OXFORD

The Computer Journal, 2024, **67**, 2995–3008
https://doi.org/10.1093/comjnl/bxae064
Advance access publication date 21 July 2024
Original Article

DRL-Tomo: a deep reinforcement learning-based approach to augmented data generation for network tomography

Changsheng Hou, Bingnan Hou*, Xionglve Li, Tongqing Zhou, Yingwen Chen, Zhiping Cai*

College of Computer, National University of Defense Technology, No. 109 Deya Road, Kaifu District, Changsha City, Hunan Province 410073, China

*Corresponding author: Department of Computational Science, College of Computer, National University of Defense Technology, No. 109 Deya Road, Kaifu District, Changsha City, Hunan Province 410073, China. E-mail: houbingnan19@nudt.edu.cn, zpcai@nudt.edu.cn

Abstract

Accurate and current comprehension of network status is crucial for efficient network management. Nevertheless, direct network measurement strategies entail substantial traffic overhead and demand intricate coordination among network entities, making them impractical. Network tomography, an indirect measurement approach, utilizes insights garnered from measured parts to deduce characteristics of the entire network. Past studies frequently depend on acquiring challenging-to-access information, such as the complete network topology or support from specialized protocols. Unfortunately, these constraints pose challenges in non-cooperative scenarios where obtaining such information is difficult. Recent endeavors pursue emancipating tomography from dependence on copious information, striving to predict unmeasured path performance using limited data. Nevertheless, the disparity between the measured data and actual performance has hindered the accuracy. In response, we introduce an innovative tomography framework named DRL-Tomo, designed to alleviate potential biases. DRL-Tomo initiates by generating augmented data through deep reinforcement learning, gradually approximating the genuine performance of unmeasured paths. Subsequently, a neural network model is trained using this augmented data, enabling precise inferences. Our experiments, encompassing both real-world and synthetic datasets, vividly demonstrate DRL-Tomo's remarkable enhancement. Specifically, it achieves a substantial 10%–67% improvement in path delay prediction and an impressive 30%–98% enhancement in path loss rate prediction.

Keywords: Tomography; Deep Reinforcement Learning; Augmented Data Generation

1. INTRODUCTION

Accurate and timely comprehension of network conditions, such as delay and congestion, is crucial for various network functionalities, including route selection, resource allocation, fault diagnosis, and service migrations [1, 2]. However, actively or passively measuring the network incurs impracticable traffic, computational, and communication overhead [3, 4]. Despite the current network infrastructure supporting larger volumes of data transmission and faster transmission rates, the large-scale, heterogeneous nature of networks and their multiple ownerships still hinder the widespread application of direct network measurement [5].

Network tomography [6, 7] is a widely explored paradigm that relies on limited measurements of end-to-end path performance to deduce concealed network states. However, most existing network tomography methods require knowledge of network topology and are often restricted to specific topologies or routing protocols [8, 9], making them challenging to deploy in practice. Ma *et al*. proposed NeuTomography [3], which utilizes deep neural networks to infer the performance of unmeasured paths in the network. However, its effectiveness is limited by insufficient measurement data, leading to compromised estimation accuracy.

In this work, we aim to generate high-quality augmented data to train the neural network, thereby improving the accuracy of network performance estimation. We introduce an innovative tomography method named DRL-Tomo, which distinguishes itself by operating without exhaustive knowledge of network intricacies and has the potential to rectify biases present in limited measurement data.

DRL-Tomo employs a deep reinforcement learning methodology to predict the performance of unmeasured end-to-end paths using a small subset of measurable paths. The key idea is to rely on deep reinforcement learning to generate augmented data for training neural networks. Unlike NeuTomography, which uses an unmeasured path set with an estimated performance upper bounds to generate augmented data, our method generates augmented data that progressively align with real-world performance, thereby enhancing accuracy.

Delving into the methodology, DRL-Tomo is organized into three pivotal stages. Firstly, it establishes performance upper bounds for unmeasured paths by leveraging a small pool of measured data. Subsequently, it employs a deep reinforcement learning algorithm to generate augmented data. Finally, DRL-Tomo integrates the initial measured data with the augmented data to create a hybrid training dataset for training the neural network model, leading to substantial improvements in estimation accuracy.

The effectiveness of DRL-Tomo is showcased through real-world and simulation experiments. It outperforms existing

Received: December 11, 2023. **Revised:** June 15, 2024. **Accepted:** July 4, 2024

state-of-the-art solutions in accurately predicting path performance, demonstrating remarkable enhancements ranging from 10% to 67% in path delay reduction and 30% to 98% in path loss rate reduction. DRL-Tomo is expected to inspire further research on network performance inference in non-collaborative scenarios. The source code for DRL-Tomo is publicly accessible at https://github.com/houchangsheng/DRL-Tomo.

To summarize, we make the following contributions:

1) **Innovative Tomography Framework**: We introduce a lightweight yet robust tomography framework leveraging the deep neural network. This framework excels in network monitoring tasks, utilizing merely a subset of the complete end-to-end path performance data. Remarkably, even with access to only 20% to 30% of path performance data, DRL-Tomo adeptly predicts the performance of the entire network.
2) **Augmented Data Generation with DRL**: Our work presents a novel approach for generating augmented data through deep reinforcement learning. This method facilitates the rapid creation of data that closely emulate real-world path performance, significantly enhancing the precision of performance predictions.
3) **Comprehensive Evaluation**: We rigorously evaluate DRL-Tomo's capabilities using both real-world and synthetic datasets. Our findings underscore the framework's superiority over current state-of-the-art solutions. Notably, DRL-Tomo achieves substantial performance improvements, ranging from 10% to 67% in path delay prediction and 30% to 98% in path loss rate prediction.

The rest of the paper is organized as follows: Section 2 discusses the background and motivation of this work. Section 3 reviews the related work. Section 4 gives the problem statement. Section 5 and Section 6 describe the DRL-Tomo framework and the augmented data generation method based on deep reinforcement learning. Section 7 evaluates DRL-Tomo's performance, Section 8 discusses its weaknesses, and Section 9 concludes this work.

2. BACKGROUND AND MOTIVATION

Early network tomography approaches perform end-to-end network measurements and infer link characteristics, usually using statistical inference or algebraic calculations. *Prevailing efforts in these methods demand an extensive grasp of network information, making this requirement challenging to implement in practical settings.* For instance, endeavors involving the inference of network link performance [8, 10] and the identification of failure links [9, 11] necessitate meticulous familiarity with network topology. Meanwhile, algorithms aimed at ascertaining network topology [12] presuppose the network adheres to a tree-like structure, an assumption that proves problematic given the intricate nature of real-world network deployments. Moreover, certain investigations [9, 12–16] require specific protocol support, such as source routing or multicast.

Recent tomography techniques utilize machine learning algorithms, such as neural networks and reinforcement learning, to infer the network's internal state [17–23], detect anomalous links, select monitoring positions, or construct measurement paths [2, 5, 14, 15, 24–26]. *However, these techniques still cannot eliminate their dependence on network topology knowledge. Some even require precise control of measurement paths, which impacts their applicability.*

Ma *et al.* [3] were the first to utilize deep neural networks to infer the performance of unmeasured paths based solely on path measurement results without requiring knowledge of network topology. *However, its effectiveness is restricted due to insufficient measurement data, leading to compromised estimation accuracy.*

All these limitations motivate us to enhance the accuracy of path performance prediction in scenarios where only rudimentary path performance information is available. In this work, we employ deep reinforcement learning algorithms to generate high-quality augmented data that closely approximate the true performance of unmeasured paths. These augmented data, combined with the raw measured data, are used to iteratively train a neural network model. This process helps rectify the bias caused by insufficient measurement data, thereby improving prediction accuracy.

3. RELATED WORK

The network tomography technique has evolved over time. Initially, tomography primarily relied on statistical learning or algebraic methods to estimate link performance, categorizing these methods as traditional tomography approaches. However, recent advancements have leveraged machine learning, including deep learning algorithms, to improve prediction accuracy. We categorize these newer methodologies as machine learning-based tomography methods. Table 1 provides a comprehensive overview of related studies.

3.1. Traditional tomography methods

Ren *et al.* [27] proposed a monitor deployment algorithm to maximize the number of identifiable links. Gao *et al.* [28] introduced Scalpel, while Dong *et al.* [29] presented OMA, both aiming to infer target link metrics through graph trimming and optimized monitor placement techniques. Yang *et al.* [30] delved into identifiability and monitor placement algorithms for target paths. Feng *et al.* [14, 31, 32] proposed algorithms to infer the tightest error bounds for unidentifiable links, explored monitor placement, and developed measurement path construction methods. Li *et al.* [13] focused on deriving the tightest upper and lower bounds for target links. He *et al.* [34] and Li *et al.* [33, 35] investigated link metric inference and monitor deployment in dynamic networks where the topology is dynamically changing.

These traditional approaches often necessitate precise topology knowledge, routing matrices, or controlled probe paths. However, network topology is frequently unobservable, and networks often restrict support for specific routing protocols due to security considerations, rendering the application of these algorithms challenging in practice.

3.2. Machine learning-based tomography methods

Rahali *et al.* [17] proposed TOM, which utilizes neural networks to infer the state of underlying resources based on measurement results in the overlay network. They modeled the inference task as a regression problem, training the neural network to learn the relationship between path measurement results and link metrics. Rkhami *et al.* [18] improved TOM by introducing a transfer learning-based solution to address topology changes. Ushizuka *et al.* [21] used a training set consisting of multiple patterns of individual link statuses and end-to-end measurement data from a simulated network to estimate the internal state of a network without path information. Sartzetakis *et al.* [22, 23] proposed a machine learning formulation for network tomography with partially known network topology and dynamic routing. They incorporated both link and node features to train neural networks to infer the performance of unmonitored paths. Ibraheem *et al.* [19,

Table 1. Comparison between the recent efforts and the proposed.

