constrained DRL increases the chances of meeting QoS degradation constraints under unpredictable traffic and network conditions by constraining the risk rather than just the expected value of QoS degradation [10]. DR is also a common technique for bridging the simulation-to-reality gap by randomizing the environment parameters during training [11]. In addition, this RL agent is fed with the output of an external traffic prediction module to avoid overfitting to any specific traffic pattern. Although the current evaluation is confined to the radio resource scaling and a single slice, it demonstrates the efficacy of the proposed solution. The contributions of this work are:

- We develop a novel framework for dynamic resource scaling which consists of a regression-based network model, risk-constrained DRL agent, and a traffic prediction module. By training the RL agent offline using random traffic, we have a generalizable agent that does not require any prior knowledge of online slice traffic patterns.
- We evaluate the effectiveness of the proposed approach against traditional, and constrained DRL-based models which encompass the state-of-the-art. In general, the proposed approach performs relatively better than others while also showing generalization to previously unseen traffic and network conditions.

- We assess the robustness of the proposed approach under varying network conditions (*e.g.*, queue congestion), and inaccurate traffic predictions and demonstrate that it can effectively scale resources even under worse-case scenarios.
- We show that our pre-trained model can be fine-tuned for increased performance while meeting QoS degradation constraint and maintaining its generalization capability.

The rest of the paper is organized as follows. In Section II, we provide an overview of CDRL, and discuss the related works. This is followed by a formal definition of the dynamic resource scaling problem in Section III. Section IV delineates the proposed solution. Finally, after showcasing the results in Section V, we conclude in Section VI and instigate future research directions.

II. BACKGROUND AND RELATED WORKS

A. Constrained DRL—A Primer

In both traditional and constrained RL, the sequential decision making and interaction of an agent with its environment can be formally described using a Markov Decision Process (MDP) and a Constrained Markov Decision Process (CMDP), respectively. A finite-horizon MDP can be defined by the tuple $(O, A, P, y, \rho_0, \gamma)$, where O is the state space, A is the action space, $y : O \times A \to \mathbb{R}$ is the reward function, $P : O \times A \times O \to [0, 1]$ is the state transition probability distribution, ρ_0 is the initial state distribution, and γ is the discount factor that specifies the relative importance of future rewards. When state transition probabilities are unknown, RL can be adopted to find a policy $\pi : O \times A \to [0, 1]$ that can maximize the expected discounted reward defined as $J(\pi) = \mathbb{E}_{(o_t, a_t) \sim \rho_\pi} [\sum_t \gamma^t y(o_t, a_t)]$, where o_t and a_t are the state and action at time step t, respectively, and ρ_π denotes the state-action distribution induced by following policy π. When the action and state spaces are large and/or continuous, the policy is learned using a parameterized deep neural network, which is known as DRL.

CMDPs extend MDPs by adding cost functions $c : O \times A \to \mathbb{R}$ such that a CMDP is defined as $(O, A, P, y, c, \rho_0, \gamma)$. Following a policy π, the cost distribution can be modeled as $p^\pi(c|s, a)$. The expected discounted cost $J_c(\pi)$ is analogous to $J(\pi)$ and is obtained by replacing the reward y with cost c in the corresponding equation, *i.e.*, $J_c(\pi) = \mathbb{E}_{(o_t, a_t) \sim \rho_\pi} [\sum_t \gamma^t c(o_t, a_t)]$. In CMDPs, the objective is to find an optimal policy that maximizes $J(\pi)$, but also keeps $J_c(\pi)$ under a certain pre-defined threshold c_{thresh}. Formally, this objective can be written as [12]:

$$\pi^* = \max_\pi J(\pi), \tag{1}$$
$$\text{s.t.} \quad J_c(\pi) \le c_{thresh}.$$

There are numerous approaches for solving CDRL problems (*cf.*, [13]). In general, these methods are either based on Constraint Policy Optimization (CPO) [12], or Lagrangian relaxation [14, 15]. CPO builds upon the TRPO algorithm [16] by adding constraint satisfaction, leading to a monoton-

ically improving policy that guarantees constraint satisfaction throughout training. On the other hand, Lagrangian relaxation methods work by relaxing constraints using Lagrangian multipliers, and updating decision variables and multipliers in an iterative manner using gradient ascent/descent. SAC-Lagrangian is one of these algorithms which is based on the actor-critic framework [14]. It utilizes an actor network to represent the policy, and two critic networks to learn the expected reward and cost for any given state and action. These critic networks are then utilized to update the policy network to maximize the return while satisfying expected cost constraints.

Although the methods discussed above satisfy a constraint on the expected value of the discounted cost distribution p^π, they do not constrain its variation. In this case, there remains a considerable probability of high-cost episodes. As a result, such methods can not be used in safety-critical applications, where it is crucial for the learned policies to be robust. For this case, rather than the expected cost, the safety metric of interest is defined in terms of the Conditional Value-at-Risk (CVaR) of the cost, *i.e.*,

$$CVaR_\alpha = \mathbb{E}_{p^\pi}[C|C \geq F_C^{-1}(1-\alpha)], \qquad (2)$$

where $F_C(.)$ is the cumulative distribution function (CDF) of p_π, and α is the risk level hyper-parameter with smaller values leading to more risk-averse policies. In this case, a policy is safe if it satisfies a constraint on this new safety measure:

$$\mathbb{E}_{p^\pi}[C(o_t, a_t)|C(o_t, a_t) \geq F_C^{-1}(1-\alpha)] \leq c_{thresh}, \quad \forall t, \quad (3)$$

where $C(o_t, a_t)$ is the discounted cumulative cost of policy π from point (o_t, a_t).

