Received 5 February 2021; revised 11 March 2021; accepted 15 March 2021. Date of publication 23 March 2021; date of current version 5 April 2021.

Digital Object Identifier 10.1109/OJCOMS.2021.3068222

Tomography Based Learning for Load Distribution Through Opaque Networks

SHENGHE XU[1] (Member, IEEE), MURALI KODIALAM[2] (Member, IEEE), T. V. LAKSHMAN[2] (Fellow, IEEE), AND SHIVENDRA S. PANWAR[1] (Fellow, IEEE)

[1]Department of ECE, New York University Tandon School of Engineering, Brooklyn, NY 11201, USA

[2]Network Systems Research, Nokia Bell Labs, Holmdel, NJ 07733, USA

CORRESPONDING AUTHOR: S. XU (e-mail: shenghexu@nyu.edu)

ABSTRACT Applications such as virtual reality and online gaming require low delays for acceptable user experience. A key task for over-the-top (OTT) service providers who provide these applications is sending traffic through the networks to minimize delays. OTT traffic is typically generated from multiple data centers which are multi-homed to several network ingresses. However, information about the path characteristics of the underlying network from the ingresses to destinations is not explicitly available to OTT services. These can only be inferred from external probing. In this paper, we combine network tomography with machine learning to minimize delays. We consider this problem in a general setting where traffic sources can choose a set of ingresses through which their traffic enter a black box network. The problem in this setting can be viewed as a reinforcement learning problem with strict linear constraints on a continuous action space. Key technical challenges to solving this problem include the high dimensionality of the problem and handling constraints that are intrinsic to networks. Evaluation results show that our methods achieve up to 60% delay reductions in comparison to standard heuristics. Moreover, the methods we develop can be used in a centralized manner or in a distributed manner by multiple independent agents.

INDEX TERMS Load distribution, segment routing, reinforcement learning, machine learning.

I. INTRODUCTION

RECENT emerging applications including virtual reality, online or cloud gaming require low delay for acceptable user experience [1], [2]. Minimizing delay by optimizing load distribution through underlying networks is an important task for providers of these services. However, since these services are often "over-the-top" services, the providers do not have full knowledge of the underlying networks and have to make load distribution decisions based purely on inference of the network characteristics from edge-based observations. Inferring network characteristics from external observations, called "network tomography", has been extensively studied. Early work in network tomography focused on the "inverse problem" of estimating traffic matrices from link-level observations only [3]–[5]. In this paper, our interest is in "active tomography" where probes from the network periphery are used to infer internal network characteristics [6]–[9].

We view the network as a black box with most of the network's features of interest for load distribution purposes being not directly observable. Most of the important information for performance optimization is hidden and hard to measure. For example, without information from the Internet Service Providers (ISPs), inferring the routing structure of the network is often an impossible task. While ping and traceroute may provide some insight, routers may not respond to these kinds of probe packets.

We consider the scenario in Figure 1 where there are a set of sources that have traffic to send through the black box network to a set of destinations.The traffic sources know the ingresses into the network and can send probes though the network to different endpoints. The probes are echoed by the endpoints; returning probes can be observed to determine the network's response to probing and infer behavior of the underlying network. Sources have the choice of distributing their traffic over the set of ingresses to the network – a source

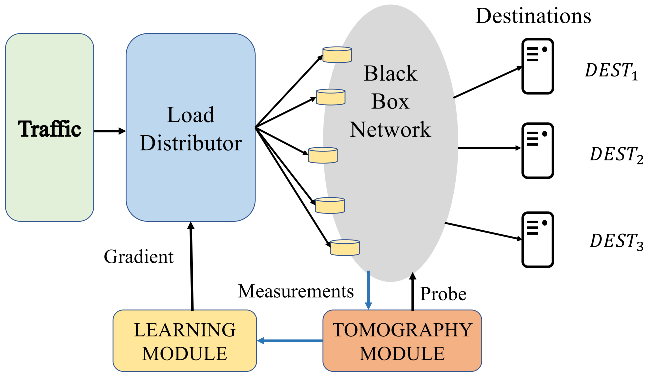


FIGURE 1. Sending traffic through a black box network.

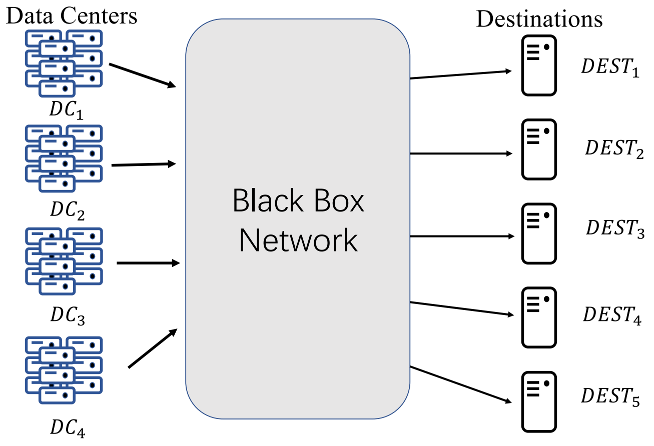


FIGURE 2. Load distribution from data centers to black box network.

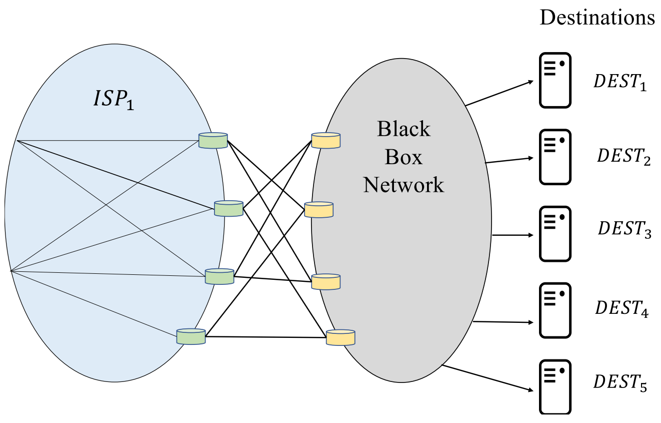


FIGURE 3. Egress picking to minimize delays.

may send all its traffic to a destination through a particular ingress or it may decide to distribute its traffic to the different ingresses in proportions that minimize the total delay through the network. Because the network is a black box, the information that is needed to make the optimal distribution choice can only be gleaned from external observations of responses to past actions. For the load-distribution problem, we use the history of tomography-obtained responses to past actions to train a neural network which we then combine with reinforcement learning to make future load distribution decisions that minimize delay through the underlying black box network. Unlike network tomography, where the primary goal is network monitoring and measurement, our goal here is automated performance optimization where information obtained through tomography is used to optimize performance through black box networks. To this end, we combine learning-based methods with network tomography to optimize performance through black box networks. In our method, a single neural network can act as both the tomography module and the learning module. By learning from the past results, the agent takes actions to minimize delay and probe the black box network at the same time.

The scenario we consider of a black box network through which traffic has to be distributed to minimize delay, is representative of many important use cases in networking. A few of these are outlined below:

- A content provider with content replicated in multiple data centers (Figure 2) has to decide what fractions of requested traffic have to be drawn from the different data centers and consequently how this traffic is to be distributed to the different ingresses of the network to which the data centers are connected. To the content provider, the network characteristics are not directly observable and so the load distribution decision has to be based on network characteristics observable from the network edge.
- An ISP has to decide how traffic towards downstream destinations has to be split amongst multiple egresses from its network into downstream networks (Figure 3), i.e., the ISP has to pick the optimal split of traffic to the different ingresses of downstream networks. Since downstream networks may belong to different providers (and hence different Autonomous Systems), the internals of downstream networks are not directly observable. Moreover, Border Gateway Protocol (BGP) does not provide path metric information sufficient for fine-grained performance optimization. Hence, traffic distribution decisions to downstream ISPs have to be based on tomography-based information obtained by probing through the black box downstream networks.
- In Software-Defined WANs (SD-WANs), gateway nodes can be multi-homed and have to decide how to split traffic to different underlay ingresses to minimize delays through the underlay network. This is shown in Figure 4. The underlay is a black box for the SD-WAN nodes and the only information about the underlay available to the SD-WAN nodes is by network tomography
- Network load balancers [10] that distribute incoming demands to a set of distributed servers, as shown in Figure 5, can view the combination of the underlying network and servers as a black box and can optimize the load distribution based on the observed delays in response to past actions.

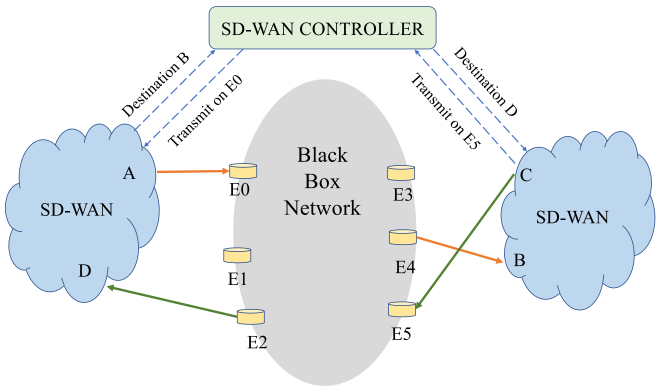


FIGURE 4. Load distribution from SD-WAN gateway nodes to underlay networks.