Methods			Unknown Topology	Unspecific Topology	Uncontrolled Probe Paths	Unlimited Monitor Position	No Self-loop Routing	Unlimited Failure Links/Nodes	Using Augmented Data
Category	**Key Technique**	**Description**							
Traditional Tomography	Algebraic Computation	Topological Structure Identification [27]	×	✓	×	✓/×	×	×	—
		Target Link Performance Inference [28–30]	×	✓	×	✓/×	✓	✓	—
		Link Metric's Error Bound [13, 14, 31, 32]	×	✓	×	✓/×	✓	✓	—
		Tomography in Dynamic Networks [33–35]	×	✓	×	×	✓/×	✓	—
Machine Learning-based Tomography	Neural Network	Network State Inference in Network Slicing [17, 18]	×	✓	✓	✓	—	—	×
		Network State Inference under Dynamic Routing [21–23]	×	✓	✓	✓	—	—	×
		In-Vehicle Network Delay Inference [19, 20]	×	✓	✓	✓	—	—	×
		NeuTomography [3]	✓	✓	✓	✓	—	—	✓
	Reinforcement Learning	Congestion Link Identification [24]	×	×	✓	✓	✓	✓	—
		Traffic Measurement Optimization [25]	×	✓	✓	×	—	—	—
		Measurement Path Construction [14, 15]	×	✓	×	✓	✓	—	—
		Probing Path Selection [2, 5]	×	✓	×	✓	—	—	—
		Traffic Load Optimization [36]	✓	✓	✓	✓	—	—	×
		The proposed	✓	✓	✓	✓	—	—	✓

—: The methods are not related to this item.

20] proposed NNDT, which utilizes deep neural networks to infer delays in in-vehicle networks by estimating the delays of internal links and path segments as much as possible.

Recently, reinforcement learning has been widely used in tomography methods. Pan *et al.* [24] introduced RIC, which employs Q-learning to identify congested links. Nie *et al.* [25] proposed a traffic measurement optimization method using Q-learning. Feng *et al.* developed GSPC, a reinforcement learning-based method for constructing measurement paths. Tao *et al.* [2, 5] proposed Subito, which utilizes reinforcement learning to find effective probe paths. Xu *et al.* [36] also used deep reinforcement learning to optimize traffic load distribution.

These machine learning-based tomography approaches require a comprehensive understanding of network topology, with some necessitating additional assumptions, such as specific topology constraints and controllable monitor nodes or probe paths. Ma *et al.* [3] introduced NeuTomography, a versatile approach that does not require any assumptions about the network. However, its accuracy is diminished by the inaccurately generated augmented data.

4. PROBLEM STATEMENT

4.1. System model

Network tomography problems can be described as

$$Y = RX, \tag{1}$$

where Y is the path performance metric vector obtained by end-to-end measurements, X is the link performance metric vector, and R is the routing matrix with each entry $R_{i,j}$ representing that link l_j is present on the path p_i. The routing matrix reveals the relationship between the measured path performance metrics and the network's internal link metrics. Usually, R and X are unknown, then Equ. (1) corresponds to an underdetermined system of equations, and X will also have infinitely many sets of estimated solutions $\hat{X}$.

Network tomography is essentially a mapping problem from low-dimensional space to high-dimensional space. Solving this kind of underdetermined inverse problem involves eliminating or reducing the uncertainty in the mapping process by finding additional information or constraints. This approach can make Equ. (1) yield a unique solution or a smaller range of solution space. Let $R^{'}$ represent a part of the routing matrix, indicating the relationship between unmeasured paths and internal links. Most previous work has assumed that R is known, utilizing the measured paths to infer the metric vector X of the internal links. Combined with the known $R^{'}$, the performance of the unmeasured paths can be calculated, i.e. $R^{'}X$. In contrast, our problem setting is more loose. Without any assumptions about R or X, the goal is to directly determine the value of $R^{'}X$ using the given measured paths.

4.2. Path measurements

For a network $\mathcal{G}$ with n nodes, any two nodes can form a node pair; thus, there are $\binom{n}{2}$ node pairs. Let V denote the node set in $\mathcal{G}$ ($|V| = n$), and let (v_i, v_j) $(v_i, v_j \in V, v_i \neq v_j)$ denote a node pair. Then $T = \{(v_i, v_j)\}_{v_i, v_j \in V, v_i \neq v_j}$ is the set of node pairs containing all nodes in V (i.e. $\binom{|T|=}{n2}$). Assuming that we have obtained the performance metrics of measured end-to-end paths, which are associated with node pairs in a subset S of T, i.e. $S \subset T$, our goal is to infer

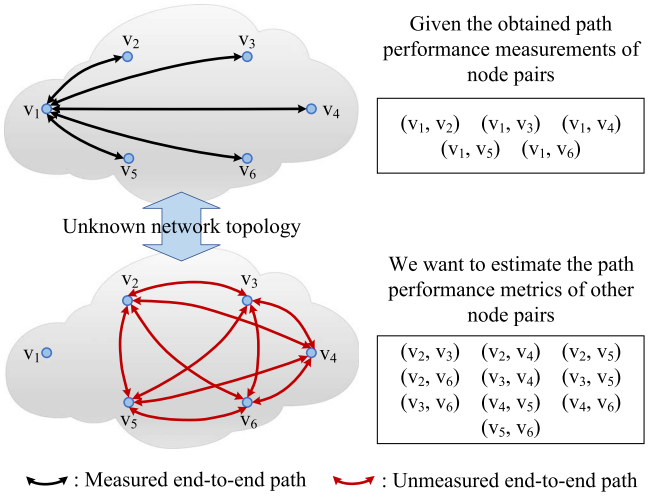


Figure 1. Example of network tomography.

end-to-end path performance metrics for all unmeasured node pairs, i.e. node pairs in the set $T \backslash S$.

To make the problem more applicable to real networks, we do not make additional assumptions about the network and limit the measurement paths. Specifically,

1) We do not know the topology of the network.
2) For any node pair $(v_i, v_j) \in S$, we only know the performance metric of the end-to-end path between v_i and v_j, and do not know how many nodes or which nodes are on that path.
3) The path between node pairs (v_i, v_j) in S is constructed through the unknown underlying routing protocol.
4) We do not restrict which specific paths the set P_S must contain. While the monitor nodes are determined, we obtain all measurable paths as a set P_S. Here, monitor nodes are randomly sampled nodes as probes, and from the selected monitor nodes, we can probe the path performance to all other nodes. Since we do not restrict how monitor nodes are selected, we do not restrict the acquisition of the set P_S.
5) S covers all nodes in the network. The method, which selects monitor nodes first and then probes the performance of the paths from the monitor nodes to all other nodes, potentially indicates that each node in V occurs at least once in the node pairs of S.

Figure 1 provides an illustrative example of the problem. We only obtain the end-to-end path performance of some node pairs and then infer the path performance of other unmeasured node pairs.

4.3. Objective

In real networks, usually, only partial end-to-end path measurements are available. Then, we explore how to infer information about unmeasured paths as accurately as possible based on only this fraction of available path measurements. Using only measured path performance metrics for inference may overly rely on network knowledge near monitor nodes, and the distribution of the training set may not be consistent with the test set. Moreover, using overestimated augmented data to assist in inference will make the estimates too large. Therefore, our goal is to develop a generic tomography framework based on deep reinforcement learning to generate augmented data closer to ground truth for predicting end-to-end path performance metrics without making any additional assumptions about the network.

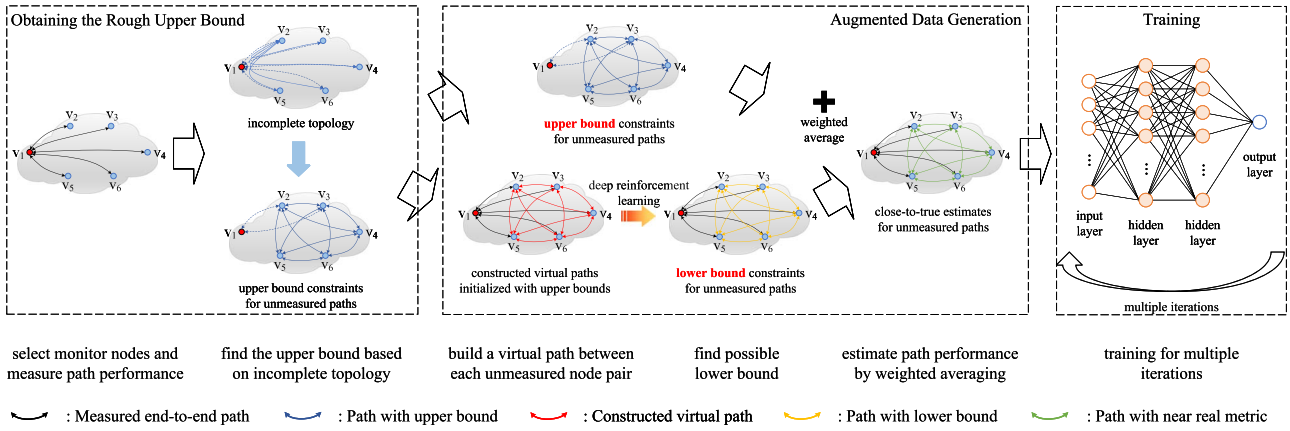


Figure 2. The framework of DRL-Tomo.

5. THE DESIGN OF DRL-TOMO

DRL-Tomo is a general tomography framework for predicting end-to-end path performance. It uses deep reinforcement learning to generate augmented data close to the ground truth, aiding in neural network training for better prediction accuracy. The DRL-Tomo framework operates in three stages, as shown in Fig. 2.

In the first stage, DRL-Tomo constructs a topology containing incomplete network information derived from measured paths. DRL-Tomo obtains the upper bounds of unmeasured path metrics based on this topology. In the second stage, DRL-Tomo generates augmented data. It first determines the lower bounds of unmeasured path metrics using deep reinforcement learning. Then, a weighted strategy is applied to thoroughly consider both the upper and lower bounds, yielding estimates closer to the real values. In the third stage, DRL-Tomo trains a neural network model for inference. It selects a portion of augmented data and combines it with raw measurement data, performing multiple rounds of iterative training on the neural network. The following subsections illustrate each stage in detail.