To address this issue, Yang et al. [10] recently proposed the Worst-Case Soft Actor Critic (WCSAC) algorithm that extends SAC-Lagrangian by replacing the cost critic with a distributional one. Specifically, they model p^π as a Gaussian distribution, and utilize two neural networks to predict its mean $Q_\pi^c(s_t, a_t)$ and variance $V_\pi^c(s_t, a_t)$ given a state-action pair (s_t, a_t). The new safety measure, *i.e.*, CVaR, can then be calculated using a closed-form equation:

$$CVaR_\alpha = \Gamma_\pi(s_t, a_t, \alpha) = Q_\pi^c(s, a) + \alpha^{-1}\phi^{-1}(\Phi^{-1}(\alpha)), \quad (4)$$

where $\phi(.)$ and $\Phi(.)$ represent the probability distribution function (PDF) and CDF of the standard normal distribution, respectively. The policy can be updated by minimizing the following KL-Divergence [17, 10]:

$$\pi^{'} = \min_\pi \mathcal{D}_{KL}\left(\pi(.|s_t)\left\|\frac{exp(\frac{1}{\beta}Q_\pi^r(s_t, .)) - k\Gamma_\pi(s_t, ., \alpha)}{Z^\pi(s_t)}\right.\right) \tag{5}$$

where $\pi^{'}$ denotes the updated policy, $Q_\pi^r(s, a)$ denotes the state-action value and $Z^\pi(s_t)$ is a normalization factor. β and k are Lagrangian multipliers that are updated iteratively to determine the trade-off between the policy entropy, the reward, and the safety measure. For a detailed exposition of the algorithm, we refer to [10]. As WCSAC is able to satisfy a percentile-based constraint (CVaR) over a QoS degradation distribution, compared to expectation-based methods, it is more adaptable to different traffic patterns and network condition scenarios.

B. Dynamic Resource Scaling

Dynamic resource scaling requires an accurate model of the relationship between the allocated resources, traffic volume, and achieved QoS. In this section, we present an overview of the methodologies commonly used to model this relationship and review state-of-the-art dynamic resource scaling works that fall into these categories.

1) Resource Isolation

In this approach, QoS is defined in terms of *resource isolation* that depends on a simple comparison between the required resources and the resource allocation, *i.e.*, $\mathbb{P}(r_t^s \geq v_t^s)$, where r_t^s and v_t^s respectively denote the amount of allocated resource to a slice, and its minimum required resource to meet the QoS threshold at a specific time t. This type of network modeling requires v_t^s to be known or easily predictable which may not be feasible in practice. Additionally, this may be undesirable for cases when the slices experience bursty traffic since even if only a small amount of resources are allocated to the slice during short periods of high traffic, high isolation can still be achieved. Based on this model, Li et al. [4] optimize the resource utilization at each time step while respecting resource isolation constraints modeled by chance constraints. These constraints are then approximated using a data-driven approach, which converts the problem to a semi-definite programming (SDP) problem that can be solved optimally.

2) Model-Driven QoS Isolation

This approach adheres to *QoS isolation* by calculating the minimum required resources based on a given QoS threshold, but only considers an abstract model of the network for which there exists concrete theoretical groundwork. Queues are one of the commonly used models in this category. In this case, either queuing theory or queue simulation can be utilized to compute different performance measurements for the packets entering the queue. By defining the queue arrival-rate in terms of slice traffic and determining the service-rate based on resource allocation, the queuing-time for any packet can be determined. The average queuing-time can represent the QoS of a slice, defined as the packet latency. In general, the drawback of this type of modeling is that it can only deal with simple network-level QoS metrics which are based on a single resource type, as it can be challenging to model the service-rate of the queue based on multiple heterogeneous metrics. Solutions based on queue simulation also suffer from high computational complexity when the traffic volume increases since they deal with the traffic at the packet-level. Finally, queues might not be able to accurately model network components such as RAN, as there can be additional factors that affect the latency such as the presence of multiple queues, channel effects or application-specific idiosyncrasies.

Some of the works that use a queue for modeling the network include [18, 19, 2, 3, 8]. Papa et al. [18] and Kasgari and Saad [19] assume the delay experienced by each slice can be exactly modeled by considering an M/M/1 queue model. The authors then leverage the Lyapunov optimization method
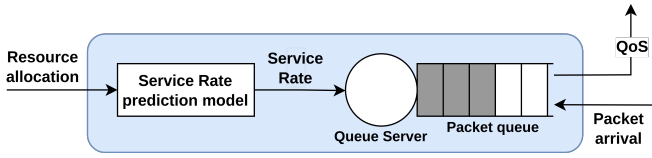
Fig. 1: Regression-Queue based network model

to minimize long-term resource usage under QoS constraints. Li et al. [3] and Hua et al. [8] utilize queue-based simulation and optimize the weighted sum of RE and SLA satisfaction ratio. For this purpose, [3] utilizes Deep Q-learning, whereas [8] leverages a combination of distributional DRL with generative adversarial network (GAN). However, in these works, the trade-off between RE and SLA satisfaction ratio depends on the weights assigned to these in the reward formulation, which have to be manually tuned. As a result, these approaches are not able to satisfy QoS degradation constraints. To ensure these constraints, the authors in [2] propose constrained RL for resource allocation. But this approach requires the slice's traffic to be known in advance and is unable to generalize to previously unseen traffic patterns during testing.