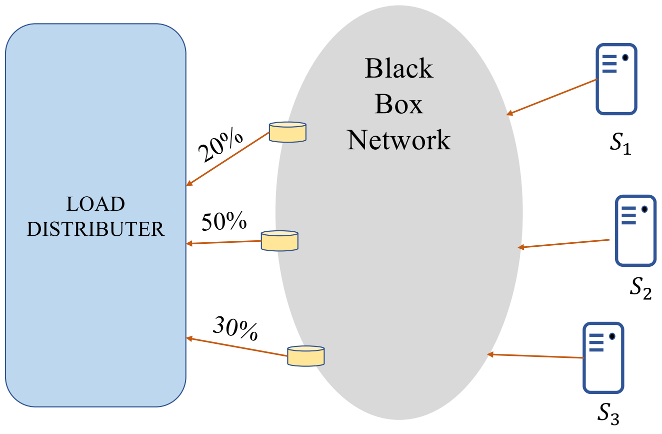


FIGURE 5. Network load balancing over black box networks.

- Segment routing [11] has been proposed for traffic engineering in networks to improve service quality and avoid link congestion. In segment routing, the end-to-end path is composed of segments (as in Figure 7) where the endpoints of the segments are carefully chosen to avoid network congestion. If segment routing is done at the overlay layer, the segments themselves are routed by Interior Gateway Protocol (IGP) picked paths in the underlay. The optimal choice of segment end-points and the optimal split of traffic through different segment routed paths to the same destination will need to be done based on information available at the sources through network tomography.

In this paper, we focus on the scenario that the underlay network is acting as a complete black box. We assume that it is impossible to directly measure the end-to-end delays such a system, only the system average delay is available. In practical systems, such measurements can be obtained by injecting measurement packets into the system. Each measurement packet provides a sample of the delay between a source node and a destination, however, the black box underlay network prevents us from inferring which nodes they are. Only by taking the mean value of a sufficient number of samples could we obtain an approximate measurement of the average delay. We use these measurements in conjunction with past decisions that resulted in the observed delays to make current decisions. The decisions made have to be robust to changing underlying network conditions and be responsive to changes in network topology like link or node failures.

Though tomography based techniques have been used to identify hot spots in networks (See for example, [12]), it is challenging to use it in a machine learning based approach to optimize load distribution. This is due to the fact that the decision space (how to split the traffic) as well as the rewards (tomography measurements) are continuous and high dimensional. This makes it very difficult to use traditional reinforcemnet learning based techniques to solve the problem. However, recent advances in reinforcement learning, especially actor-critic networks, make it possible to implicitly store the actions and rewards in a neural network. Our problem, apart from being continuous and high dimensional, also has constraints on the set of actions. Instead of actor-critic learning, we propose a critic only learning algorithm and use a Frank-Wolfe [13] based technique to enforce the constraints. This approach leads to learning algorithms with rapid convergence, with robustness to topology changes and the large number of nodes.

To our knowledge, the problem of reinforcement learning with strict linear constraints on a continuous action space has not been investigated in the machine learning field. Though previous papers have investigated reinforcement learning or MDP with constraints, in this paper the setting of the problems and constraints is different. MDPs with constraints was covered in [14], however, with a focus on problems with finite action spaces. Achiam *et al.* [15] proposed to use surrogate functions to achieve a worst case constraint-violation bound. Similarly, Tessler *et al.* [16] adopted a penalty signal to guide the agent to satisfy constraints. Le *et al.* [17] focused on batch policy learning, with multiple constraints, however, in their setting, at each time the agent does not have to strictly follow the constraints, their method focus on minimizing a cost function of violating the constraints. A method was proposed in [18] to train agents that satisfy general convex constraints, including safety and diversity constraints. Constraint satisfaction has also been studied in multi-armed bandit problems. In [19] and [20], taking an action will result in a random cost and the agent attempts to satisfy a cost budget constraint. In [21] and [22], the constraint function is unknown and the agent aims at satisfying the constraints according to feedback from the environment. The papers on multi-armed bandit problems all assume unknown constraint functions, which is different from the specific constraints on the action space in our case. In addition, the algorithms in [15]–[22] do not ensure the constraints are strictly satisfied. In this paper, we focus on the problems with continuous action spaces, and the characteristics of our problem require that the given constraints must always be satisfied. This is also the first paper that proposes to use gradient estimates provided by a neural network in a Frank-Wolfe based technique, for the purpose of solving reinforcement learning problems.

Overall, the main contribution of this paper are

- We provide a general model for problems including load distribution, egress picking, load balancing and segment routing.
- The problems can be formulated as a reinforcement learning problem, with constraints on a continuous action space. To the best of our knowledge no prior work has proposed any method to enforce constraints on a continuous action space.
- We propose the method of critic only reinforcement learning and combine it with the Frank-Wolfe method. The overall algorithm achieves better performance compared with the state-of-the-art method DDPG [23], with higher data efficiency.
- The proposed method can be used in a centralized manner, or independently by multiple distributed agents.

II. RELATED WORK

Traffic engineering problems have been extensively investigated [24]–[26]. Several papers focus on managing the elephant flows to achieve better performance. In [27], a method called Hedera is proposed to detect elephant flows using edge switches and assign better paths. Devflow [28] uses wild carded OpenFlow rules and a static multi-path routing algorithm to manage elephant flows. Mahout [29] detects large flows at end hosts and computes the best path or the least congested paths for the flows. MicroTE [30] also utilizes the short term correlation of flows for route assignment. MiceTrap [31] also considers mice flows to improve system scalability. On the other hand, other papers propose to classify the traffic and route the flows according to different requirements [32], [33].

In general, traffic engineering assumes that the network topology, link capacities as well as the estimated point-to-point traffic is known and the objective is to determine how to route the incident traffic in a congestion-free manner. While this is an appropriate model for an ISP that is designing MPLS tunnels or OSPF weights, full knowledge of the topology and routing cannot be assumed for "over-the-top" routing. This is the reason we have to use tomography to implicitly infer the topology and capacities of the opaque network.

Network tomography, as originally proposed in [3], was aimed at estimating the traffic matrix from link measurements [4], [5]. Tomography has evolved to the problem of inferring internal information of a network from end point measurements. A maximum-likelihood estimator for loss rates on internal links based on losses observed by multicast receivers was proposed in [6], [7], [9]. In [8] it was found that sending stripes of probe packets helps increase estimation accuracy of link loss. There has also been recent interest [12] in using large scale end-to-end pings for network diagnostics in very large networks. In [34], performance and capacity aware routing methods were proposed to help large content providers avoid congested edges and improve user experience. A traffic controller received real time traffic and performance measurements to make routing and traffic balancing decisions. Reference [35] investigated the problem of steering large scale traffic at the ISP level. They show that traffic on long-haul links can be reduced by 30 percent if suitable egress points are recommended to a large content provider.

The tomography literature has focused mainly on determining hot spots in networks by making edge-to-edge measurements. In this paper, we extend this idea and use tomography data to actually optimize network performance. Using tomography measurements for optimizing network performance is possible due to recent developments in the machine learning literature. The idea of using reinforcement learning for routing was initiated in [36] before the recent developments in machine learning. More recently, [37] investigated the problem of using machine learning for routing. These machine learning approaches assume that the network topology is known to the learning algorithm and do not deal with optimizing routing over opaque networks. There has been recent work on using machine learning for flow scheduling [38], congestion control [39]–[42] and optimization in video streaming [43]. In addition, several papers also study the application of RL on routing and network performance optimization problems [44]–[46]. In [45], the authors proposed to use RL for path selection in routing problems. In [46], the authors propose to use RL for traffic engineering, however, their method focus on a flow level load distribution and require flow level delay feedback. In this paper, we adopt the RL techniques for optimizing load distribution using tomographic information only, our method only require the minimum information of the demand sizes and average delay.

III. TWO REPRESENTATIVE PROBLEMS

Though the idea of tomography based learning is applicable to several networking scenarios, this paper focuses on two applications, Egress Picking and Traffic Engineering using Segment Routing, to illustrate the applicability of the method. We now describe these two problems in more detail. In order to achieve better performance and scalability, network operators frequently split traffic across different components of a network. This is especially important when the capacity is asymmetric either due to different types of equipment deployed in different parts of the network or due to asymmetric sharing of capacity between multiple users. A common (implicit or explicit) objective when sharing is to minimize the average packet delay. Minimizing average packet delay leads the traffic being split roughly in proportion to the capacity. Traffic splitting is complicated by the fact that different network components are shared between multiple users and the available capacity for a given user varies over time. Both the problems that we study in detail are traffic splitting problems over networks where we do not have visibility into the topology or interfering traffic.

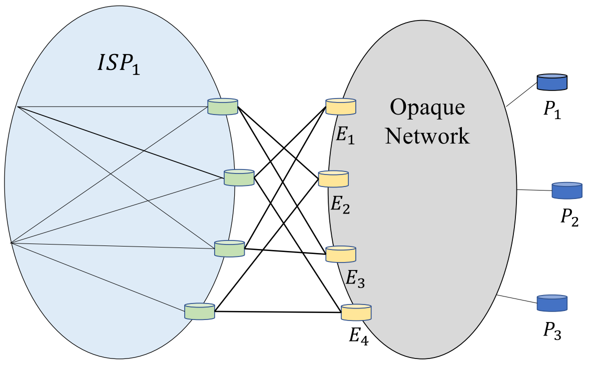


FIGURE 6. An illustration of egress picking.

However, we can use tomography to obtain delay estimates and we use these measurements to guide the traffic splitting process.