5.1. Obtaining the upper bound

In the first stage, DRL-Tomo obtains the upper bounds of the unmeasured path metrics. The essence of monitor node selection is probe deployment. DRL-Tomo can measure/probe the path metrics of all other nodes from the monitor nodes.

All end nodes, including monitor nodes, along with the measured paths, form a topology that contains incomplete information about the real network. Next, transformation operations are performed on this original topology, as shown in Fig. 3. In this transformation, end nodes in the real network are treated as ordinary nodes, and the measured paths connecting these end nodes are treated as ordinary internal links. We can find the shortest path between any two nodes using the shortest path algorithm on the transformed topology and then compute its metrics. Figure 3 illustrates a straightforward case with only one monitor node; the complexity increases as the number of monitor nodes grows.

Since the probes performed by the monitor nodes capture only partial knowledge of the real network, the path metric of any unmeasured node pair derived from this incomplete topology serves as an upper bound on the real path metric. For example, in Fig. 2, the path metric from v_2 to v_3 ($v_2 \rightarrow v_1 \rightarrow v_3$) should be considered an upper bound, as there may be more efficient paths between v_2 and v_3 in the real network that do not pass through v_1.

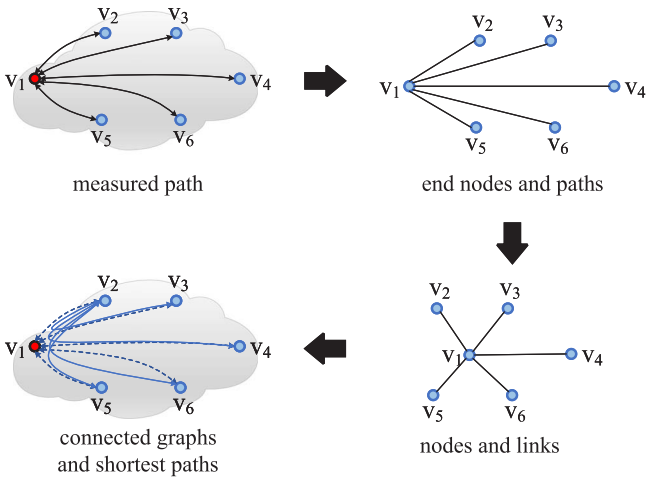


Figure 3. Example of topology transformation.

5.2. Augmented data generation using deep reinforcement learning

In the second stage, DRL-Tomo generates augmented data through deep reinforcement learning. First, it constructs virtual paths between unmeasured node pairs and initializes their metrics as upper bounds. Then, without violating real-world measurement results, DRL-Tomo gradually reduces the virtual path metrics to find the possible minimum values, establishing the lower bounds.

Not violating real-world measurements means that the optimization performed on the virtual path (i.e. reducing the virtual path metric) cannot result in improved performance for the measured paths. The performance of the measured paths is optimal, and no paths through any other end node can surpass these measured results. Therefore, while gradually reducing the virtual path metric, the optimal path metric value of the measured node pair must remain unaffected.

DRL-Tomo determines the lower bounds of path metrics through deep reinforcement learning. Reinforcement learning is a machine learning paradigm where an agent learns a policy through interaction with its environment. DRL-Tomo treats the transformed topology, including virtual paths, as the environment and views the optimization of a virtual path's metric as an action. During interaction with this environment, DRL-Tomo learns a policy to reduce the virtual path metric, gradually approaching the minimum value without violating real-world measurements.

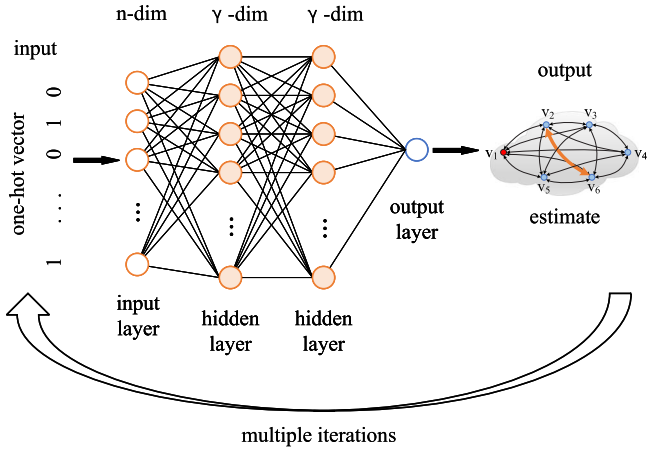


Figure 4. Multiple iterative training of the neural network model.

After determining a virtual path's upper and lower bounds, DRL-Tomo estimates a metric closer to the real value using a weighted average. A comprehensive illustration of augmented data generation is provided in Section 6.

5.3. Neural network model training

In the third stage, DRL-Tomo conducts multiple rounds of iterative training on the neural network model.

5.3.1. *Neural network model*

DRL-Tomo uses a fully connected neural network with two hidden layers to estimate the path metric for unmeasured node pairs, as shown in Fig. 4. The input layer contains n neurons, and the input data are a one-hot vector where only two positions are "1", indicating a node pair, and the rest are "0". The output layer has a single neuron, and the output value is an estimate of the node pair's path metric.

The number of neurons in the hidden layer is set to γ, representing the estimated number of links in the real network. We use the sigmoid function as the activation function $\sigma(\cdot)$ for the hidden layer, where the output represents the probability of a link existing on the path in the network. This design is motivated by the insight that path performance is the aggregate result of the performance of the links along that path.

5.3.2. *Training set*

The training set consists of two parts: the raw measured data and the augmented data generated in the second stage. DRL-Tomo performs multiple rounds of iterative training on the neural network. In each iteration, it randomly selects a certain proportion of node pairs from the unmeasured node pair set (i.e. $T \backslash S$) and combines their estimated path metrics as the augmented data training set.

DRL-Tomo uses the Adam gradient descent method to train the neural network. After each training iteration, DRL-Tomo estimates the unmeasured path metrics using the neural network. Then, the estimates of unmeasured path metrics in the augmented data set are updated with the neural network's results. Let m_{DRL} denote the metric estimate of an unmeasured path generated by the deep reinforcement learning method, and let m_{NN} denote the estimate from the neural network. DRL-Tomo updates m_{DRL} using a soft update method: $m_{\mathrm{DRL}} = \beta \cdot m_{\mathrm{DRL}} + (1-\beta) \cdot m_{\mathrm{NN}}$, where β is a control parameter balancing the new and original estimates. As the number of training iterations increases, the neural network gains a better understanding of the real network, resulting in more accurate estimates of unmeasured path metrics.

5.3.3. *Path performance prediction*

Training ceases upon reaching the maximum number of iterations. By this point, the estimate for the unmeasured path metric has been updated multiple times, aggregating the initial deep reinforcement learning-based estimate and several neural network estimates. DRL-Tomo uses the final updated estimate as the prediction for the unmeasured path metric.

5.4. The algorithm of DRL-Tomo

Algorithm 1 outlines the process of training the neural network to infer unmeasured path performance within the DRL-Tomo framework. The algorithm inputs include the set T of all node pairs, the set S of measured node pairs, the proportion of augmented data α, and the update weight β. The output is the inferred path performance metrics M^* for unmeasured node pairs. Firstly, we measure the path metrics of all node pairs in set S to obtain the matrix $M_{measure}$ containing only the measured path metrics (Line 1). Next, we apply the shortest path algorithm on the incomplete topology to calculate the shortest path between all unmeasured node pairs, forming the matrix M_{upper} with the upper bounds of unmeasured path metrics (Line 2). Any shortest path algorithm can be used here. Since we need to determine the shortest path performance between all unmeasured node pairs, we adopted the *Floyd* algorithm, a typical multi-source shortest path algorithm. Subsequently, we use the deep reinforcement learning-based algorithm *ADG_DRL* to generate augmented data (Line 3). Section 6.3 provides a detailed explanation of the *ADG_DRL* algorithm, which returns the unmeasured node pair set T' and the estimated metric matrix M^*. Finally, we train the neural network using both measured and augmented data (Lines 4-14).

First, initialize the neural network $NN(w)$ with random parameters w (Line 4). Then, perform multiple rounds of iterative training. In each iteration, randomly select a certain proportion (α) of unmeasured paths to form the augmented data training set TS_{AD} (Line 7). Combine this with the training set TS_{RD} derived from the measured paths (Line 6) to form a mixed training set TS (Line 8). Next, the neural network $NN(w)$ is trained for multiple epochs using the mixed training set (Lines 9–11). After each round of iterative training, use the trained neural network $NN(w)$ to estimate the unmeasured paths' metrics, forming the matrix M_{NN} (Line 12). Update the path metric estimate matrix M^* with these estimates using a soft update method (Line 13). After multiple iterations, the matrix M^* is returned as the performance estimates for the unmeasured paths (Line 14).

6. THE DETAILS OF AUGMENTED DATA GENERATION USING DEEP REINFORCEMENT LEARNING

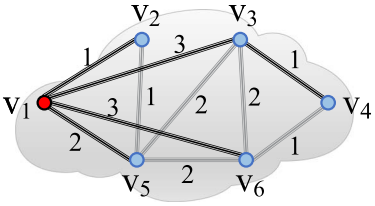
6.1. Deep reinforcement learning

Reinforcement Learning (RL) is a machine learning paradigm where an autonomous agent learns to improve task performance by interacting with its environment. In RL, the agent periodically makes decisions, observes outcomes, and adjusts actions to achieve an optimal policy [37–40]. The goal is for the agent to learn actions based on the environment's state that maximizes expected long-term rewards. This learning problem is typically modeled as a Markov Decision Problem [15, 37].