3) Data-Driven QoS Isolation

In this method, QoS isolation is assured and the behavior of the network is learned using data-driven approaches based on historical data. As there is no public dataset available, this mandates access to either a testbed or a production network where slices can be easily created and scaled. Due to this reason, there are only a limited number of works that utilize this approach. This type of network modeling is classified further into the following categories:

Regression-Queue Based: As shown in Fig. 1, this approach is also based on a queue, with the difference that a regression model, learned using a network dataset, is used to predict the service rate of the queue. Since multiple resources can be used as the input to the service rate prediction model, this type of network modeling is not restricted to only one resource type. However, this approach remains constrained by other drawbacks associated with queue-based model-driven approaches. Liu et al. [7] used offline training based on this model to minimize multi-domain resource utilization under capacity and end-to-end delay constraints of slices. They relax SLA satisfaction constraints and incorporate them into the objective by utilizing the primal-dual Lagrangian method, and train the model using conventional actor-critic DRL methods. Although offline training reduces the time for the RL model to converge in the real environment, similar to [2] it suffers from low generalizability.

Online Learning Based: In this approach, rather than modeling a network, the data from a production network during a slice's operation is leveraged to learn the resource scaling algorithm. In a subsequent work to [7], Liu et al. [9] proposed an RL-based approach with online-only training and behaviour cloning, which can deal with various QoS metrics. However, considering the granularity of updates in a real network (*i.e.*, around 15 minutes [9]), the adoption of this

method in a production environment is impractical due to the long convergence time.

III. PROBLEM STATEMENT

Let TTI be the transmission time interval, *i.e.*, small isometric time intervals into which the time horizon is divided. We define T_Δ, consisting of N TTIs, to be the decision time interval (DTI), *i.e.*, the minimum time interval required between resource scaling decisions due to practical limitations such as the time required for horizontally or vertically scaling virtual machines (VMs). Let $\mathbf{T}$ denote the set of starting points of all DTIs. For slice $s \in S$, we denote the traffic at the DTI starting at t by column vector $\mathbf{x}_t^s = [x_{tn}^s]_{n \in [N]}$, where $x_{tn}^s \in \mathbb{N}$ is the traffic at the nth TTI within DTI starting at t. We also define $\mathbf{r}_t^s = [r_t^{s,k}]_{k \in [K]}$ with $r_t^{s,k} \in \mathbb{R}$ to be the K different types of resources allocated to slice s over DTI t. We assume that the traffic, measured in users/sec, stays constant within a TTI and the resources allocated to a slice are divided fairly among all its users.

Let vector $\mathbf{q}_t^s = [q_{tn}^s] \in \mathbb{R}^{n \times 1}$ represent per-user QoS of each slice at DTI t, which can be determined using a network model. We are interested in QoS degradation probability at any point in time, which can be defined as the portion of traffic that receives QoS below the minimum threshold, *i.e.*,

$$\beta_t^s = \frac{\sum_{\tau \in T : \tau \leq t} \mathbf{x}_\tau^{s\top} \, \mathbb{1}_{[q_\tau^s \leq q_{thresh}^s]}}{\sum_{\tau \in T : \tau \leq t} \mathbf{1}_N \, \mathbf{x}_\tau^s}, \qquad (6)$$

where $\mathbb{1}_{[q_\tau^s \leq q_{thresh}^s]}$ is an indicator vector whose n^{th} element equals to 1 only when $q_{\tau n}^s \leq q_{thresh}^s$, $\mathbf{1}_N$ is a 1-vector of size N and q_{thresh}^s is the expected minimum QoS of slice users. Finally, based on the introduced notations, the dynamic resource scaling problem can be formulated as:

$$\min_{\mathbf{r}} \quad \frac{1}{|T|} \sum_{t \in T} \sum_{s \in \mathbf{S}} \eta^\top \mathbf{r}_t^s$$

$$\text{s.t.} \quad \mathbb{E}\left(\beta_{max(T)}^s\right) \leq \beta_{s,thresh}, \quad \forall s \in S \qquad (7)$$

$$\sum_{s \in S} \mathbf{r}_t^s \leq \mathbf{R}, \quad \forall t \in T,$$

where $\eta \in \mathbb{R}^{k \times 1}$ is the resource normalization vector, $\mathbb{E}$ is the expectation over the distributions of QoS and traffic, $\mathbf{R} = [R_k]_{k \in [K]}$ represents the capacities of resources, and $\beta_{s,thresh}$ is the acceptable *QoS* degradation threshold for slice s.

IV. PROPOSED FRAMEWORK

In this section, we describe the proposed framework which is shown in Fig. 2. It consists of three components: a future traffic forecast module, an RL-based dynamic resource scaler and a network model. Note that we do not expound on the traffic prediction module, since it is a well-studied topic and there are off-the-shelf packages available for it (*e.g.*, [20]).