A. EGRESS PICKING

Consider an ISP that is routing traffic to destinations downstream through other Autonomous Systems which are opaque to the ISP originating traffic. In general, the originating ISP has multiple choices through which it can transit traffic. The egress picking problem [47] is one where the originating ISP has to determine how to split traffic among the different egress choices in order to efficiently use the capacity downstream. In particular, it is important to ensure that the amount of flow sent on any path does not exceed the capacity of the components on that path, Since the capacities are not observable, it is possible to estimate whether capacities are being violated by measuring the delay. One way of ensuring that capacity is used efficiently is for the egress picking algorithm to minimize the mean delay. Traffic from an ISP is routed using destination prefixes. Each prefix is routed to a destination along an egress point. In this work, we assume that the prefixes can be split across multiple egresses. The splitting is done such that individual flows are routed along the same path in order to avoid out of sequence packets. In general, there can be thousands of prefixes that are routed but typically there are a small subset of prefixes that carry the bulk of the traffic [48]. Therefore, we focus attention on the top few prefixes. We illustrate the egress picking problem in Figure 6, which shows an example with four egress points and three prefixes. Assume that there are n large destination prefixes and m egress point choices. Let t_i denote the amount of traffic for prefix i and let $\boldsymbol{A}_{ij}$ represent the fraction of traffic from prefix i that is routed through egress j. Traffic that is routed to egress j is now forwarded along some path to the destination through the opaque network. Let $D_{ij}(\boldsymbol{A})$ represent the delay from egress j to the destination of prefix i, with the load distribution of $\boldsymbol{A}$. This delay $D_{ij}(\boldsymbol{A})$ is a non-linear function of the traffic on the links of the opaque destination network, which is determined by $\boldsymbol{A}$, assuming all the t_i are constant in a short time frame. The objective is to determine the split of each prefix to minimize the mean delay.

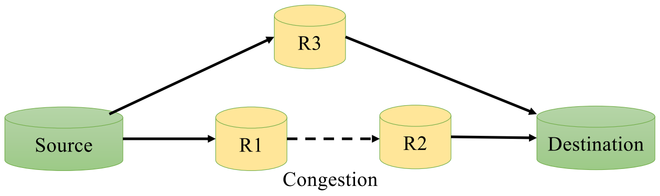


FIGURE 7. An illustration of segment routing for congestion avoidance.

$$\min_{\boldsymbol{A}} \frac{1}{\sum_{i=1}^{n} t_i} \sum_{i=1}^{n} \sum_{j=1}^{m} t_i D_{ij}(\boldsymbol{A}) \boldsymbol{A}_{ij} \tag{1}$$

subject to

$$\sum_{j=1}^{m} \boldsymbol{A}_{ij} = 1, \forall i$$
$$\boldsymbol{A}_{ij} \geq 0, \forall i \ \ \forall j$$

The $\sum_{j=1}^{m} \boldsymbol{A}_{ij} = 1$ constraints are called the simplex constraints. In addition to the simplex and non-negativity constraints there may be additional constraints that may result from policy considerations. For example, there may be upper bounds on the total amount of traffic that can be routed to a particular egress point. If this is the capacity of the egress point then the objective of minimizing mean delay will automatically enforce the constraint, but if it is a policy constraint, then it has to be explicitly enforced by the optimization algorithm. Note that the delay $D_{ij}(\boldsymbol{A})$ is a non-linear function of $\boldsymbol{A}$, and if this function is known (and convex), then we can use projected gradient descent based techniques to solve this problem. Since the network is opaque, we do not know the function $D_{ij}(\boldsymbol{A})$ and therefore we use a tomography based learning algorithm to solve this problem. The tomography module measures delays across the opaque network using probing and this is used as a feedback for the optimization algorithm.

B. TRAFFIC ENGINEERING USING SEGMENT ROUTING

Another application of tomography based learning is traffic engineering using Segment Routing. Segment routing is an IETF protocol for traffic engineering [11]. The key idea of segment routing is to break the route of a flow into several segments. Each segment is a shortest path between the two end points of the segment. Segment information is carried in the packet header and therefore there is no per-flow state maintained in the network. Assume that we have a opaque network where we only have access to a set of edge nodes. We want to route traffic between the edge nodes. Assume that the amount of traffic between edge node i and edge node j is T_{ij}. One option is to directly route from i to j through the opaque network. If the shortest path from i to j is congested then it is possible to segment route the connection from i to some other edge node k and then from k to j. If the set of two shortest paths $i-k$ and $k-j$ are not congested, then this will result in better delay performance. Figure 7 shows an example of using segment routing to avoid a congested link. In this case, the link between router R1 and R2 is congested. We can route along the two segment path $R1-R3$, $R3-R2$ to avoid the congested link. In general, we can route along a path with several segments. In this paper, we restrict the solutions to paths having at most two segments. The techniques developed extend directly to paths having more than two segments. Let $\boldsymbol{B}_{ikj}$ represent the fraction of traffic from i to j that is routed though node k. The fraction of traffic that is routed on a single segment from i to j is represented by $\boldsymbol{B}_{ijj}$. As in the egress picking case, the delay suffered in the opaque network is a non-linear function of the traffic T_{ij}, the traffic splits $\boldsymbol{B}_{ijk}$ as well as any other background traffic carried by the network. Let Δ_{ij} represent the delay on direct path between nodes i and j. We can think of this as the single hop segment delay. The delay $D_{ikj}(\boldsymbol{B})$ incurred on the two segment path $i-k-j$ is $D_{ikj}(\boldsymbol{B}) = \Delta_{ik} + \Delta_{kj}$. Assume that we have n edge nodes in the opaque network and l nodes can be selected as a middle point. We can then write the problem of finding the split values to minimize the average delay as

$$\min_{\boldsymbol{B}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} D_{ikj}(\boldsymbol{B})\boldsymbol{B}_{ikj}T_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} T_{ij}} \tag{2}$$

subject to

$$\sum_{k=1}^{l}\boldsymbol{B}_{ikj} = 1, \forall i\ , \forall j$$
$$\boldsymbol{B}_{ikj} \geq 0, \forall i\ , \forall j\ , \forall k.$$

The constraints that sets the sum of the traffic splits to one are the simplex constraints. The main challenge in solving this problem is the fact that we do not know how the delay varies with the traffic split parameters. As in the egress picking problem, tomography provides end-to-end delay measurements that guides the machine learning based optimization algorithm. We now outline the tomography based learning techniques that we use to solve this problem.

IV. TOMOGRAPHY BASED LEARNING

The only information that we obtain from the opaque network is the tomography measurements. The idea is to use these measurements to determine how to distribute the load. The standard approach to using these measurements is in a learning based algorithm. The most straightforward approach is to use **reinforcement learning** (**RL**), where an agent interacts with an environment in discrete time steps. A general setting for reinforcement learning is shown in Figure 8. At each time step t, the agent observes the state of the environment s_t, takes certain action and receives reward r_t. The common objective for the agent is to maximize the expected cumulative discounted reward $E[\sum_{t=0}^{\infty}\gamma^t r_t]$. In our case, the state is the current traffic demands and the reward corresponds to the tomography inferred delays. The action that we take is to change the traffic split to optimize the objective function. In order to convert the minimization problem to a maximization problem, the reward can be modeled as the negative of the weighed sum of mean delays. The agent maintains a buffer comprising of the received rewards for each state-action pair that has been used thus far. When a new state is observed, the action that results in the maximum reward is chosen in the exploitation mode or a random action is taken in the exploration mode. For small and discrete state and action space, this can be done by using a simple table. However, for large and continuous state and action spaces, it is impossible to store the table directly. This is the case for our problems.

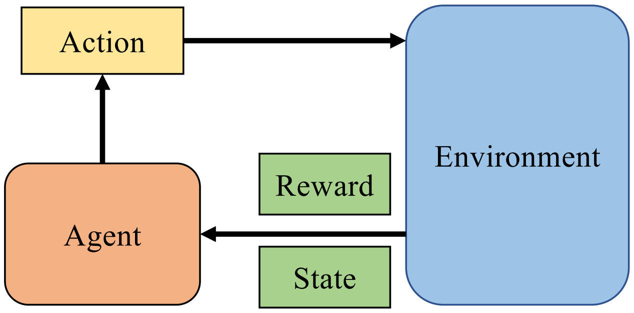


FIGURE 8. A general setting for reinforcement learning.

Since deep neural networks have shown great potential in function approximation, it is possible to replace the state-action-reward table with a deep neural network. This technique, called **Deep Q Network** (**DQN**), has been used for solving problems with a large continuous state space [49]. In the DQN algorithm, a deep neural network is used to estimate the reward for each discrete action under a given state. At each step, the agent chooses the action with the maximum estimated reward. Though DQN shows great potential for solving problems with large state spaces, it can only solve problems with discrete and low-dimensional action spaces. In our case, the action space is the traffic split values, which is continuous. For problems with continuous action space, DQN cannot be directly applied because the action is chosen based on discrete maximization of the estimated reward. Though the continuous action space can be discretized, the number of actions will increase exponentially with the number of degrees of freedom. For a continuous action space, the preferred approach is to use **Deep Deterministic Policy Gradient** (**DDPG**) [23]. DDPG comprises of two neural networks to determine the optimal action. An *actor network* determines the optimal action for a given state and a *critic network* estimates the reward for a given state-action pair. The actor is trained with the policy gradient provided by the critic network. Without special parameter tuning, DDPG has shown promising results on various continuous control problems [23]. However, DDPG is applicable when the action space is unconstrained and our problems have several constraints.