Algorithm 1 Network Tomography based on Deep Learning

Input: T, S, $\alpha \in (0,1)$ and $\beta \in [0,1)$
Output: M^*

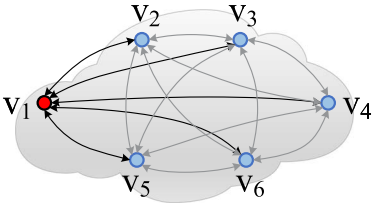
$M_{measure} = Measure(S)$
$M_{upper} = Floyd(M_{measure})$
$(T', M^*) = ADG\_DRL(T, S, M_{upper})$
Initialize neural network $NN(w)$ with random parameters w
for iteration $i = 1, \cdots, I$ **do**
  Form raw data training set TS_{RD} from S and $M_{measure}$
  Randomly select $\alpha|T'|$ node pairs from T', find the corresponding path performance estimates of these node pairs from M^*, and form the augmented data training set TS_{AD}
  $TS = TS_{RD} + TS_{AD}$
  for epoch $e = 1, \cdots, E$ **do**
    Train neural network $NN(w)$ using the training set TS and update parameters w
  end for
  Employ neural network $NN(w)$ to estimate the path performance of unmeasured node pairs in T' and form M_{NN}
  $M^* = \beta \cdot M^* + (1-\beta) \cdot M_{NN}$
end for
return M^*



internal link of the real network

	v1	v2	v3	v4	v5	v6
v1	0	1	3	-	2	3
v2	1	0	-	-	1	-
v3	3	-	0	1	2	2
v4	-	-	1	0	-	1
v5	2	1	2	-	0	2
v6	3	-	2	1	2	0

adjacency matrix



paths between pairs of end nodes

	v1	v2	v3	v4	v5	v6
v1	0	1	3	4	2	3
v2	1	0	3	4	1	3
v3	3	3	0	1	2	2
v4	4	4	1	0	3	1
v5	2	1	2	3	0	2
v6	3	3	2	1	2	0

path metric matrix (ground truth)

Figure 5. Internal link and real path performance of an example network.

Conventional reinforcement learning algorithms have significant limitations, primarily being able to solve tasks in low-dimensional state/action spaces. Deep Reinforcement Learning (DRL) combines the feature representation capabilities of deep learning with the decision-making abilities of reinforcement learning to address problems involving large state spaces and continuous action spaces.

Reinforcement Learning methods can be classified into value-based and policy-based approaches. Value-based methods, such as Deep Q-Network [41], explicitly learn the value function and implicitly learn its policy. Policy-based methods directly learn policy. Policy-based methods have the advantages of fast convergence and are well-suited for continuous or high-dimensional action spaces.

Policy Gradient is a policy-based method. Typically, a policy is parameterized as a neural network π_θ with parameter θ. Policy Gradient learns from trajectory samples through gradient descent and updates the policy parameters accordingly. The Trust Region Policy Optimization (TRPO) [42] algorithm introduces importance sampling, allowing the use of another policy, $\pi_{\theta'}$, to interact with the environment and update θ with the obtained samples. TRPO also introduces a trust-region constraint to ensure that the new and old policies do not diverge significantly after each update.

DRL-Tomo generates augmented data using the Proximal Policy Optimization (PPO) [43] algorithm, an improvement over TRPO. PPO is an Actor-Critic method that simultaneously learns policy and value functions. The actor generates policies, selects actions, and interacts with the environment, while the value function evaluates the policies. PPO employs Experience Replay to store historical interaction samples in a replay buffer, randomly and uniformly selecting samples for training.

6.2. Basic idea

In addition to end nodes, a real network contains many internal nodes directly connected by links. Paths between end node pairs consist of these links, making the path metric an aggregate of the link metrics. Given our limited knowledge of the network's interior, exploring its internal links is inefficient. Instead, we can directly explore paths between end node pairs.

Figure 5 shows a simple example, illustrating the adjacency matrix with the real link metrics of the network and the path matrix based on this adjacency matrix as the ground truth. We treat the measured path (e.g. (v_1, v_4)) as a link, which is the shortest path between the two end nodes and part of the path between other node pairs (e.g. (v_2, v_4)). Thus, when measurements are taken at the monitor node, the results only include partial links (the black links in Fig. 5). Based on this incomplete topology, only an overestimation of unmeasured path metrics can be obtained.

Our goal is to approximate the real performance of unmeasured paths based on the measured paths. However, for large-scale networks, the numerous links and nodes result in a high-dimensional state space. We use deep reinforcement learning to obtain the lower bounds of unmeasured path metrics, as it can learn the optimal policy to rapidly find the possible minimum values of the metrics for the links (the gray links in Fig. 5) not covered by the measured paths. Next, we illustrate the specific process of the proposed algorithm using the example in Fig. 5.

6.2.1. *Preprocessing*

As shown in Fig. 6, DRL-Tomo obtains measured path metrics only for the data marked in red in the matrix, leaving the areas with a gray background unknown. In the first stage of DRL-Tomo, the shortest path algorithm is employed to obtain the overestimated values of the unmeasured path metrics from the incomplete topology, serving as the upper bounds. In the preprocessing step, we construct a virtual path between any two nodes and initialize its value as the upper bound of the path metric. Then, the deep reinforcement learning algorithm is used to bring the estimated values closer to the true values.

6.2.2. *Obtain lower bound*

Due to the high complexity caused by the large number of links and nodes in the network, we use the PPO algorithm, a policy-based deep reinforcement learning method, to gradually reduce the possible values of the path metrics.

Next, we define the State, Action, Reward, and Constraint of this reinforcement Learning task.

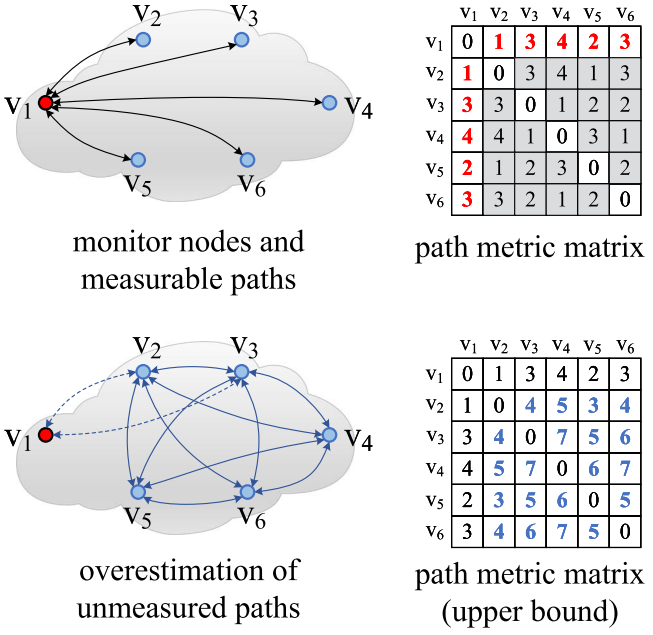


Figure 6. Overestimation of unmeasured path performance.

- **State:** the state s is defined as the current values of path metrics, organized as a matrix M of size $n \times n$, where $M(v_i, v_j)$ is the metric of the path $p(v_i, v_j)$ between end node v_i and v_j, $i, j \in n$. We consider the network's operational status as the environment, depicted by the performance of all end-to-end paths. We can only obtain the true metrics of a small portion of paths through measurement. We estimate the upper bounds of unmeasured path metrics and combine these estimates with the true metrics of the measured paths as the initial state. The estimated metrics of unmeasured paths gradually approach the true values through training.
- **Action:** action a is the operation that reduces the metric value of a certain path, i.e. causes the estimated metric $M(v_i, v_j)$ of a certain path $p(v_i, v_j)$ to be reduced to $M'(v_i, v_j)$. We consider the operation performed on a path that tightens its metric as an action selected to interact with the environment.
- **Reward:** the reward for performing the action a that reduces $M(v_i, v_j)$ to $M'(v_i, v_j)$ is defined as

$$r(s,\ a)\ = \begin{cases} \mathrm{M}(v_i, v_j) - M'(v_i, v_j) & \textit{if successful} \\ -c & \textit{if failed} \end{cases}$$

For an action selected in a certain state, if the action is successfully performed (i.e. the path metric is indeed tightened), the reward is set to the corresponding reduction of the metric value; if it is failed (i.e. subject to constraints), the reward is set to a small negative value to discourage the agent from selecting inefficient actions.

- **Constraint:** the metric of the measured path cannot be changed when reducing the metric of a path. This constraint ensures that there cannot be a better-performing path between the measured node pairs via the changed paths. This constraint ensures that there cannot be a better performing path between the measured node pairs via the changed paths. Such a path does not exist in the real network; otherwise, the measurement result would be better. For example, assuming $p(v_i, v_k)$ is a measured path, and $M(v_i, v_k)$ is its true path metric. If an action a reduces $M(v_i, v_j)$ to $M'(v_i, v_j)$ such that $M'(v_i, v_j) + M(v_j, v_k) < M(v_i, v_k)$, then the action will not be performed, and the state will not change. This is because if the metric of $p(v_i, v_k)$ in the real network were $M'(v_i, v_j)$, the measured $M(v_i, v_k)$ would be smaller (better), at least as small as $\mathrm{M}M'(v_i, v_j) + M(v_j, v_k)$, which is inconsistent with the measurement results. Under this constraint, the agent learns the optimal policy to rapidly reach a state that can no longer be changed (i.e. each unmeasured path metric carries the minimum value).