A. Network Model

We propose using a regression-based network model. Similar to the state-of-the-art [8, 9], we assume that there are only a limited number of standard slice types (*e.g.*, URLLC, eMBB, mMTC), and their QoS can be monitored by creating
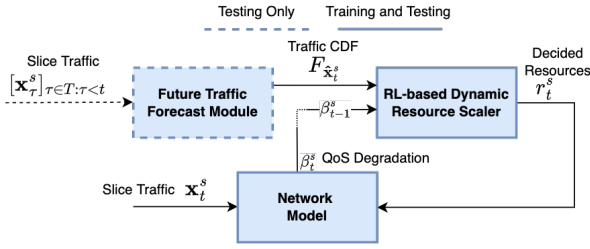
Fig. 2: Overview of the proposed framework

isolated slices and scaling their resources. Compared to the regression-queue based model in Section II-B3, the queue and the service rate prediction model are replaced by a single regression model. This regression model learns the function $f(x, \mathbf{r}) : \mathbb{N} \times \mathbb{R}^K \rightarrow \mathbb{R} \times \mathbb{R}$, which maps the traffic x_{tn}^s and resource allocation vector $\mathbf{r}_t^s$ of nth TTI of DTI t to Gaussian distribution parameters μ_{tn}^s and σ_{tn}^s. The QoS can then be sampled using these parameters, *i.e.*, $q_{tn}^s \sim \mathcal{N}(\mu, \sigma)$. In this method, we assume that the effect of past traffic on the QoS is only transient, and can be subsumed by the distribution of QoS around the mean. This is because it can be infeasible to gather a dataset that correlates the past traffic as well as the current traffic, and a resource allocation to a certain QoS.

Based on the complexity of the network (*e.g.*, in terms of the number of resources), a simple query-based method, a linear regression model, or neural networks could be used to learn the function $f(x, \mathbf{r})$. To train this model, the dataset is gathered by performing a grid search over different resource allocations at different traffic levels, and measuring the corresponding QoS under varying network conditions. As each QoS value sampled from the Gaussian distribution $\mathcal{N}(\mu, \sigma)$ can be written as $\mu - d\sigma$, we use the parameter $d \in \mathbb{R}$ to represent network conditions such as queue congestion and channel quality. When d is known, *i.e.*, in deterministic network conditions, the QoS vector at DTI t can be computed as:

$$q_t^{s,det}(d) = \left[\mu_{tn}^s - d\sigma_{tn}^s \right]_{n \in [N]}. \tag{8}$$

As opposed to queue-based models, the regression-based model can deal with scaling heterogeneous resources, and can predict different types of QoS. Additionally, the complexity of this approach does not depend on the traffic volume and it is not restricted to packet-level traffic.

B. Risk-constrained DRL-based Resource Scaling Algorithm

For dynamically scaling the resources allocated to a slice, we propose using the WCSAC [10] algorithm trained using the training loop shown in Fig. 2. Each step in an episode, *i.e.*, $t \in T$, corresponds to a DTI.

1) CMDP Formulation

At step t, the RL agent's *state* o_t includes the CDF of the traffic of the next DTI over its TTIs, $F_{x_{tn}^s}(.)$, and the QoS degradation up until that time, β_{t-1}^s, for each slice. Note that since the traffic is in terms of users/sec, $F_{x_{tn}^s}(a) = \mathbb{P}(x_{tn}^s \leq a)$ is a discrete CDF which can be represented by a vector of finite size. As opposed to existing works [7, 2], instead of mean, we include the CDF of the future traffic in the agent's

state space. This is because multiple different distributions can have the same mean, and can lead the agent to learn spurious correlations between actions in a given state and their effect on the environment [21]. Additionally, β_{t-1}^s, which is calculated using Eq. (6), acts as a feedback mechanism for the agent to adjust the resource allocation for the future DTIs based on the effect of past actions on QoS degradation. The agent's *action* a_t corresponds to the resource scaling decision $\mathbf{r}_t^s$ for each slice and we define the *reward* function as:

$$y(o_t, a_t) = 1 - \sum_{s \in S} \eta^\top \mathbf{r}_t^s. \tag{9}$$

Finally, the cost function is defined as the marginal QoS degradation where the denominator is the sum of traffic over the episode, as shown below:

$$c(o_t, a_t) = \frac{\mathbf{x}_t^{s\top} \mathbb{1}_{[q_t^s \leq q_{thresh}^s]}}{\sum_{\tau \leq max(T)} \mathbf{1}_N \mathbf{x}_\tau^s}. \tag{10}$$

2) Training

Fig. 2 shows how WCSAC algorithm is integrated with the network model and future traffic prediction module during training and testing. To make the learned policy generalizable across different traffic patterns and network conditions, we utilize uniform DR during training. For this purpose, at the start of each episode, the traffic for each DTI, $\mathbf{x}_t^s$, is generated by sampling i.i.d from a randomized distribution, *i.e.*, $F_{x_{tn}^s}$. By randomizing the traffic distribution and sampling the QoS from the network model distribution during training, the risk-constrained RL agent learns to maximize the reward while keeping the QoS degradation under the specified threshold β_{thresh}^s even in the worst-case scenarios. These extreme conditions can arise due to congestion in the network, interference by other slices, or a specific traffic pattern (*e.g.*, bursty) that may require higher resource allocation.

Since the future traffic, $\hat{\mathbf{x}}_t^s$, is unknown during testing, an external traffic prediction module is used to predict its CDF. The QoS degradation during testing can be computed using actual slice traffic, and by utilizing either the offline network model or actual QoS reported by users. Note that by including the future traffic distribution in the state space, the agent is not confounded by the varying effect of different resource allocations under varying traffic levels.