A. ENFORCING CONSTRAINTS

In our case, the action space comprises of traffic splits and is constrained both by the simplex constraint (sum of the splits equals one) as well as non-negativity constraints. In addition, as stated earlier, there may be policy constraints that have to be enforced. Therefore, it will be convenient to use a method where it is easy to enforce constraints on the action space. A technique that is suitable for these types of problems is the **Critic Only Reinforcement Learning** (**CORL**) [50]. Unlike DDPG that uses two neural networks, the critic network for estimating rewards and an actor network for determining the optimal actions, CORL trains only a critic neural network. The action or policy can be derived directly from the critic network by determining the action that minimizes the estimated cost provided by the critic. While solving the optimization problem we can enforce constraints on the solution space. There are two ways to enforce the simplex constraints.

- *Enforcing Constraints Using Softmax:* The standard approach to enforcing simplex constraints in a neural network is to use the softmax function. The softmax function $f(x_i) = e^{x_i} / \sum_1^K e^{x_k}$ is enforced at the output layer of the actor network. This ensures that sum of the probability over all the actions is one. However, the softmax function does not fully cover the entire action space. For example, the softmax function cannot set one of the outputs to one and all other outputs to zero. However, the softmax function is simple to implement in a neural network and works reasonably well in practice to enforce simplex constraints.
- *Enforcing Constraints Using Projection or Frank-Wolfe:* An alternative approach is to start off with a feasible operating point that satisfies all constraints (but may not be optimal). Then the constraints are explicitly enforced by either projecting the gradient or by the projection free Frank-Wolfe method to ensure that the new operating point does not leave the feasible region. We show that by combining CORL with the Frank-Wolfe algorithm, we obtain rapid convergence to the optimal solution. To the best of our knowledge, this is the first use of Frank-Wolfe in CORL.

B. TOMOGRAPHY AND LOAD DISTRIBUTION IN DDPG AND CORL

For DDPG and CORL, the tomography module and load distributor correspond to different parts of the architecture. For both DDPG and CORL, the tomography module is the part of the critic network that measures the effect of a given action. The actor network serves as the load distributor in DDPG. In CORL where there is no actor network, the critic network along with the constraint enforcement module acts as the load distributor. During each interaction with the environment, the critic network estimates the best possible action, gets the corresponding reward from the environment using tomography, and learns from the reward-action pair. By directly serving as both the load distributor and the reward estimator, the critic network is able to probe the black box network more efficiently. According to our results, CORL converges faster and performs better than DDPG in most cases.

V. CRITIC ONLY REINFORCEMENT LEARNING METHODS FOR TRAFFIC SPLITTING

We describe the Critic Only Reinforcement Learning (CORL) algorithm as applied to our problem in more detail. First, we outline CORL where the simplex constraints are enforced using a softmax layer. Since typical neural networks for classification tasks come equipped with the softmax function, this algorithm is easy to implement and performs reasonably well in terms of delay minimization on the topologies tested. Next we briefly outline the Frank-Wolfe algorithm that is used to enforce the constraints. The Frank-Wolfe based approach can be used to enforce linear constraints as long as we can solve a linear optimization problem over these constraints. In the case of the simplex and non-negativity constraints, the linear programming problem is trivial to solve, and this makes the Frank-Wolfe approach extremely attractive.

A. DEFINITION OF BASIC CONCEPTS

We first define the basic components of reinforcement learning in the traffic splitting and segment routing settings. In these problems, the **state** is the original traffic demand vector $\boldsymbol{t}$ in the traffic splitting case, or the traffic demand matrix $\boldsymbol{T}$ in the segment routing scenario. The RL agent has no knowledge of the topology. The **action** is $\boldsymbol{A}$ in traffic splitting or $\boldsymbol{B}$ in segment routing. The **environment** is the network for data transmission. During each **interaction** with the environment, the RL **agent** receives an observation of the state, and takes an action. The action is executed through the environment; finally the agent receives a reward. The **reward** is the negative of the average delay over the system after each interaction, so the agent aims at minimizing the average delay. Since the change of traffic splits is unlikely to impact the volume of traffic demand, there is little correlation between the actions and the change of the environment, so instead of using a total discounted reward, the negative of average delay after each interaction is used as the reward. In addition, in our experiments, the collected traffic traces are replayed, so the state transitions are completely defined by the dataset. The duration of each interaction is determined by the sampling interval of the traffic demands.

B. CRITIC ONLY REINFORCEMENT LEARNING

The CORL algorithm has a critic neural network $Q(\boldsymbol{s}, \boldsymbol{a}|\theta)$, where $\boldsymbol{s}$ is the state, and $\boldsymbol{a}$ is the action. θ is the parameter for the neural network. The critic network estimates the reward for a given pair of state and action. A suitable action given a state can be derived by performing gradient descent with the critic. The CORL method is described in Algorithm 1. Note that for the problem of traffic splitting, the choice of action has no impact on the transition of states, since the traffic demands depend solely on the users. So instead

Algorithm 1 CORL Algorithm

```
1: Randomly initialize critic network $Q(\boldsymbol{s}, \boldsymbol{a}|\theta^Q)$ with weights $\theta^Q$.
2: Initialize target network $Q'$ with weights $\theta^{Q'} \leftarrow \theta^Q$.
3: Initialize replay buffer R.
4: for $t = 0, \dots, T$ do
5:     Collect traffic demand $\boldsymbol{s}_t$ from the system.
6:     Generate one batch of random traffic split vectors $\boldsymbol{v}_0, \dots \boldsymbol{v}_N$.
7:     Enforce the constraint by setting $\boldsymbol{a}_i = f(\boldsymbol{v}_i)$
8:     Set $\boldsymbol{v} = \text{argmin}_{\boldsymbol{v}_i} Q(\boldsymbol{s_t}, f(\boldsymbol{v}_i)|\theta^{Q'})$
9:     for $k = 0, \dots, K$ do
10:        Update $\boldsymbol{v}$: $\boldsymbol{v} \leftarrow \boldsymbol{v} + \gamma \nabla_{\boldsymbol{v}} Q'(\boldsymbol{s}_t, f(\boldsymbol{v})|\theta^{Q'})$.
11:    end for
12:    Execute traffic split $\boldsymbol{a} = f(\boldsymbol{v})$.
13:    Collect information from the black box network and estimate the average delay $c_t$.
14:    Store traffic demand, traffic split and average delay $(\boldsymbol{s}_t, \boldsymbol{a}_t, c_t)$ in R.
15:    Sample a minibatch of $M$ buffer samples $(\boldsymbol{s}_i, \boldsymbol{a}_i, c_i)$ from R.
16:    Update critic by minimizing the loss: $L = \frac{1}{M}\sum_i (c_i - Q(\boldsymbol{s}_i, \boldsymbol{a}_i|\theta^Q))^2$. In this case the critic is also the tomography module and it is capable of emulating the network.
17:    Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$.
18: end for
```

of maximizing the discounted reward, at each time, only the current reward is maximized. Standard CORL is used for maximization. Since our objective is to minimize mean delay, we maximize the negative of the mean delay.

Similar to DDPG [23], we use the "soft" update mechanism for the critic network. A copy of the critic network is created as the target critic network $Q(\boldsymbol{s}, \boldsymbol{a}|\theta)$. The weights of the target network is updated slowly according to the learned critic network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$ with $\tau \ll 1$. At the beginning of the experiment, both networks are initialized with the same random weights. A fixed length replay buffer is used to store past action, state and reward data for training. At the beginning of each time slot, a batch of random vectors that correspond to traffic splits $\boldsymbol{v}_0, \dots, \boldsymbol{v}_N$ is generated. The softmax activation function $f(x_i) = e^{x_i}/\sum_1^K e^{x_k}$ is used on the corresponding elements of $\boldsymbol{v}$ to enforce the simplex and non-negativity constraints. An initial split is selected by picking the vector with the lowest estimated cost. Then the action is optimized for K iterations using the gradient provided by the target critic network. Finally the traffic split is executed and a reward is estimated from the environment. After each interaction with the environment, the original demand, traffic split and average delay is stored in the buffer. A minibatch of data is selected from the buffer to update the critic target by minimizing the MSE of the estimated average delay. At the end of each time slot, the target critic network is "soft" updated. Note that for the problems we consider in this paper, the states are represented by a traffic demand matrix. Since the action taken by the agent has no impact on the traffic demand, so the agent is trying to minimize the average delay at current time. We now outline the Frank-Wolfe algorithm in general and then show how we incorporate it into CORL.

Algorithm 2 Frank-Wolfe

```
1: Pick an arbitrary $\boldsymbol{x_0} \in \boldsymbol{D}$
2: for $k = 0, \dots, K$ do
3:     Compute $z := \text{argmin}_{z\in\boldsymbol{D}} \langle z, \nabla f(\boldsymbol{x}_k)\rangle$
4:     $\boldsymbol{x}_{k+1} = (1-\gamma)\boldsymbol{x}_k + \gamma z, \gamma = \frac{2}{k+2}$
5: end for
```

C. THE FRANK-WOLFE ALGORITHM

The Frank-Wolfe algorithm [13], [51] was proposed to solve convex optimization problems over linear polytopes. The optimization problem that we want to solve is

$$\min_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x}), \tag{3}$$

where $\mathcal{D}$ is a linear polytope. The Frank-Wolfe algorithm starts off at an initial feasible point. If the polytope $\mathcal{D}$ is complicated, then finding a feasible point itself is non-trivial. For our problem, any arbitrary set of traffic splits is feasible. The algorithm iterates through a sequence of feasible points approaching the optimal solution. At each step of the algorithm, the non-linear objective function is linearized at the current feasible point. Next, a linear programming problem is solved with this linear objective function over the polytope $\mathcal{D}$. This solution will be an extreme point of $\mathcal{D}$. We then move along the straight line from the current feasible point to the current optimal extreme point. We can either perform a line search to determine the optimal point to move to or we can use a step length function that guarantees convergence. We use the second approach. A description of the algorithm is given in Algorithm 2. The solution at step k satisfies $f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \le \mathcal{O}(\frac{1}{k})$, where $\boldsymbol{x}^*$ is the optimal solution. In [52], the authors show that even with noisy estimates of the gradient, the Frank-Wolfe based method can achieve a bounded approximation of the optimal solution for several types of linear polytopes. We now show how to incorporate the Frank-Wolfe algorithm into CORL. We call this algorithm CORL-FW.