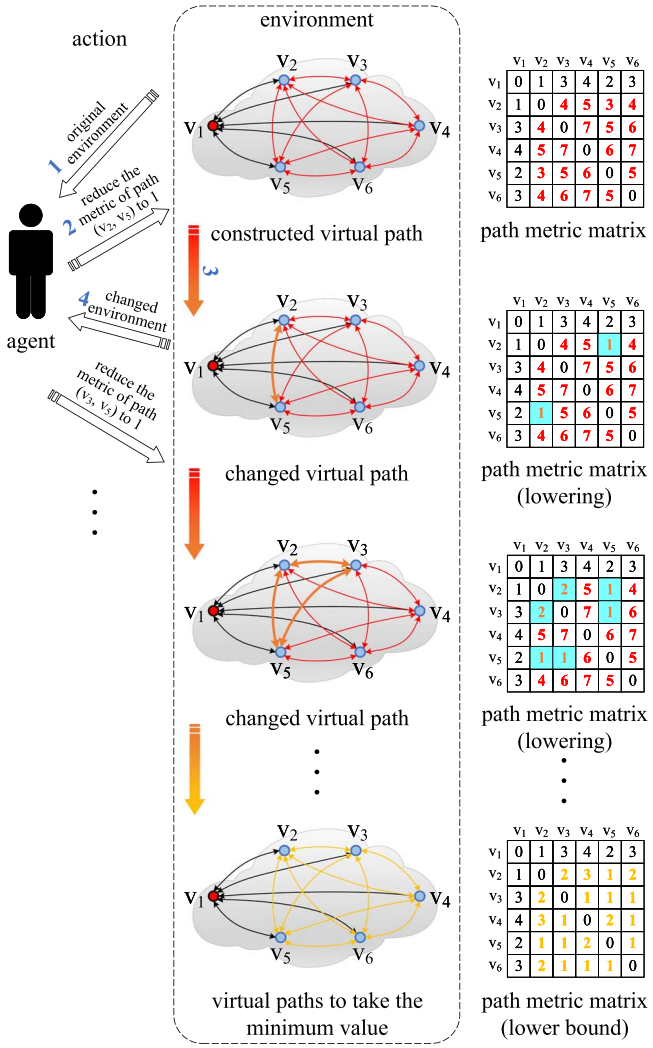


Figure 7. Interaction process between the agent and the environment.

DRL-Tomo learns a policy to reduce path metrics, as shown in Fig. 7. It first observes the state of the original environment and then chooses an action. After the environment changes, DRL-Tomo observes the new state and selects another action. Through continuous interaction, the agent learns the optimal policy. Note that when the agent reduces the metric of one or several paths (e.g. $v_2 \leftrightarrow v_5$ and $v_3 \leftrightarrow v_5$ in Fig. 7), the metric of other paths (e.g. $v_2 \leftrightarrow v_3$ in Fig. 7) will change accordingly. In large-scale networks, paths are more correlated. DRL-Tomo learns the critical links in the network to obtain lower bounds of unmeasured paths as fast as possible.

6.2.3. *Obtain close-to-true estimate*

After obtaining the upper and lower bounds of the path metrics, we estimate the path metrics closer to the true values using

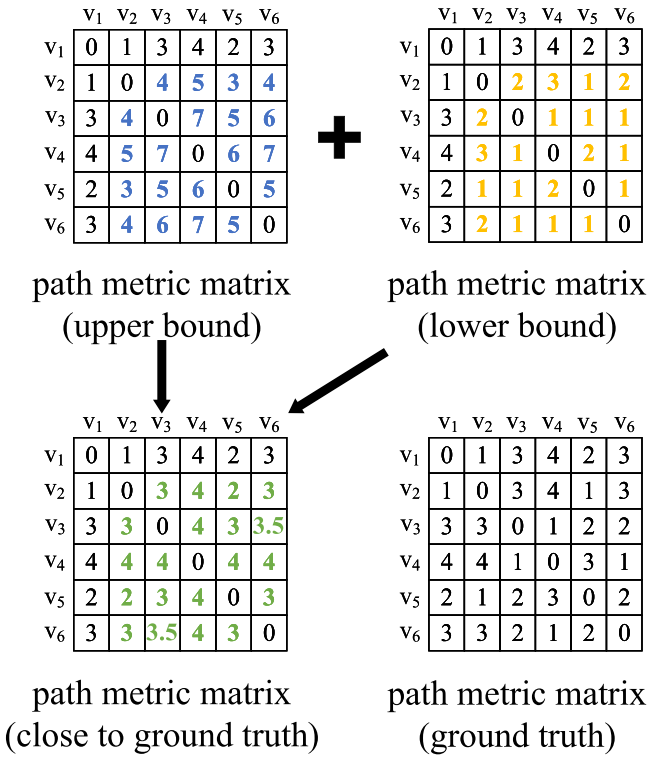


Figure 8. Schematic of average weighting.

weighted averaging. DRL-Tomo employs the following two strategies:

- **Average Weighting:** DRL-Tomo uses equal weights for the upper and lower bounds; thus, the metric for a given path is estimated as $m^* = (m_{upper} + m_{lower})/2$.
- **Adaptive Weighting:** When the measured path covers only a small portion of the links in the real network, it imposes weaker constraints on the uncovered links, especially those farther from the monitor nodes. Some unmeasured paths may have extremely low metric values. To mitigate this influence on the estimation results, DRL-Tomo adopts adaptive weighting, adjusting the weights according to the change in the lower bound relative to the upper bound. DRL-Tomo sets a weight α ($\alpha = (3 \times m_{lower} + m_{upper})/(3 \times m_{upper} + m_{lower})$) for the path metric lower bound, such that $m^* = (1 - \alpha) \times m_{upper} + \alpha \times m_{lower}$.

Figure 8 provides an example of average weighting, showing that the path metric estimation after applying the weighted average is closer to the actual value.

6.3. Augmented data generation algorithm

Algorithm 2 details the process of augmented data generation based on deep reinforcement learning. The input consists of the set T of all node pairs, the set S of measured node pairs, and the path metric overestimate matrix M_{upper}. The output is the set $T \backslash S$ of unmeasured node pairs and the path metric estimate matrix M^*. First, we set $M' = M_{upper}$, constructing virtual paths and initializing them with the upper bounds (Line 1). M' represents the training environment. Next, we call the function *DRL_training*(), which performs the deep reinforcement learning training process (Line 2). *DRL_training*() returns the final state, from which we derive the lower bound matrix M_{lower} for unmeasured path metrics (Line 3). The path metric estimate matrix M^* is then calculated based on the selected weighting strategy (Line 4). Finally, all unmeasured node pairs and their corresponding close-to-true path metric estimates are returned (Line 5).

Algorithm 2 Augmented Data Generation based on Deep Reinforcement Learning (ADG_DRL)

Input: T, S and M_{upper}
Output: $T \backslash S$, M^*
1: $M' = M_{upper}$
2: $State = DRL\_training(M')$
3: $M_{lower} = Get\_path\_matrix(State)$
4: $M^* = Weighted\_average(M_{upper}, M_{lower})$
5: **return** $T \backslash S$, M^*
6: **function** $DRL\_training(M')$
7: Initialize actor network $\mu(s, a|\theta^\mu)$ and critic network $Q(s, a|\theta^Q)$ with random parameters θ^μ, θ^Q
8: Initialize replay buffer R
9: **for** episode $e = 1, \cdots, E$ **do**
10: $State = Env\_reset(T, S, M')$
11: $Done = False$
12: $t = 1$
13: **while** $Done == False$ **do**
14: $Action = Choose\_action(State)$
15: $(State', Reward, Done) == Env\_update(Action)$
16: $t = t + 1$
17: $R.Store(State, Action, Reward, Done)$
18: **if** $t\%Update\_Period == 0$ **then**
19: **for** minibatch $b = 1, \cdots, B$ **do**
20: $R' = R.Sample()$
21: Update actor network parameters θ^μ
22: Update critic network parameters θ^Q
23: **end for**
24: **end if**
25: $State = State'$
26: **end while**
27: **end for**
28: **return** $State$
29: **end function**

Function *DRL_training*() learns a policy for taking actions to rapidly tighten the unmeasured path metrics (Lines 6–29). As mentioned earlier, the PPO algorithm we adopt follows an Actor-Critic structure, consisting of two networks: the actor network and the critic network. The actor network learns the policy, while the critic network evaluates it. We first initialize both networks with random parameters (Line 7). Then, we initialize the replay buffer pool (Line 8). PPO utilizes experience replay, where historical samples generated by interacting with the environment are stored in the replay buffer. PPO uniformly and randomly selects samples from the buffer for training. The training process involves multiple episodes. For each episode, the state is initialized with M' (Line 10). After setting *Done* to *False* (Line 11), it interacts with the environment multiple times, executing several steps. For each step, the PPO algorithm chooses an action based on the current state, and the environment changes, returning the new *State*, *Reward*, and *Done* (Lines 14–15). The action is chosen by sampling the probability distribution output by the actor network. The environment executes the chosen action, transitions to the new state, and returns the reward based on the action's success. We store (*State*, *Action*, *Reward*, *Done*) in the replay buffer R (Line 17), facilitating experience replay. When the step satisfies the update condition (i.e. enough interactions have occurred), we sample entries from the replay buffer and update the parameters

Table 2. AS Topology in Rocketfuel and Internet Topology Zoo.

AS	Nodes	Links	Link average metric	Path average metric
3967	79	147	5.19 ms	24.18 ms
1221	104	151	2.78 ms	15.79 ms
3257	161	328	4.30 ms	15.65 ms
Columbus (delay)	70	85	3.79 ms	24.19 ms
Columbus (loss)	70	85	0.96%	5.77%

of the actor network and critic network (Lines 20–22). Finally, the function returns the final state (Line 28).

7. EXPERIMENTS

In this section, we evaluate the accuracy performance of DRL-Tomo for path delay inference and path loss rate inference. We first describe the real-world and synthetic datasets used in the experiments. Then, we present benchmark solutions along with experimental settings. We conducted experiments on a Linux platform with an Intel Core i9-9900KF CPU (3.60GHz) and 32 GB DRAM memory.

7.1. Dataset

We use measurement data of Autonomous System (AS) networks collected by the Rocketfuel [44] project as a real-world dataset for path delay inference. Rocketfuel is an Internet Service Provider (ISP) topology mapping engine that uses routing information to focus efforts on one ISP at a time and employs ISP-specific router naming conventions to understand the topology. Additionally, Rocketfuel publishes the inferred link metrics dataset, which is consistent with observed routing. The Rocketfuel project provides topologies (IP-level connections between backbone/gateway routers) and corresponding link metrics for several ASes from major ISPs around the globe. We selected AS3967, AS1221, and AS3257 from the dataset for experiments in path delay inference.

Besides, we constructed synthetic datasets for path delay inference and path loss rate inference based on *Columbus Networks* (Columbus for short), a network topology collected by the Internet Topology Zoo [45] project. The Internet Topology Zoo is an ongoing project that collects network topologies from around the world. It features over 250 network topologies from various network providers. Unlike Rocketfuel, the Internet Topology Zoo does not provide link metric information. For the path delay inference experiment, we assigned random values to the links of Columbus, making their distribution approximate the link metric distribution of AS3257 in Rocketfuel. For the path loss rate inference experiment, we assigned random values within the interval [0.002, 0.018] to the links in Columbus.