V. EXPERIMENTS

A. Testbed and Simulation Setup

To gather the required dataset for creating a realistic network model, we deploy the SDRAN-in-a-Box (RiaB) [22] on an Intel NUC PC. RiaB utilizes Kubernetes to deploy end-to-end SD-RAN components that include EPC, emulated RAN and user equipment (UE), and ONOS RAN Intelligence Controller (RIC). The testbed can be used to create slices, associate UEs to them, and scale their allocated resources dynamically. However, there are limitations on the number of UEs per slice and the granularity of resource allocation. To circumvent the former restriction, we count the number of parallel connections made by an emulated UE as the slice's users. For the
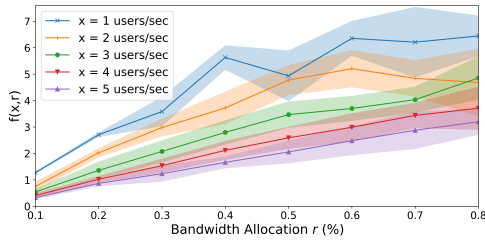
Fig. 3: Regression-based network model

latter, we allocate radio resources in intervals of 10% of the total resource capacity. For simplicity, during evaluation, we consider a single type of slice (*i.e.*, eMBB) and resource (*i.e.*, radio resource in Mbps), and thus we omit the subscript s and k in the subsequent sections. The slice is tailored for eMBB services by deploying an image download application over it. We define the *QoS* of the slice in terms of frames per second (Fps), *i.e.*, the number of times the image can be downloaded from a local server within a second.

To gather the QoS dataset, we perform a grid-search over the alterable parameters by varying the traffic from 1 to 5 users/sec, and the corresponding resource allocation from 10% to 80%. To encompass different network conditions, we repeat this grid-search multiple times and gather the data for 30 seconds for each point in the grid per iteration. Fig. 3 shows the corresponding network model. The shaded area around the mean represents the distribution of QoS achieved at different network conditions. The network model is fed with this data to predict the distribution of *QoS* at any traffic and resource allocation. In this paper, we learn the network model using a simple query-based model that returns the mean and standard deviation of the QoS as the size of the grid is small.

For training and testing the different scaling approaches, we set the QoS threshold and the acceptable QoS degradation threshold, *i.e.*, q_{thresh} and β_{thresh}, to 2.0 Fps and 10%, respectively. As we can see in Fig. 3, 80% bandwidth allocation is required to have no (*i.e.*, 0) QoS degradation under the highest traffic rate (*i.e.*, 5 users/sec) and worse-case network conditions which occur at around $d = -2$. For WCSAC, we set the risk level hyper-parameter α to 0.1. DTI and TTI are set to one minute and one second, respectively. As discussed previously, we assume the traffic varies across TTIs, but remains constant within each TTI. For testing, we utilize the real-world traffic pattern of Internet events in Milan from the Telecom Italia dataset [1]. We scale this traffic pattern to represent the traffic of 1 to 3 users/sec and add truncated Gaussian noise $\mathcal{N}(0, 0.75^2)$ to create dynamic traffic within each DTI. We refer to the resulting traffic curve as the *dataset traffic* throughout this section. The reported results are averaged over 100 episodes of 10 minutes in length to ensure statistically stable results.

B. Comparative Approaches

For evaluating the proposed framework comprising the regression-based network model, WCSAC algorithm and described training paradigm, we implement a number of DRL-based approaches, and a heuristic method. These approaches encompass the baselines and emulate a number of the solutions proposed in contemporary literature. For training and testing these methods, we utilize the proposed regression-based network model for a fair comparison.

1) DRL-based

Avg-CPO, Avg-PPO: A number of DRL-based works in the literature train the RL agent on the same traffic pattern which is also used during testing [2, 7, 8, 3]. Clearly, these approaches fail to perform well if the test traffic varies from the training traffic. To encompass both constraint-aware and traditional DRL approaches presented in these works, we implement the CPO [12] and the PPO [23] algorithms and refer to them as Avg-CPO and Avg-PPO, respectively. For Avg-CPO, the reward and cost are the same as the ones used for WCSAC, described in equations (9) and (10). Since a cost constraint cannot be incorporated with PPO, we formulated its reward function as the sum of the reward and the cost used for WCSAC, weighted by w_{RE} and w_{QoS}, respectively. The values for these weights are manually fine-tuned. Some of these works assume that the RL agent learns to scale the slice resource allocation only based on past traffic information [7, 2]. This requires the RL agent to predict the traffic trend in addition to resource scaling. However, for a fair comparison, we include the CDF of the future traffic in the RL agent's state.

WC-CPO: As described in Section II-A, CPO is designed to constrain the expected cost under a specific limit while learning actions that maximize the reward. To make this approach risk-aware, *i.e.*, robust to different traffic distributions and network conditions that arise during testing, we leverage cost-shaping to ensure that the QoS degradation stays below β_{thresh} for all scenarios. For this purpose, in addition to the per-step cost in Eq. (10), the agent is given an additional cost at the end of each episode. This cost is an exponential function of QoS degradation that is in excess of the threshold, *i.e.*, $\gamma(e^{[\beta_{max(T)} - \beta_{thresh}]^+} - 1)$ where $[x]^+$ denotes the maximum of x and 0, and γ is a scaling factor which controls the degree of risk adverseness. Since the average cost is affected significantly by this exponentially weighted cost component, the agent learns to keep the QoS degradation for scenarios giving rise to high QoS degradation under the threshold. We refer to this CPO-based approach with the shaped cost function as worst-case CPO, and denote it by WC-CPO.