D. CRITIC ONLY REINFORCEMENT LEARNING WITH FRANK-WOLFE OPTIMIZATION

We now outline CORL-FW, which combines Frank-Wolfe with CORL. In the standard Frank-Wolfe algorithm, the gradient of the non-linear objective function is computed at the current operating point. Since the network is opaque, we do not know the objective function. Therefore, we use the critic network to provide the estimate of the gradient for the Frank-Wolfe method. The linear programming problem

Algorithm 3 CORL-FW Algorithm

```
1: Randomly initialize critic network $Q(\boldsymbol{s}, \boldsymbol{a}|\theta^Q)$ with weights $\theta^Q$.
2: Initialize target network $Q'$ with weights $\theta^{Q'} \leftarrow \theta^Q$.
3: Initialize replay buffer R.
4: for $t = 0, \ldots, T$ do
5:     Collect traffic demand $\boldsymbol{s}_t$ from the system.
6:     Generate one batch of random traffic splits $\boldsymbol{a}_0, \ldots \boldsymbol{a}_N$ from the action space $\mathcal{D}$.
7:     Set $\boldsymbol{a} = \text{argmin}_{\boldsymbol{a}_i} Q(s, \boldsymbol{a}_i|\theta^{Q'})$
8:     for $k = 0, \ldots, K$ do
9:         Compute $\boldsymbol{z} := \text{argmin}_{z \in \mathcal{D}} \langle z, \nabla_{\boldsymbol{a}} Q'(\boldsymbol{s}, \boldsymbol{a}|\theta^{Q'})$
10:        $\boldsymbol{a} \leftarrow (1-\gamma)\boldsymbol{a} + \gamma z, \gamma = \frac{2}{k+2}$
11:    end for
12:    Execute traffic split $\boldsymbol{a}$.
13:    Collect information from the black box network and estimate the average delay $c_t$.
14:    Store original demand, traffic split and average delay $(\boldsymbol{s}_t, \boldsymbol{a}_t, c_t)$ in R.
15:    Sample a minibatch of $M$ buffer samples $(\boldsymbol{s}_i, \boldsymbol{a}_i, c_i)$ from R.
16:    Update critic/tomography module by minimizing the loss: $L = \frac{1}{M}\sum_i (c_i - Q(\boldsymbol{s}_i, \boldsymbol{a}_i|\theta^Q))^2$.
17:    Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$.
18: end for
```

is trivial to solve for both the representative problems. In the case of egress picking, the linear programming problem is separable over the different prefixes and in the traffic engineering problem the linear program is separable over different traffic source-destination pairs. Once the optimal solution is determined, the new traffic splits are computed by using the step length shown in Algorithm 2. Simulation results show that even with estimated gradients CORL-FW can achieve close to optimal solutions. Details of CORL-FW is shown in Algorithm 3.

VI. SIMULATION RESULTS

We test the performance of our methods[1] on real topologies from the Rocketfuel project [53] and Abilene dataset [4]. Note that this explicit knowledge of topologies is used only for evaluation purposes, i.e., to have the ground truth for evaluation purposes. Clearly, the traffic sources do not use this information in their load distribution decisions, since from their perspective, the network to which they are sending traffic is a black box, and the only usable information about the network is that inferred from external probing (network tomography). The delay between source node i and destination node j, $D_{i,j}(\boldsymbol{A})$ or $D_{i,j}(\boldsymbol{B})$, consists of queuing delay and propagation delay on each link along the path. We use a common non-linear model [54] from queuing analysis

$$g(x) = \begin{cases} \frac{w}{1-x/C} + p & \text{If } x < C \\ D + p & \text{If } x \geq C \end{cases} \quad (4)$$

1. Code available at https://github.com/shenghexu/CORL.

to model the delay on a given link. Here $w = 1/\mu$, μ is the service rate, x is the link load, p is the fixed propagation delay, C is the capacity of the link and D is a fixed congestion delay if the utilization of the link gets close to or greater than its capacity. Depending on the scenario, the value of C and w may be calculated differently, In our experiments, we set D to one second, so no link will have a queuing delay greater than one second. When the link load is less than the link capacity, our delay function follows the M/M/1 model which is widely used in delay analysis including the 5G core network [55], congestion control [56] and rate allocation [57]. When the delay is greater than the link capacity, $D + p$ simulates the time-out delay. Again, it is worthwhile to stress that explicit knowledge of these parameters is used only for evaluation purposes. This knowledge is not used for determining optimal load distribution.

To model realistic network traversal, for all the experiments, we assume ECMP is used throughout the black box network. So the overall end to end delay is a weighted sum of the delay from all the paths between the source and destination node.

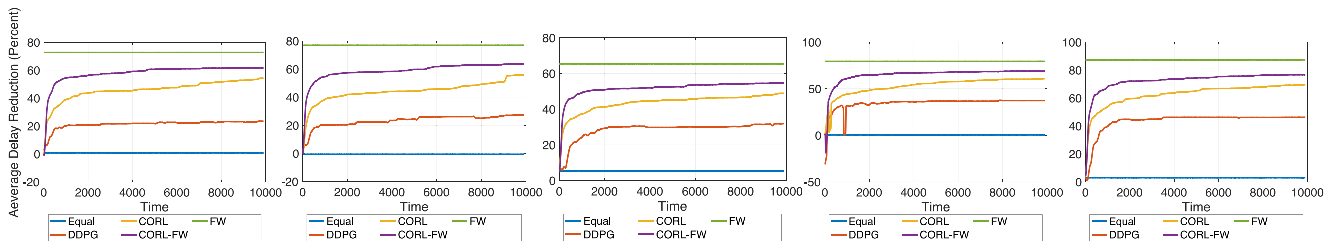
For both sets of topologies, we evaluate the effectiveness of our load distribution scheme for the two representative use cases (of routing through black box networks) discussed in detail earlier. For egress picking problems, all the neural networks for DDPG, CORL and CORL-FW are simple fully connected networks, with two hidden layers of size 256. For the case of segment routing, all the neural networks have two hidden layers of size 512 and 256. The size of the first layer is increased to cover the wider range of delays caused by congestion.

For the DRL methods, to select a suitable initial point, 1000 random initial points are first evaluated with the critic networks, then the best one is selected as the initial point. All the DRL agents maintain a replay buffer with the 1000 most recent states, actions and rewards. After each interaction with the environment, a batch of 32 samples are used for the update of the neural networks. A learning rate of 0.001 is used for all the neural networks. For CORL, we use the Adam optimizer for optimization of the action with a learning rate of 0.05.

For CORL, at each time the optimization is run for 100 iterations. For CORL-FW, we set the stopping criteria to be either reaching 10 steps of optimization or when the Euclidean distance between $\boldsymbol{D}$ and $\boldsymbol{x}_k$ is under 0.00001. To derive a close lower bound, for the Frank-Wolfe (FW) method we assume that optimization is performed with accurate gradients for 100 iterations or until the distance between $\boldsymbol{D}$ and $\boldsymbol{x}_k$ is under 0.00001.

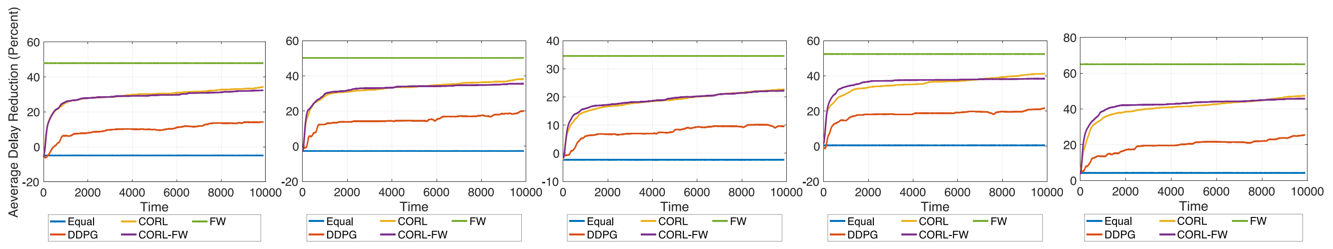
A. RESULTS OF EXPERIMENTS WITH ROCKETFUEL TOPOLOGIES

We first show results comparing the performance of the different load distribution methods when each of five Rocketfuel topologies is used as the topology of the black box network. Details of the five topologies are shown in Table 1.



(a) rf1221 (b) rf1755 (c) rf3257 (d) rf3967 (e) rf6461

FIGURE 9. Egress Picking.



(a) rf1221 (b) rf1755 (c) rf3257 (d) rf3967 (e) rf6461

FIGURE 10. Egress Picking with Four Egress Points.

TABLE 1. Rocketfuel topologies.