Table 2 shows the number of nodes, the number of links, the average link performance metric, and the path average performance metric for the selected network topologies.

7.2. Benchmark solutions

We compared DRL-Tomo with ***NeuTomogrphy*** [3] and selected three benchmark solutions:

- **Random:** When training the neural network model, path measurement data of node pairs randomly sampled from the entire network are used as training data without any augmented data.
- **Monitor:** When training the neural network model, randomly selected monitor nodes in the entire network are used, and the path measurement data probed from monitor nodes to other nodes serve as training data without augmented data.
- **PAT:** When training the neural network model, the measurements probed from the monitor nodes and the overestimated data (i.e. only the upper bounds) based on the incomplete topology formed from these measurements are used as training data.

We use ADG-AVW (Augmented Data Generation-AVerage Weighting) and ADG-ADW (Augmented Data Generation-ADaptive Weighting) as shorthand for our proposed scheme.

7.3. Experiment settings

DRL-Tomo chooses the following parameters in the experiments. We sample 20%, 25%, and 30% of the path measurement data as training data. Since the given measurement data cover all nodes in the network, the dimension n of the input layer in Fig. 4 can be determined. For the dimension γ of the hidden layer, i.e. the number of network links, we estimate it by the average node degree (defined as $2\gamma/n$) in the real network. Previous work [46] pointed out that the average node degree in practical communication networks is generally between 1 and 5. Therefore, we set γ as $\gamma = 2.5n$. For the four selected topologies, γ is overestimated, as shown in Table 2. In order to balance accuracy and training time, the number of hidden layers is set to 2. Meanwhile, we appropriately scale down the four network topologies without affecting the network characteristics. We randomly sample a limited number (40, 52, 54, 35, respectively) of nodes and the links between them in AS3967, AS1221, AS3257, and Columbus, to form four smaller networks. To better learn the relationship between the measured paths and the unmeasured paths, for path loss rate inference, the output of the neural network is set as the logarithm of the path loss rate. This way, the neural network needs to learn an additive relationship, not a multiplicative one. We choose the mean squared error as the loss function and Adam as the optimizer. The maximum number of iterations is set to 6, the augmented data sampling rate α is set to 15%, the soft update parameter β is set to 0.6, and training is performed 1000 epochs in each iteration. Note that the Random and Monitor schemes do not use augmented data; thus, instead of iterative training, only one round of training is performed, and the neural network output is the estimate of the path performance.

For the training process of deep reinforcement learning, we choose the following parameters for the actor network and critic network. Both networks are fully connected neural networks. The input layer dimension is set to n^{Θ} to fully describe the state of the network. The hidden layer dimension is set to $\gamma = 2.5n$. The output layer of the actor network is a sampling probability model for k actions (where k is the number of actions), thus having a dimension of k. The output layer of the critic network has a dimension of 1. We set the number of hidden layers to 2 to learn the correlation between nodes and internal links in the network. Both the actor network and critic network use Adam as the optimizer, and the number of episodes is set to tor-network and critic-network use Adam as the optimizer, and the number of episodes is set to 10.

7.4. Experimetal results

We use mean absolute percentage error (MAPE) as the accuracy evaluation metric, i.e.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|.$$

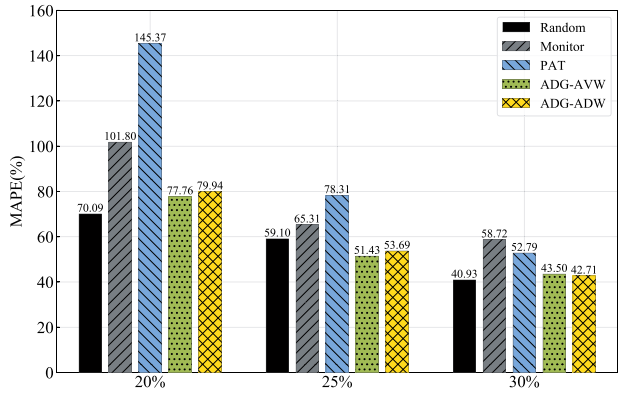


Figure 9. Path Delay Prediction Error (MAPE in %) of DRL-Tomo on AS3967 (ADG-AVW/ADG-ADW: Augmented Data Generation with AVerage Weighting/ADaptive Weighting).

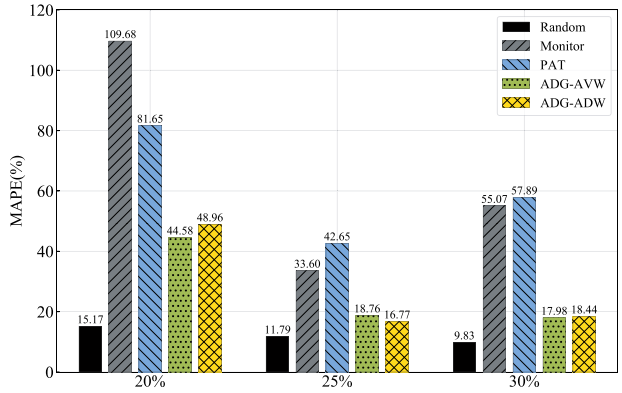


Figure 10. Path Delay Prediction Error (MAPE in %) of DRL-Tomo on AS1221 (ADG-AVW/ADG-ADW: Augmented Data Generation with AVerage Weighting/ADaptive Weighting).

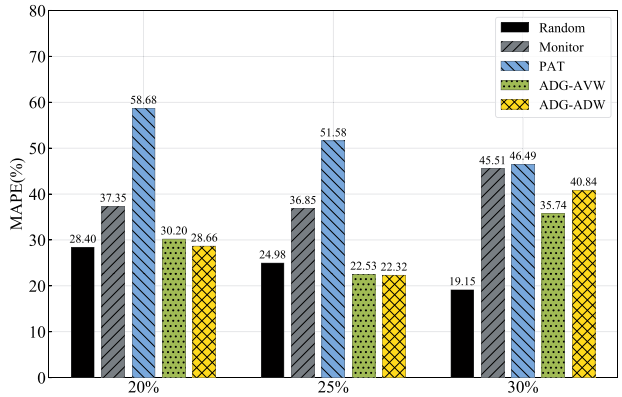


Figure 11. Path Delay Prediction Error (MAPE in %) of DRL-Tomo on AS3257 (ADG-AVW/ADG-ADW: Augmented Data Generation with AVerage Weighting/ADaptive Weighting).

Figures 9, 10, 11, and 12 present the experimental results of path delay inference for the four topologies, while Fig. 13 shows the experimental results of path loss rate inference for Columbus.

7.4.1. *Overall trends*

First of all, for path delay inference, it is evident that the three schemes—Random, ADG-AVM, and ADG-ADW—are more accurate for path prediction on AS1221. AS1221 is a relatively sparse graph with fewer links (the ratio of the number of links to the number of nodes is about 1.45). This sparsity allows for better

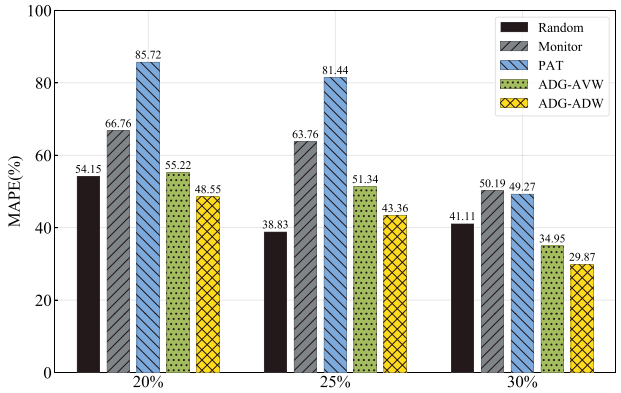


Figure 12. Path Delay Prediction Error (MAPE in %) of DRL-Tomo on Columbus (ADG-AVW/ADG-ADW: Augmented Data Generation with AVerage Weighting/ADaptive Weighting).

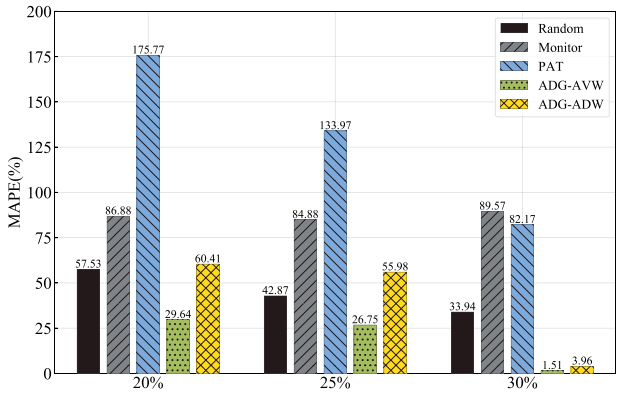


Figure 13. Path Loss Rate Prediction Error (MAPE in %) of DRL-Tomo on Columbus (ADG-AVW/ADG-ADW: Augmented Data Generation with AVerage Weighting/ADaptive Weighting).

estimation of links not covered by the measured paths. Conversely, if the unknown area of the network contains more details, we can only provide a rough estimate for the numerous links, as the measurement data are insufficient for precise estimation. For Columbus, which is even sparser than AS1221 (the ratio of the number of links to the number of nodes is about 1.21), path performance inference performed poorly for both delay inference and loss rate inference. In such a sparse network, there is a high probability of overlap among different measurement paths. Consequently, less information about the network can be obtained from the measurement paths, leading to less accurate estimation results.