2) Heuristics-based

Pred-Alloc: As an upper bound for evaluating DRL-based approaches, we also implement a simple heuristic. This algorithm is based on two simplifying assumptions. First, it assumes the traffic in any DTI to be constant and equal to the peak of the curve predicted by the traffic prediction module. The second simplification is that it considers somewhat extreme and deterministic network conditions, *i.e.*, at any given traffic and resource allocation, the QoS vector is equal to $q_t^{det}(-2)$ defined in Eq. (8). These assumptions make the resource allocation at each DTI trivial since it can be calculated using a simple grid-search. However, it can lead

(a) Avg. BW allocation per epoch

(b) Avg. QoS degradation per epoch ($\beta_{thresh} = 10\%$)

(c) Avg. BW allocation per epoch

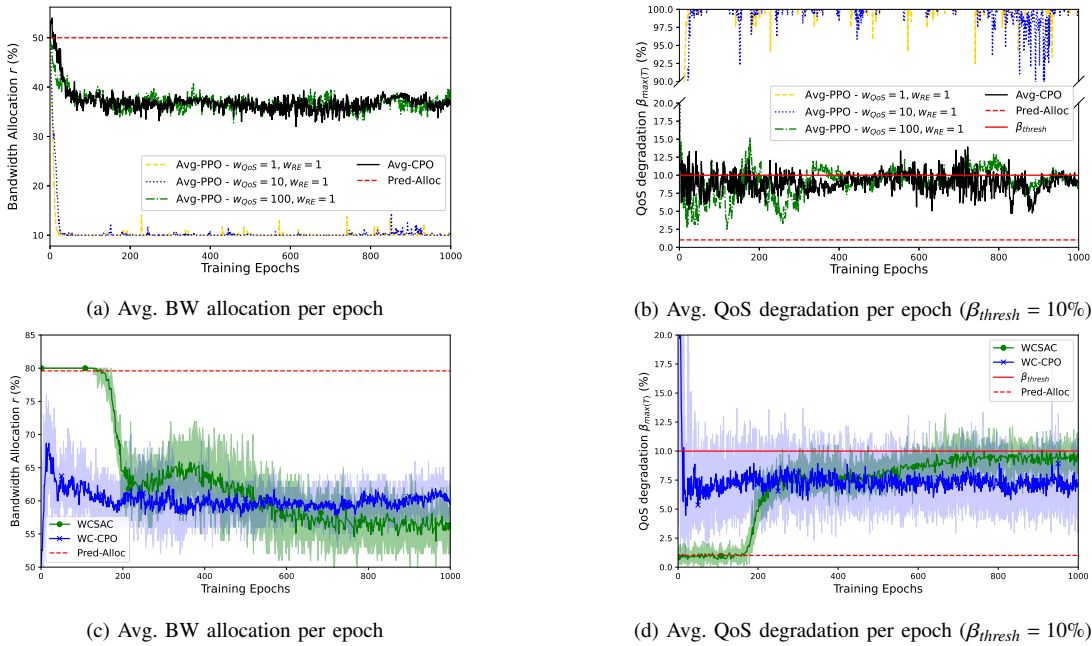(d) Avg. QoS degradation per epoch ($\beta_{thresh} = 10\%$)

Fig. 4: Training process of different methods; (a),(b) are trained on the dataset traffic, and (c),(d) are trained on randomized traffic.

to significant over-provisioning and non-optimal results. We refer to this approach as Pred-Alloc.

C. Training Performance

For training, we use two dense layers each having 64 neurons for the policy network for the different algorithms. The learning rates can differ across methods and are determined experimentally. We train Avg-CPO and Avg-PPO on the dataset traffic. For Avg-PPO, we set w_{RE} to 1, and tune w_{QoS} to achieve a nearly optimized tradeoff between resource allocation and QoS. Fig. 4a and Fig. 4b show the mean bandwidth allocation and mean QoS degradation of these methods during the training phase. We observe that Avg-PPO is able to keep QoS degradation under 10% with w_{QoS}=100. When evaluated, on the same traffic distribution as the one used for training (*cf.*, Table I), Avg-CPO and Avg-PPO (w_{QoS}=1, w_{RE}=100) achieve the same mean bandwidth allocation of 37.2%, and a mean QoS degradation of 8.42% and 8.3%, respectively. This bandwidth allocation is 12.8% lower than Pred-Alloc, which allocates a 50.8% mean bandwidth in this scenario.

Fig. 4c and Fig. 4d show the training progress of WC-CPO and WCSAC approaches on the randomized traffic. The shaded area around the curves denotes the minimum and maximum of the corresponding quantities. The variation range of mean bandwidth and QoS degradation represent the ability of the two approaches to operate under different network conditions and traffic distributions. As reported in Table I, when evaluated on dataset traffic, WC-CPO and WCSAC lead to a mean bandwidth allocation of 52.5% and 39%, and a mean QoS degradation of 1.06% and 6.73%, respectively. In this scenario, the respective approaches lead to 2.5% higher, and 11.8% lower bandwidth allocation compared to Pred-Alloc.