Topology	rf1221	rf1755	rf3257	rf3967	rf6461
Number of Nodes	104	87	161	79	138
Number of Links	302	322	656	294	744

Since real traffic matrices (TMs) are not available for these five topologies, we randomly generate TMs using the gravity model [58]. The gravity model assumes demand p_{ij} from node i to node j is,

$$p_{ij} = p_i^{in} p_j^{out} \tag{5}$$

where p_i^{in} and p_j^{out} can be randomly generated from an exponential distribution for each node. To model the correlation across time, we assume that for the duration of the experiment, at each time each p_i^{in} and p_j^{out} is drawn from a Gaussian distribution. In our experiments, we first generate the mean values for each p_i^{in} and p_j^{out}, and scale up the mean values so that the maximum link utilization is over 90 percent (to effectively illustrate the delay impact). Then at each time slot, the p_i^{in} and p_j^{out} are multiplied with factors randomly generated from a Gaussian distribution with mean value of 1 and std of 0.01.

For egress picking, we run ten experiments on each topology. For each run, 20 egress points are randomly selected; these are egresses from the sending ISP and implicitly correspond to the ingresses into the downstream black box network. 20 other nodes in the black box network are randomly selected as destinations – these correspond to true destinations or egresses from the black box network toward the final destinations. Again note that knowledge of these exit nodes from the black box network is for evaluation purposes only. Egress picking is performed by the sending ISP only for 20 egresses from its network. The black box network also has background traffic which we generated from the randomly chosen traffic matrix. Figure 9 shows, for each topology and time instant, the reduction in average delay through the black box network (between its ingress nodes and destination nodes) for ten runs.

For all the five topologies, CORL-FW achieves around 60 percent average delay reduction, with a gap of less than 20 percent from the lower bound. CORL-FW also achieves the highest performance improvement in the least amount of time. CORL achieves around 50 percent average delay reduction. While DDPG also reduces the average delay, it performs worse than the CORL methods. The naive heuristic of equally splitting traffic amongst the egress points hardly improves and sometimes even degrades performance. The results clearly show the effectiveness of using network tomography combined with learning-based decision making. Despite the network being a black box, considerable performance improvement is possible by judicious load distribution using only externally observable information.

To evaluate the performance of the methods on large scale problems, we run another experiment in which four egress points are randomly selected, and all the other nodes are chosen as destinations. In this case there are 100, 83, 157, 75 and 134 prefixes for topology rf1221, rf1755, rf3257, rf3967 and rf6461, respectively. Results are shown in Figure 10. With less egress points to choose from and more prefixes, the performance gains are lower than the previous case. However the CORL methods still perform consistently better than DDPG. With larger problem sizes, the performance of CORL is very close to CORL-FW. This may be because the hidden layers of the NNs is kept at 256, and the capacity of the NNs is bounding the performance of CORL-FW.

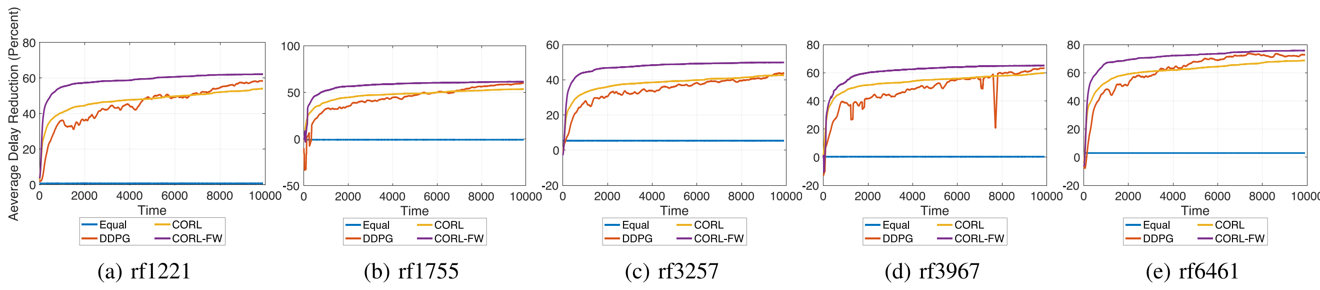


FIGURE 11. **Distributed Egress Picking.**

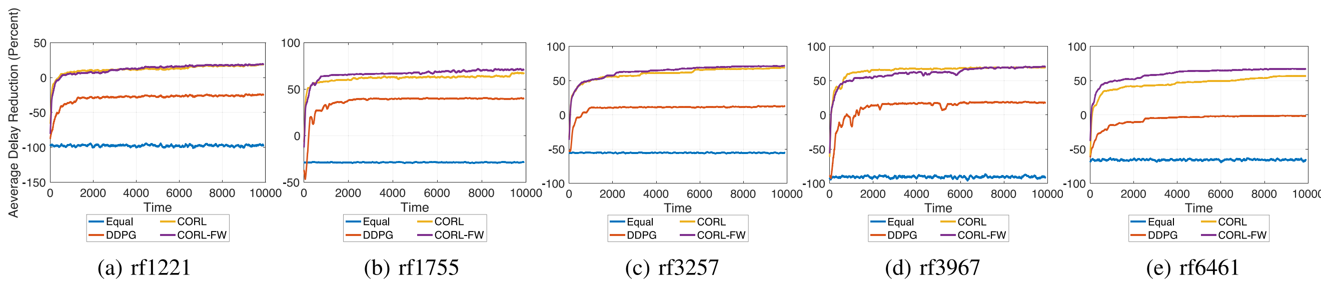


FIGURE 12. **Segement Routing.**

Next, we test the performance when different sending ISPs simultaneously send traffic (using independent egress picking on their networks) to a common black box network. We assume four service providers are sending traffic using optimized egress picking. Each ISP picks 20 egresses for traffic toward 20 prefixes. One exception is for rf3967 where each service provider uses 19 egress nodes and 19 prefixes. Results are shown in Figure 11.

For this distributed egress picking, CORL-FW outperforms all other methods in terms of average delay reduction and efficiency. For DDPG the overall performance actually oscillates with time – this may be caused by the instability of the agents. The DDPG agents possibly were not able to learn the suitable policies to cooperate with others and were competing over the same link resources. This again shows the gains that can be achieved by combining the limited tomography derived information with learning-based decision making, even in a distributed setting with independent decision makers, as would be the case when multiple autonomous systems independently optimize their egress picking to a common opaque network.

For segment routing, after generating the mean values for each p_i^{in} and p_j^{out}, we scale them up so that the maximum link utilization is over 105 percent (to show the need for avoiding highly utilized links). Similar to egress picking, for each topology 10 experiments are performed, with 4 randomly selected source nodes and 16 destinations. For the middle points, we include the 4 source nodes with an option of routing the traffic with no middle point and randomly select 12 other nodes as possible middle nodes. Results are shown in Figure 12. In this case, since the congested link generates a fixed delay providing no gradient, the direct FW approach fails to work.

Since the congested link has to be avoided by spreading the traffic among certain paths, the problem of segment routing is harder than egress picking. For segment routing, the CORL methods achieve over 10 percent average delay reduction on rf1221 and 50 percent on the other four topologies, while DDPG fails to improve performance on four of the topologies.

For segment routing, we also simulate the scenario where four service providers are simultaneously running the RL methods for congestion avoidance. Each service provider uses 4 source nodes, 16 middle points and 16 prefixes.

In the distributed segment routing case, as shown in Figure 13, the CORL methods are still able to improve overall performance across the system, while DDPG and equal splitting fails to work in this case. Note that the DDPG is included only for comparison purposes since it is a widely used method. We do not advocate its use for our purpose. Clearly, traffic engineering using segment routing is much harder with only externally observable information. Nevertheless, we can still use learning-based methods to optimize performance as the experiments show.

B. RESULTS OF EXPERIMENTS ON THE ABILENE DATASET

To further validate the performance of our methods with real traffic demands, we test our methods on the Abilene dataset [4]. The Abilene dataset consists of about 40,0000 measurements of network traffic matrices (TM) on a topology with 12 nodes. The capacities of all the links in this topology are known, however the propagation delays are unknown. We use the known geographical locations to calculate the propagation delays p. We assume that we have control over four of the egress points (or associated ingresses

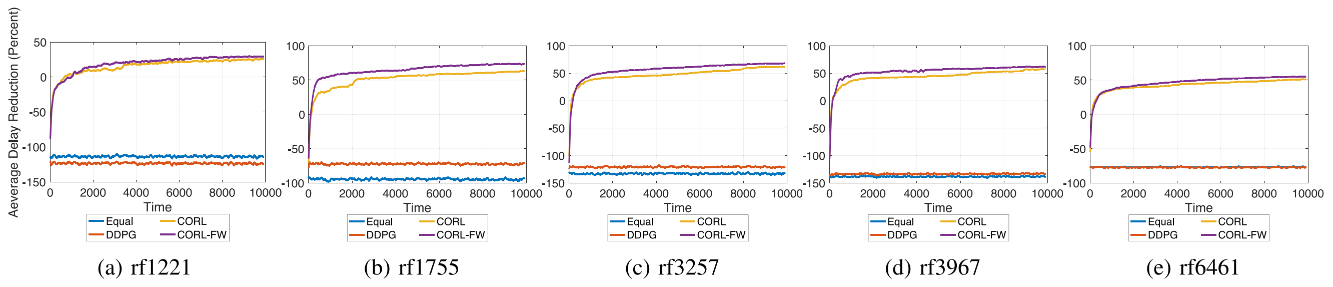


FIGURE 13. Distributed Segment Routing.

into the black box network) in the topology. The destinations includes all the other eight nodes. Each of the TMs is an average of traffic demands over 5 minutes. So the results on the Abilene dataset shows the performance of the methods over a longer range of time, with less frequent opportunities for changing the traffic splits. To simulate a realistic application of our methods, the TMs are replayed in time order. For each TM the learning agent is able to perform one splitting decision and obtain the delays between all four egress points and eight destinations. We ran ten experiments and each time the egress points and prefixes are selected randomly.