Then, we observe that as the sampling rate increases, the MAPE of the Random scheme decreases for path delay predictions on the three topologies (AS3967, AS1221, AS3257) and for path loss rate prediction on Columbus, which aligns with expectations. However, for path delay prediction on Columbus, the MAPE of the Random scheme at a 30% sampling rate is slightly larger than at 25%. This is because, in an overly sparse network, the additional information from sampling 5% more paths is insufficient, potentially causing the sampled paths to become more clustered. This clustering makes the neural network focus more on the links within the aggregated area during learning. Except for AS3967, in predictions for the other three topologies, the MAPE of the Monitor, PAT, and ADG schemes first decreases and then increases to varying degrees. The reason is that, in experiments with different sampling rates, the random sampling of monitor nodes may result in clustering within one area of the network. As a result, despite an

increased sampling rate, the clustering of monitor nodes leads to less information about the network being obtained through their measurements, resulting in worse prediction outcomes.

7.4.2. Accuracy of path delay inference

Next, we analyze the path delay inference performance of the five schemes in detail.

The Random scheme performs nearly the best among all schemes, only slightly worse than our proposed ADG-AVW and/or ADG-ADW in a few cases (AS3967 with a 25% sampling rate; Columbus with a 30% sampling rate). This is reasonable because the Random scheme selects node pairs uniformly and randomly across the entire network, allowing the resulting measurement paths to cover more internal links in the network.

The Monitor scheme presented poor results, even reaching 109.68% MAPE (as shown in Fig. 10). The measurement path selection method based on the monitor nodes leads to the measurement results repeatedly covering many links. For example, in the network shown in Fig. 5, the paths $v_1 \leftrightarrow v_3$, $v_2 \leftrightarrow v_3$, and $v_2 \leftrightarrow v_4$ all make the neural network learn the information of the link $v_2 \leftrightarrow v_3$. Measurement paths from monitor nodes cover the links closer to them with higher frequency, making it difficult to learn the knowledge of links far from the monitor nodes. Using only the measurement data for training causes the neural network to focus on learning the partial network knowledge covered by the measured paths and does not explore the area not covered by these paths, resulting in inaccurate estimation.

In most cases, the accuracy performance of the PAT scheme using augmented data is worse than that of the Monitor scheme. However, in some instances (AS3967 with a 30% sampling rate; AS1221 with a 20% sampling rate; Columbus with a 30% sampling rate), the prediction performance of PAT is better than that of Monitor. The augmented data in the PAT scheme completely depend on the estimation of the upper bound of the path performance. If the estimated upper bound is too large, the prediction performance of the PAT scheme will also be relatively poor. PAT is trained using augmented data that overestimate the actual value, and although it attempts to learn the areas not covered by the measured paths, it ends up learning larger results for links within these areas. Consequently, it is not optimal.

For DRL-Tomo, the results in Fig. 9, Fig. 10, Fig. 11, and Fig. 12 show that the prediction performance of the ADG-AVW and ADG-ADW schemes outperforms the Monitor and PAT schemes in all cases and is only slightly worse than the Random scheme. For links in areas not covered by the measured paths, these schemes use estimates close to the actual value as augmented data to help train the neural network model. The neural network can better learn the relationship between paths and links in the real network. Next, we discuss the performance improvements of DRL-Tomo over the existing neural network-based schemes (Monitor and PAT). We choose the one with the better performance between the Monitor and PAT schemes as the basis of comparison (e.g. for the predictions on AS3967, Monitor is chosen when the sampling rate is 25%, and PAT is chosen when the sampling rate is 30%). For predictions on AS1221, DRL-Tomo shows significant performance improvement, with the ADG-AVW and ADG-ADW schemes bringing 44%–67% and 40%–66% improvements, respectively. This aligns with the conclusion in Section 7.4.1 that simpler network topologies make it easier for DRL-Tomo to estimate links not covered by measured paths. For predictions on AS3967, the ADG-AVW and ADG-ADW schemes bring performance improvements of 17%–23% and 19%–21%, respectively. For AS3257, these schemes bring improvements of 19%–38% and 10%–39%, respectively. For Columbus, the ADG-AVW and ADG-ADW schemes bring improvements of 17%–29% and 27%–39%, respectively. It is noted that the performance improvement of the ADG-ADW scheme is only 10.2% when 30% of the node pairs are sampled for prediction on AS3257. This is because the adaptive weighting strategy is conservative, assigning more weight to the estimated upper bound. When the upper bound estimation error is larger, the ADG-ADW scheme also has a larger estimation error. In the same scenario, the ADG-AVW scheme shows a 21.5% performance improvement because it assigns equal weight to the upper and lower bounds.

Next, we compare the performance of ADG-AVW and ADG-ADW. As shown in Fig. 9, when the sampling rate is relatively small, the prediction performance of ADG-AVW is better than that of ADG-ADW. However, when the sampling rate is large, ADG-ADW outperforms ADG-AVW. This is because, at a small sampling rate, the measurement paths obtained through the monitor nodes cover only a minor portion of the links in the real network, providing limited knowledge of other links. At this stage, the estimation of the upper bound of path performance is relatively loose and inaccurate. Average weighting employs an aggressive strategy that assigns equal confidence and weight to the upper and lower performance bounds, resulting in better performance. As the sampling rate increases, more links in the real network are covered by the measured paths, and the information about the network carried by the path performance upper bounds becomes more useful. By avoiding assigning a larger weight to m_{lower} with lower confidence, the adaptive weighting method outperforms the average weighting method. In the predictions for AS1221 and AS3257, the performance comparison of ADG-AVW and ADG-ADW follows the same trend, except for the results at a 30% sampling rate. This anomaly is due to the clustering of monitor nodes, which leads to less network information being obtained through measurement compared with a 20% sampling rate. For predictions on Columbus, the performance of ADG-ADW is consistently better than that of ADG-AVW across different sampling rates, indicating that ADG-ADW benefits from relatively tight upper bounds even at a 20% sampling rate.

7.4.3. Accuracy of path loss rate inference

Then, we analyze the performance of path loss rate inference on Columbus for the five schemes.

First, we note that the estimation errors of the Monitor and PAT schemes are relatively large in all cases, with MAPE even reaching 175.77% under the 20% sampling rate. This is because the path loss rate is small relative to the path delay. Although the estimation errors of the Monitor and PAT schemes are small in absolute value, the result is unacceptable compared with the real metrics. The Random, ADG-AVW, and ADG-ADW schemes have relatively good estimation accuracy. Notably, ADG-AVW has the best performance among all schemes, outperforming the Random scheme under all sampling rates. ADG-ADW performs better than Random only at a sampling rate of 30%. At 20% and 25% sampling rates, the upper bounds on path performance obtained based on the measured paths are very loose, as indicated by the estimation performance of PAT (the PAT scheme uses the upper bounds as augmented data for training). When the sampling rate reaches 30%, the estimation performance of PAT significantly improves, even surpassing the Monitor scheme. In this case, the upper bounds of the path loss rate are relatively tight. The ADG-ADW scheme sets higher confidence in the upper bounds of path performance, resulting in better performance as the upper bounds of the path loss rate tighten. In contrast, the ADG-AVW scheme employs a more aggressive estimation strategy, assigning equal confidence to the upper and lower bounds of the

path performance. Despite the loose upper bounds of the path loss rate, ADG-AVW still demonstrates better estimation performance. When the sampling rate reaches 30%, the MAPE of the ADG-AVW scheme is even as low as 1.51%.

Next, we discuss the performance improvements of DRL-Tomo over existing neural network-based solutions (Monitor and PAT). We select the better-performing scheme between Monitor and PAT as the basis for comparison (e.g. Monitor is chosen when the sampling rate is 25%, and PAT is chosen when the sampling rate is 30%). As shown, the ADG-AVW and ADG-ADW schemes bring performance improvements of 65%–98% and 30%–95%, respectively. DRL-Tomo significantly enhances the accuracy of path loss rate inference.

8. DISCUSSION

In this section, we discuss the limitations of DRL-Tomo and outline relevant future research directions.

First, the estimation accuracy of DRL-Tomo depends on the selection of measurement paths, i.e. the placement of monitor nodes. When the monitor nodes are clustered, the knowledge about the network is insufficient and biased. The upper bounds of unmeasured path performance based on measured paths are also loose. DRL-Tomo constructs augmented data closer to the real value based on the measured paths to train the neural network, and the estimation accuracy will be relatively poor. Therefore, future studies should consider how to deploy new monitor nodes to maximize the estimation accuracy of DRL-Tomo.

Second, the basic premise for accurate estimation by DRL-Tomo is that the measured path performance is reliable. However, existing work [16, 47, 48] has investigated attacks against network tomography that maliciously manipulate end-to-end measurements. When a part of the probed paths is manipulated, the performance of the unmeasured paths inferred by DRL-Tomo is inaccurate based on the manipulated measurements. The estimates may significantly deviate from the real path performance because the knowledge about the network is incorrect. Therefore, DRL-Tomo is vulnerable to attacks targeting network tomography. Future research should explore methods to detect maliciously manipulated path measurements and mitigate the effects of such adversarial actions on DRL-Tomo.

9. CONCLUSION

In this work, we propose a network tomography framework, DRL-Tomo, which utilizes only a subset of complete end-to-end path performance data to infer the characteristics of the entire network through a deep neural network. Additionally, we introduce a novel approach for generating augmented data through deep reinforcement learning, facilitating the rapid creation of data that closely emulate real-world path performance. We conducted experiments on a real-world dataset collected by the Rocketfuel project and a synthetic dataset built on a topology from the Internet Topology Zoo project. The results demonstrate that our approach outperforms existing state-of-the-art solutions in accurately predicting path performance. DRL-Tomo achieves substantial performance improvements, ranging from 10% to 67% in path delay prediction and 30% to 98% in path loss rate prediction.

FUNDING

This work is supported by the National Natural Science Foundation of China [62072465, 62172155]; the China Postdoctoral Science Foundation [2023TQ0089]; and the Science and Technology Innovation Program of Hunan Province [2022RC3061, 2023RC3027].

DATA AVAILABILITY

The dataset used in this article is publicly available on the Internet. The data will be shared on reasonable request [44, 45]. The source code of DRL-Tomo and the underlying code used to perform the analyses described in this study can be found at https://github.com/houchangsheng/DRL-Tomo.