TABLE I: Evaluation Performance of Different Approaches

Method	Dataset Traffic		Offset Dataset Traffic	
	BW allocation(%)	QoS degradation(%)	BW allocation(%)	QoS degradation(%)
Avg-PPO	37.2	8.3	39.7	40.8
Avg-CPO	37.2	8.42	39.4	44.5
WC-CPO	52.5	1.06	53.9	8.98
WCSAC	39	6.73	56	7.19
Pred-Alloc	50.8	1.01	77.7	0.995

D. Generalization to Unseen Traffic Pattern

To assess generalization, we offset the traffic curve by 2 users/sec, *i.e.*, the trained agents are evaluated on the *dataset traffic* that represents 3 to 5 users/sec. The results are reported in Table I. In this scenario, Avg-CPO and Avg-PPO lead to similar mean bandwidth allocation of 39.4% and 39.7%, but a mean QoS degradation of 44.5% and 40.8%, respectively. The results assert that these methods over-fit to the training traffic pattern and are unable to generalize to previously unseen traffic, leading to high QoS degradation beyond the threshold. Note that, in this scenario, although Pred-Alloc leads to a high bandwidth allocation of 77.7%, it keeps the mean QoS degradation $\beta_{max(T)}$ under 1%. WC-CPO and WCSAC lead to a mean bandwidth allocation of 53.9% and 56%, and a mean QoS degradation of 8.98% and 7.19%, respectively. Although resulting in better generalization than the previous approaches, these results show that WC-CPO overfits to the worse-case traffic scenarios due to the exponential cost, while WCSAC adapts well to both average and worse-case scenarios.

E. Robustness to Varying Network Conditions and Inaccurate Traffic Predictions

First, we evaluate the proposed approach, on the dataset traffic, under different network conditions. To control the severity of network conditions (*i.e.*, from favorable to worst-case), we assume them to be deterministic. In this case, $q_t^{det}(-3)$ and $q_t^{det}(+3)$ represent worst-case and best-case network
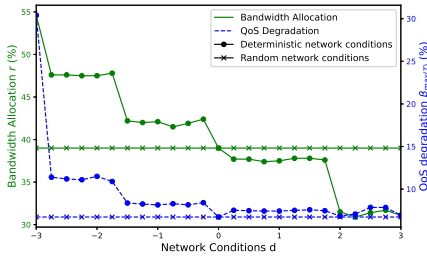
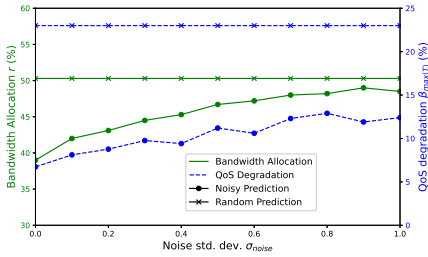Fig. 5: WCSAC performance at different network conditions ($\beta_{thresh} = 10\%$)

Fig. 6: WCSAC performance with inaccurate traffic prediction ($\beta_{thresh} = 10\%$)
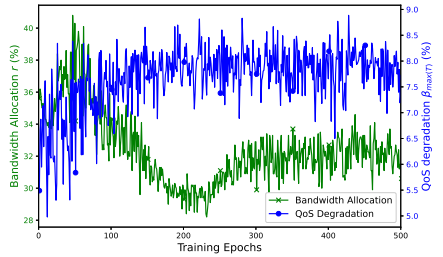
Fig. 7: WCSAC avg. BW and QoS degradation during fine-tuning on deterministic network condition $q_t^{det}(+1)$ with $\beta_{thresh} = 10\%$

conditions, respectively. In Fig. 5, the horizontal lines show the bandwidth allocation and QoS degradation when the QoS is randomly sampled using the network model output, while the curves show their trend under static network conditions with varying d. Evidently, as the network conditions aggravate, both the QoS degradation and bandwidth allocation increase. The QoS degradation stays within the threshold β_{thresh} as long as $d \geq -1.5$. For $1.5 < d < 3$, the QoS degradation is slightly higher than the threshold. Finally, at $d = -3$, the QoS degradation hits 30.4%. This is expected since WCSAC constrains only the CVaR of the cost distribution at a given risk level (*i.e.*, α) under the limit. On the other hand, as network conditions become more favorable, it results in an increase in resource efficiency and a decrease in QoS degradation.

Subsequently, we evaluate the sensitivity of the proposed approach to inaccurate traffic prediction. For this purpose, we use dataset traffic but introduce truncated Gaussian noise $\mathcal{N}(0, \sigma_{noise}^2)$ to the traffic probability distribution, at each step. After normalization, this resulting CDF is fed as the state to the RL agent. In Fig. 6, the horizontal lines show the resource allocation and the QoS degradation when the traffic prediction is fully random (*i.e.*, $\hat{x}_{tn} \sim \mathcal{U}(1,5)$), while the curves show these metrics as σ_{noise} increases. We can observe that when prediction inaccuracy increases (*i.e.*, for higher σ_{noise}), both the QoS degradation and bandwidth allocation increase. However, even with fully randomized traffic prediction, the proposed approach shows a QoS degradation that is only 13% higher than the threshold β_{thresh}. This is attributed to the RL agent taking the current QoS degradation level into account when allocating resources, which acts as a feedback mechanism and allows the agent to allocate higher resources even if the traffic prediction is inaccurate.