As for the Rocketfuel topologies, for comparison, we also test the performance of DDPG [23]. For DDPG, CORL and CORL-FW we use NNs with two hidden layers of size 256. For all methods a replay buffer with size 1000 and learning batch size of 32 is used. Soft update is adopted for better performance stability [59]. Figure 15 shows the performance of the two learning methods on the Abilene dataset. The delays are moving averages of 100 samples. For CORL and CORL-FW each action selection step consists of 100 iterations of gradient descent on the actions space, using the Adam optimizer [60]. It can be seen that the Critic-Only method achieves lower delay with more robust performance compared with DDPG. To show the effectiveness of CORL methods, we also run a direct FW simulation. In this case we assume all the information about the link characteristics and other traffic are known. A step size of $\frac{2}{2+k}$ is used and we set the stopping criteria to be either reaching 100 steps of optimization or the distance between $\boldsymbol{D}$ and $\boldsymbol{x}_k$ is under 0.00001. The FW method converges before 100 iterations for over 90 percent of the cases. So it serves as a very close estimate of the lower bound of the average delay.

C. ANALYSIS OF THE QUALITY OF SOLUTIONS

To better understand the solution qualities of the different methods, we saved the traffic splits from CORL, CORL-FW, DDPG and direct FW for the egress picking simulations in Figure 9. The traffic splits of the last 10 iterations in each topology are used for comparison. Since the FW method achieves the optimum solution, it is used for MSE computation for the traffic splits of the algorithms. DDPG, CORL and CORL-FW achieve MSE of 0.0891, 0.0715 and 0.0689, respectively. The relatively large MSE of DDPG's traffic splits indicates that its solutions are further from the optimum. While CORL and CORL-FW achieve very close MSE values, according to the empirical CDF in Figure 14, CORL almost never gives a traffic split of 1, meaning that it never assigns a demand completely to a single egress point. By contrast for FW, which is the optimal solution, around 5% of the time a demand is completely assigned to a single egress point. This may have been caused by the softmax function used in CORL, which is preventing it from getting close to the optimal solution.

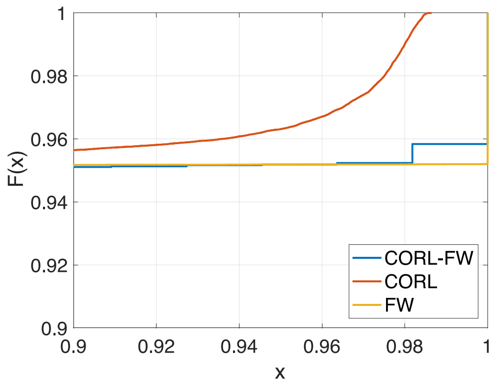


FIGURE 14. Empirical CDF of Traffic Splits.

While CORL-FW displays considerable improvement in Figure 9, it performs similar to CORL in Figure 10 and Figure 12. This is likely because that in the latter cases the action space is much larger, consequently the agents have not found solutions close enough to the optimum, so the difference between a 97% assignment and a 100% assignment does not make much impact on performance.

D. IMPACT OF LINK FAILURES

We study the impact of link failures in the black box network. Though the black box network will have its own restoration mechanisms (such as IP or MPLS fast re-route) to handle link failures, clearly link failures can result in loss of network capacity and consequent delay increases. When external probing shows a delay increase, our decision making algorithms react to mitigate the delay increase. We perform experiments to study how well our algorithms respond to changes in network conditions. To study the effect of link failure, for the first 1000 samples the original topology is

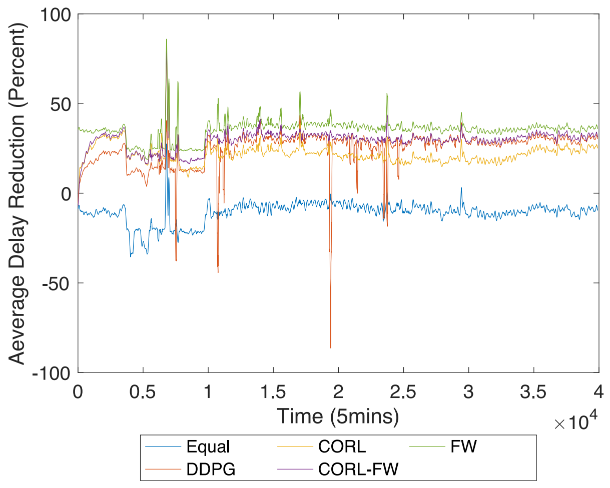


FIGURE 15. Performance Comparison on Abilene Dataset.

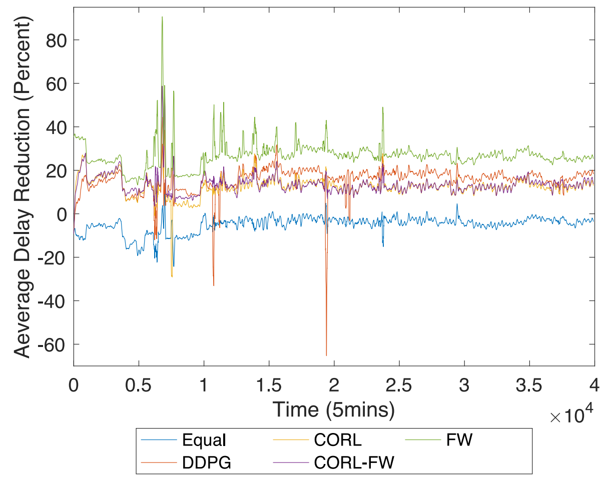


FIGURE 16. Performance Comparison on Abilene Topology with Link Failure.

used. Then after every 1000 samples an impaired topology is used. The topology is generated by randomly dropping one link from the original graph, while still keeping all the nodes connected. Results are shown in Figure 16. In this case CORL-FW achieves a good compromise between close to optimum results and robust performance, showing more stable performance after link failures.

VII. CONCLUDING REMARKS

Network tomography has been extensively studied as a means to infer pertinent network characteristics from external observations. With the growing use of network virtualization and deployment of overlay technologies such as SD-WANs, the use of network tomography by overlay networks to infer characteristics of the underlay networks is likely to grow. In this paper, we considered the use of network tomography in a broader context – can we combine network tomography with machine learning based decision making for automated performance optimization from the periphery of the network. We considered this problem in a general setting where an external entity that is generating traffic (such as an over the top service provider) to a set of its clients (destinations) has to send the traffic over a black box network (such as an underlay) while minimizing average delay. Using learning-based approaches for this problem poses several technical challenges including the dimension of the solution space and the enforcement of constraints that arise naturally in the networking context (such as policy constraints, capacity constraints, etc.). We show how these constraints can be handled while using a deep reinforcement learning framework for decision making. For two representative problems (egress traffic picking and optimized segment routing) we show simulation results on two widely used network topology databases. The methods we use can be used both in a centralized manner and distributedly by multiple independent agents. The effectiveness of our method is illustrated by delay reductions of as much as 60% in comparison to standard heuristics. However, the assumptions made in the simulation environment may limit the accuracy of the simulation results. The performance of the methods could be further verified in testbed or real systems. We also believe that the methods proposed in this paper have many other applications. The methods could also be used to maximize throughput or minimized jitter in the load distribution problems. They could also be used for the problem of power allocation in wireless networks [61], which can be formulated as a RL problem with a constrained continuous action space.

REFERENCES

[1] R. Albert, A. Patney, D. Luebke, and J. Kim, “Latency requirements for foveated rendering in virtual reality,” *ACM Trans. Appl. Percept.*, vol. 14, no. 4, p. 25, 2017.
[2] M. Claypool and K. Claypool, “Latency can kill: Precision and deadline in online games,” in *Proc. ACM 1st Annu. SIGMM Conf. Multimedia Syst.*, 2010, pp. 215–222.
[3] Y. Vardi, “Network tomography: Estimating source-destination traffic intensities from link data,” *J. Amer. Stat. Assoc.*, vol. 91, no. 433, pp. 365–377, 1996.
[4] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, “Fast accurate computation of large-scale ip traffic matrices from link loads,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 206–217, 2003.
[5] A. Soule, A. Nucci, R. L. Cruz, E. Leonardi, and N. Taft, “Estimating dynamic traffic matrices by using viable routing changes,” *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 485–498, Jun. 2007.
[6] R. Cáceres, N. G. Duffield, J. Horowitz, and D. F. Towsley, “Multicast-based inference of network-internal loss characteristics,” *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2462–2480, Nov. 1999.
[7] Y. Chen, D. Bindel, and R. H. Katz, “Tomography-based overlay network monitoring,” in *Proc. 3rd ACM SIGCOMM Conf. Internet Meas.*, 2003, pp. 216–231.
[8] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley, “Network loss tomography using striped unicast probes,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 697–710, Aug. 2006.
[9] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, “Network tomography on general topologies,” *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 1, pp. 21–30, 2002.
[10] N. Handigol, M. Flajslik, S. Seetharaman, N. McKeown, and R. Johari, “Aster*X: Load-balancing as a network primitive,” in *Proc. 9th GENI Eng. Conf. (Plenary)*, 2010, pp. 1–2.
[11] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, “The segment routing architecture,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2015, pp. 1–6.