F. Fine-tuning Performance

In a production network, the online data regarding a slice's traffic pattern and the network condition can be used to continuously improve the network model and resource scaling algorithm with fine-tuning. Since the traffic pattern and network conditions do not change during testing, learning to adapt to other scenarios is not required during fine-tuning. Therefore, we train the WCSAC algorithm with $\alpha = 0.99$ (*i.e.*, risk-neutral), and at a lowered learning rate for 500 epochs. Fig. 7 shows the bandwidth allocation and the QoS degradation when WCSAC is fine-tuned on the dataset traffic

with a favorable and deterministic network condition, *i.e.*, $q_t^{det}(+1)$. The optimal bandwidth allocation is achieved within 250 epochs while maintaining QoS degradation below the set threshold of 10%. We evaluate the fine-tuned model using the best checkpoint during training. Evaluation results on the dataset traffic show a mean bandwidth allocation of 29.7% and a mean QoS degradation of 7.85%. Compared to previous results in Fig. 5, there is 7.8% less bandwidth usage while maintaining QoS degradation under the threshold. We also test this fine-tuned algorithm on the dataset traffic offset by 2 users, with randomly sampled network conditions, which leads to a mean bandwidth allocation of 56.9% and a mean QoS degradation of 9.06%. Comparing these to the results in Table I, we can conclude that the gain in performance during fine-tuning comes at only a slight cost to the generalization ability of the algorithm.

VI. Conclusion

In this work, we developed a novel framework utilizing a risk-constraint DRL algorithm, a regression-based network model and a traffic prediction module, for dynamically scaling slice resources in a 5G network. We proposed a regression-based network model to learn the distribution of QoS at any resource allocation and traffic, and utilized that to train the RL agent offline. To achieve generalization, the RL agent is fed with future traffic and current QoS degradation level, and is trained on randomized traffic. Our results show that the resulting RL agent is able to show similar performance as RL agents trained on the exact test-time traffic. Furthermore, it is able to maintain its performance across different real-world traffic patterns. Additionally, we demonstrated the agent's robustness under extreme network conditions and inaccurate traffic prediction. Finally, we showed that fine-tuning can be used to improve the performance further when the network condition or the slice traffic pattern is known in advance. As a future direction for this work, we intend to extend the evaluations over multiple types of resources and slices. We also plan to validate the algorithm on an expansive 5G testbed.

Acknowledgement

REFERENCES

[1] Telecom Italia, "Telecommunications - SMS, Call, Internet - MI," 2015. [Online]. Available: https://doi.org/10.7910/DVN/EGZHFV

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

[3] R. Li, Z. Zhao, Q. Sun, C.-L. I, 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.

[4] J. Li, J. Liu, T. Huang, and Y. Liu, "DRA-IG: the balance of performance isolation and resource utilization efficiency in network slicing," in *IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[5] J. X. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, "Overbooking network slices through yield-driven end-to-end orchestration," in *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2018, pp. 353–365.

[6] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *USENIX Annual Technical Conference (ATC)*, 2018, pp. 731–743.

[7] Q. Liu, N. Choi, and T. Han, "Constraint-aware deep reinforcement learning for end-to-end resource orchestration in mobile networks," in *IEEE International Conference on Network Protocols (ICNP)*, 2021, pp. 1–11.

[8] 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.

[9] Q. Liu, N. Choi, and T. Han, "OnSlicing: online end-to-end network slicing with reinforcement learning," in *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2021, pp. 141–153.

[10] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. J. Spaan, "WCSAC: worst-case soft actor critic for safety-constrained reinforcement learning," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 639–10 646.

[11] M. Xu, Z. Liu, P. Huang, W. Ding, Z. Cen, B. Li, and D. Zhao, "Trustworthy reinforcement learning against intrinsic vulnerabilities: Robustness, safety, and generalizability," 2022. [Online]. Available: https://arxiv.org/abs/2209.08025

[12] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," 2017. [Online]. Available: https://arxiv.org/abs/1705.10528

[13] Y. Liu, A. Halev, and X. Liu, "Policy learning with constraints in model-free reinforcement learning: A survey," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[14] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," 2020. [Online]. Available: https://arxiv.org/abs/2002.08550

[15] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," 2018. [Online]. Available: https://arxiv.org/abs/1805.11074

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning (ICML)*, 2015, pp. 1889–1897.

[17] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[18] A. Papa, M. Klugel, L. Goratti, T. Rasheed, and W. Kellerer, "Optimizing dynamic RAN slicing in programmable 5G networks," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[19] A. T. Z. Kasgari and W. Saad, "Stochastic optimization and control framework for 5G network slicing with effective isolation," in *IEEE Annual Conference on Information Sciences and Systems (CISS)*, 2018, pp. 1–6.

[20] O. Triebe, H. Hewamalage, P. Pilyugina, N. Laptev, C. Bergmeir, and R. Rajagopal, "NeuralProphet: Explainable forecasting at scale," 2021. [Online]. Available: https://arxiv.org/abs/2111.15397

[21] F. D. Johansson, U. Shalit, and D. Sontag, "Learning representations for counterfactual inference," 2016. [Online]. Available: https://arxiv.org/abs/1605.03661

[22] "Introduction — SD-RAN Docs 1.4.1-dev documentation," https://docs.sd-ran.org/sdran-1.4/introduction.html, [Accessed 29-Sep-2022].

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347