[12] C. Guo *et al.*, "PINGMESH: A large-scale system for data center network latency measurement and analysis," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 139–152.

[13] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logist. Quart.*, vol. 3, nos. 1–2, pp. 95–110, 1956.

[14] E. Altman, *Constrained Markov Decision Processes*, vol. 7. Hoboken, NJ, USA: CRC Press, 1999.

[15] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 22–31.

[16] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–9.

[17] H. M. Le, C. Voloshin, and Y. Yue, "Batch policy learning under constraints," in *Proc. Mach. Learn. Res.*, vol. 97, 2019, pp. 3703–3712.

[18] S. Miryoosefi, K. Brantley, H. Daume, III, M. Dudik, and R. E. Schapire, "Reinforcement learning with convex constraints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14093–14102.

[19] W. Ding, T. Qin, X.-D. Zhang, and T.-Y. Liu, "Multi-armed bandit with budget constraint and variable costs," in *Proc. AAAI Conf. Artif. Intell.*, vol. 27, 2013, p. 1.

[20] A. Burnetas, O. Kanavetas, and M. N. Katehakis, "Asymptotically optimal multi-armed bandit policies under a cost constraint," *Prob. Eng. Inf. Sci.*, vol. 31, no. 3, pp. 284–310, 2017.

[21] X. Cao and K. J. R. Liu, "Online convex optimization with time-varying constraints and bandit feedback," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2665–2680, Jul. 2019.

[22] X. Yi, X. Li, L. Xie, and K. H. Johansson, "Distributed online convex optimization with time-varying coupled inequality constraints," *IEEE Trans. Signal Process.*, vol. 68, pp. 731–746, Jan. 2020.

[23] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2016, pp. 1–6.

[24] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for ip networks," *IEEE Netw.*, vol. 14, no. 2, pp. 11–19, Mar./Apr. 2000.

[25] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 118–124, Oct. 2002.

[26] M. R. Abbasi, A. Guleria, and M. S. Devi, "Traffic engineering in software defined networks: A survey," *J. Telecommun. Inf. Technol.*, vol. 14, no. 2, pp. 28–33, 2000.

[27] M. Al-Fares *et al.*, "HEDERA: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, vol. 10. San Jose, CA, USA, 2010, pp. 89–92.

[28] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 254–265.

[29] A. R. Curtis, W. Kim, and P. Yalagandula, "MAHOUT: Low-overhead datacenter traffic management using end-host-based elephant detection," in *Proc. IEEE INFOCOM*, 2011, pp. 1629–1637.

[30] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proc. 7th Conf. Emerg. Netw. Exp. Technol.*, 2011, pp. 1–12.

[31] R. Trestian, G.-M. Muntean, and K. Katrinis, "MiceTrap: Scalable traffic engineering of datacenter mice flows using openflow," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, 2013, pp. 904–907.

[32] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir, "Application-awareness in SDN," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 487–488.

[33] H. Farhadi and A. Nakao, "Rethinking flow classification in SDN," in *Proc. IEEE Int. Conf. Cloud Eng.*, 2014, pp. 598–603.

[34] B. Schlinker *et al.*, "Engineering egress with edge fabric: Steering oceans of content to the world," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 418–431.

[35] E. Pujol, I. Poese, J. Zerwas, G. Smaragdakis, and A. Feldmann, "Steering hyper-giants' traffic at scale," in *Proc. 15th Int. Conf. Emerg. Netw. Exp. Technol.*, 2019, pp. 82–95.

[36] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 671–678.

[37] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proc. 16th ACM Workshop Hot Topics Netw.*, 2017, pp. 185–191.

[38] L. Chen, J. Lingys, K. Chen, and F. Liu, "AUTO: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2018, pp. 191–205.

[39] K. Winstein and H. Balakrishnan, "TCP EX machina: Computer-generated congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 123–134, 2013.

[40] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "QTCP: Adaptive congestion control with reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 445–458, Jul.–Sep. 2018.

[41] K. Xiao, S. Mao, and J. K. Tugnait, "TCP-Drinc: Smart congestion control based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 11892–11904, 2019.

[42] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on Internet congestion control," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3050–3059.

[43] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 197–210.

[44] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[45] G. Stampa, M. Arias, D. Sánchez-Charles, V. Muntés-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," 2017. [Online]. Available: arXiv:1709.07080.

[46] Z. Xu *et al.*, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 1871–1879.

[47] N. Kang, M. Ghobadi, J. Reumann, A. Shraer, and J. Rexford, "Efficient traffic splitting on commodity switches," in *Proc. 11th ACM Conf. Emerg. Netw. Exp. Technol.*, 2015, p. 6.

[48] E. Kohler, J. Li, V. Paxson, and S. Shenker, "Observed structure of addresses in IP traffic," in *Proc. 2nd ACM SIGCOMM Workshop Internet Meas.*, 2002, pp. 253–266.

[49] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[50] E. Alibekov, J. Kubalík, and R. Babuška, "Policy derivation methods for critic-only reinforcement learning in continuous spaces," *Eng. Appl. Artif. Intell.*, vol. 69, pp. 178–187, Mar. 2018.

[51] M. Jaggi, "Revisiting Frank–Wolfe: Projection-free sparse convex optimization," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 427–435.

[52] A. Mokhtari, H. Hassani, and A. Karbasi, "Stochastic conditional gradient methods: From convex minimization to submodular maximization," *J. Mach. Learn. Res.*, vol. 21, no. 105, pp. 1–49, 2020.

[53] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.

[54] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Networks*, vol. 2. Upper Saddle River, NJ, USA: Prentice-Hall Int., 1992.

[55] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded VNF chains in 5G core networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 692–704, Feb. 2019.

[56] D. Wang and Y. Song, "ECCO: A novel end-to-end congestion control scheme in multi-hop cognitive radio ad hoc networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 1, pp. 93–102, Mar. 2018.

[57] M. H. Hajiesmaili, M. S. Talebi, and A. Khonsari, "Multiperiod network rate allocation with end-to-end delay constraints," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1087–1097, Sep. 2017.

[58] M. Roughan, "Simplifying the synthesis of Internet traffic matrices," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 5, pp. 93–96, 2005.

[59] R. Fox, A. Pakman, and N. Tishby, "Taming the noise in reinforcement learning via soft updates," in *Proc. 32nd Conf. Uncertainty Artif. Intell. (UAI)*, Jun. 2016, pp. 1–8. [Online]. Available: http://auai.org/uai2016/proceedings/papers/219.pdf

[60] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, May 2015, pp. 1–6. [Online]. Available: http://arxiv.org/abs/1412.6980

[61] S. Xu, P. Liu, R. Wang, and S. S. Panwar, "Realtime scheduling and power allocation using deep neural networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2019, pp. 1–5.

SHENGHE XU (Member, IEEE) received the B.Sc. degree in information engineering from Xi'an Jiaotong University in 2015, and the Ph.D. degree in electrical engineering from New York University Tandon School of Engineering in 2020. He worked as a summer research intern with Cisco Systems, San Jose, CA, USA, in 2017. In 2019, he worked as a summer research intern with Nokia Bell Labs, Holmdel, NJ, USA. His research interests include wireless communications, computer networks, and machine learning.

MURALI KODIALAM (Member, IEEE) received the Ph.D. degree from the Massachusetts Institute of Technology. In 1992, he joined Bell Laboratories, where he is currently a Bell Labs Fellow. He has authored over 100 refereed conference and journal publications, and holds over 40 patents. His general research interests are in resource allocation in communication networks. He has worked on IP routing, switch scheduling and data structures and algorithms for fast packet processing in wired and wireless networks. He was a co-recipient of the 2008 IEEE Leonard Abraham Prize, the 2011 IEEE Bennett Prize, and the IEEE INFOCOM 2014 best paper awards. He served as an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING.

T. V. LAKSHMAN (Fellow, IEEE) received the master's degree in physics from the Indian Institute of Science, Bengaluru, India, in 1984, and the Ph.D. degree in computer science from the University of Maryland, College Park, in 1986. He is currently the Head of the Networks Research Group, Nokia Bell Laboratories. His research contributions span a spectrum of networking topics, including switch architectures, network design, TCP performance, traffic management, and software-defined networking. He was a recipient of several IEEE and ACM awards, including the IEEE Leonard Abraham Prize, the IEEE Communication Society William R. Bennett Prize, the IEEE INFOCOM Achievement Award, the IEEE Fred W. Ellersick Prize Paper Award, and the ACM SIGMETRICS Best Paper Award. He also received the 2010 Thomas Alva Edison Patent Award from the R&D Council of New Jersey. He has been an Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING and the IEEE TRANSACTIONS ON MOBILE COMPUTING. He is a Fellow of Bell Laboratories and ACM.

SHIVENDRA S. PANWAR (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, in 1986. He is a Professor with the Electrical and Computer Engineering Department, NYU Tandon School of Engineering. He is the Director of the New York State Center for Advanced Technology in Telecommunications (CATT), the Faculty Director and a Co-Founder of the New York City Media Lab, and a member of NYU Wireless. His research interests include the performance analysis and design of networks. His current research focused on cross-layer research issues in wireless networks, and multimedia transport over networks. He has coauthored a textbook: *TCP/IP Essentials: A Lab Based Approach* (Cambridge University Press). He was a winner of the IEEE Communication Society's Leonard Abraham Prize for 2004, the ICC Best Paper Award in 2016, and the Sony Research Award. He was also co-awarded the Best Paper in 2011 Multimedia Communications Award. He has served as the Secretary of the Technical Affairs Council of the IEEE Communications Society.