REFERENCES

1. Arrigoni V, Prata M, Bartolini N. Tomography-based progressive network recovery and critical service restoration after massive failures. *IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, 17–20 May, 2023, pp. 1–10. IEEE, Piscataway.
2. Tao X, Silvestri S. Network tomography and reinforcement learning for efficient routing. *IEEE International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, Toronto, ON, 25–27 September, 2023, pp. 384–389. IEEE, Piscataway.
3. Ma L, Zhang Z, Srivatsa M. *Neural network tomography* 2020.
4. Lo Presti F, Duffield NG, Horowitz J. et al. Multicast-based inference of network-internal delay distributions. *IEEE/ACM Trans Netw* 2002; **10**:761–75. https://doi.org/10.1109/TNET.2002.805026.
5. Tao X, Monaco D, Sacco A. et al. Delay-aware routing in software-defined networks via network tomography and reinforcement learning. *IEEE Trans Netw Sci Eng* 2024; **11**:3383–97. https://doi.org/10.1109/TNSE.2024.3371384.
6. Vardi Y. Network tomography: estimating source-destination traffic intensities from link data. *J Am Stat Assoc* 1996; **91**:365–77. https://doi.org/10.1080/01621459.1996.10476697.
7. Medina A, Taft N, Salamatian K. et al. Traffic matrix estimation: existing techniques and new directions. *ACM SIGCOMM Comput Commun Rev* 2002; **32**:161–74. https://doi.org/10.1145/964725.633041.
8. Li H, Gao Y, Dong W, Chen C. Bound-based network tomography for inferring interesting link metrics. *IEEE Conference on Computer Communications (INFOCOM)*, Toronto, ON, 06-09 July, 2020, pp. 1588–1597. IEEE, Piscataway.
9. Ikeuchi H, Saito H, Matsuda K. Network tomography based on adaptive measurements in probabilistic routing. *IEEE Conference on Computer Communications (INFOCOM)*, London, 02–05 May, 2022, pp. 2148–2157. IEEE, Piscataway.
10. Xue L, Marina MK, Li G, Zheng K. PAINT: path aware iterative network tomography for link metric inference. *IEEE International Conference on Network Protocols (ICNP)*, Lexington, KY, 30 October - 02 November, 2022, pp. 1–12. IEEE, Piscataway.
11. Arrigoni V, Bartolini N, Massini A, Trombetti F. Failure localization through progressive network tomography. *IEEE Conference on Computer Communications (INFOCOM)*, Vancouver, BC, 10–13 May, 2021, pp. 1–10. IEEE, Piscataway.
12. Li Y, Deng K. Simultaneous topology and loss tomography via a modified theme dictionary model. *IEEE Trans Signal Process* 2022; **70**:4239–51. https://doi.org/10.1109/TSP.2022.3191807.
13. Feng C, Wang L, Wu K, Wang J. Controlling the maximum link estimation error in network performance tomography. *IEEE/ACM International Symposium on Quality of Service (IWQOS)*, Tokyo, 25–28 June, 2021, pp. 1–7. IEEE, Piscataway.
14. Feng C, An J, Wu K, Wang J. Bound inference and reinforcement learning-based path construction in bandwidth

tomography. *IEEE Conference on Computer Communications (INFOCOM)*, Vancouver, BC, 10–13 May, 2021, pp. 1–10. IEEE, Piscataway.
15. Feng C, An J, Wu K. et al. Bound inference and reinforcement learning-based path construction in bandwidth tomography. *IEEE/ACM Trans Netw* 2022; **30**:501–14. https://doi.org/10.1109/TNET.2021.3118006.
16. Zhao S, Lu Z, Wang C. Measurement integrity attacks against network tomography: feasibility and defense. *IEEE Trans Dependable Secure Comput* 2021; **18**:2617–30.
17. Rahali M, Sanner J.-M, Rubino G. TOM: a self-trained tomography solution for overlay networks monitoring. *IEEE Annual Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, 10–13 January, 2020, pp. 1–6. IEEE, Piscataway.
18. Rkhami A, Hadjadj-Aoul Y, Rubino G, Outtagarts A. On the use of machine learning and network tomography for network slices monitoring. *IEEE International Conference on High Performance Switching and Routing (HPSR)*, Paris, 07–10 June, 2021, pp. 1–7. IEEE, Piscataway.
19. Ibraheem A, Sheng Z, Parisis G, Tian D. Neural network based partial tomography for in-vehicle network monitoring. *IEEE International Conference on Communications Workshops (ICC Workshops)*, Montreal, QC, 14–23 June, 2021, pp. 1–6. IEEE, Piscataway.
20. Ibraheem A, Sheng Z, Parisis G. *et al.* Internal network monitoring with dnn and network tomography for in-vehicle networks. *IEEE International Conference on Unmanned Systems (ICUS)*, Guangzhou, 28–30 October, 2022, pp. 928–933. IEEE, Piscataway.
21. Ushizuka Y, Kawahara R. Link delay estimation under undeterministic routing using neural network. *IEICE Communications Express* 2024; **13**:5–8. https://doi.org/10.23919/comex.2023XBL0132.
22. Sartzetakis I, Varvarigos E. Machine learning network tomography with partial topology knowledge and dynamic routing. *IEEE Global Communications Conference (GLOBECOM)*, Rio de Janeiro, 04–08 December, 2022, pp. 4922–4927. IEEE, Piscataway.
23. Sartzetakis I, Varvarigos E. Network tomography with partial topology knowledge and dynamic routing. *J Network Syst Manage* 2023; **31**. https://doi.org/10.1007/s10922-023-09763-y.
24. Pan S, Li P, Zeng D. et al. A Q -learning based framework for congested link identification. *IEEE Internet Things J.* 2019; **6**:9668–78. https://doi.org/10.1109/JIOT.2019.2930459.
25. Nie L, Wang H, Jiang X. et al. Traffic measurement optimization based on reinforcement learning in large-scale its-oriented backbone networks. *IEEE Access* 2020; **8**:36988–96. https://doi.org/10.1109/ACCESS.2020.2975238.
26. Zhan P, Qin G, Qian X. *et al.* Deep reinforcement learning based fast anomaly detection and localization for programmable networks. *IEEE International Conference on Communications (ICC)*, Rome, 28 May - 01 June, 2023, pp. 4866–4872. IEEE, Piscataway.
27. Ren W, Dong W. Robust network tomography: K-identifiability and monitor assignment. *IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, 10–14 April, 2016, pp. 1–9. IEEE, Piscataway.
28. Gao Y, Dong W, Wu W. et al. Scalpel: scalable preferential link tomography based on graph trimming. *IEEE/ACM Trans Netw* 2016; **24**:1392–403. https://doi.org/10.1109/TNET.2015.2411691.
29. Dong W, Gao Y, Wu W. et al. Optimal monitor assignment for preferential link tomography in communication networks. *IEEE/ACM Trans Netw* 2017; **25**:210–23. https://doi.org/10.1109/TNET.2016.2581176.
30. Yang R, Feng C, Wang L. *et al.* On the optimal monitor placement for inferring additive metrics of interested paths. *IEEE International Conference on Computer Communications (INFOCOM)*, Honolulu, HI, 16–19 April, 2018, pp. 2141–2149. IEEE, Piscataway.
31. Feng C, Wang L, Wu K, Wang J. Bound-based network tomography with additive metrics. *IEEE Conference on Computer Communications (INFOCOM)*, Paris, 29 April - 02 May, 2019, pp. 316–324. IEEE, Piscataway.
32. Feng C, Wang L, Wu K. et al. Bound inference in network performance tomography with additive metrics. *IEEE/ACM Trans Netw* 2020; **28**:1859–71. https://doi.org/10.1109/TNET.2020.3000115.
33. Li H, Gao Y, Dong W. et al. Preferential link tomography in dynamic networks. *IEEE/ACM Trans Netw* 2019; **27**:1801–14. https://doi.org/10.1109/TNET.2019.2931047.
34. He T, Gkelias A, Ma L. et al. Robust and efficient monitor placement for network tomography in dynamic networks. *IEEE/ACM Trans Netw* 2017; **25**:1732–45. https://doi.org/10.1109/TNET.2016.2642185.
35. Li H, Gao Y, Dong W. et al. Taming both predictable and unpredictable link failures for network tomography. *IEEE/ACM Trans Netw* 2018; **26**:1460–73. https://doi.org/10.1109/TNET.2018.2834141.
36. Xu S, Kodialam M, Lakshman TV. et al. Tomography based learning for load distribution through opaque networks. *IEEE Open J Commun Soc* 2021; **2**:656–70. https://doi.org/10.1109/OJCOMS.2021.3068222.
37. Wang X, Wang S, Liang X. et al. Deep reinforcement learning: a survey. *IEEE Trans Neural Networks Learn Syst* 2024; **35**:5064–78. https://doi.org/10.1109/TNNLS.2022.3207346.
38. Kiran BR, Sobh I, Talpaert V. et al. Deep reinforcement learning for autonomous driving: a survey. *IEEE Trans Intell Transp Syst* 2022; **23**:4909–26.
39. Arulkumaran K, Deisenroth MP, Brundage M. et al. Deep reinforcement learning: a brief survey. *IEEE Signal Process. Mag.* 2017; **34**:26–38. https://doi.org/10.1109/MSP.2017.2743240.
40. Luong NC, Hoang DT, Gong S. et al. Applications of deep reinforcement learning in communications and networking: a survey. *IEEE Commun Surv Tutorials* 2019; **21**:3133–74. https://doi.org/10.1109/COMST.2019.2916583.
41. Mnih V, Kavukcuoglu K, Silver D. et al. Human-level control through deep reinforcement learning. *Nature* 2015; **518**:529–33. https://doi.org/10.1038/nature14236.
42. Schulman J, Levine S, Abbeel P. *et al.* Trust region policy optimization. In Bach F. and Blei D. (eds.), *Proceedings of the International Conference on Machine Learning*, Lille, 07–09 July, 2015, pp. 1889–1897. PMLR.
43. Schulman J, Wolski F, Dhariwal P. et al. *Proximal policy optimization algorithms* 2017.
44. *Rocketfuel: An isp topology mapping engine*. University of Washington, Seattle, WA.
45. Knight S, Nguyen H, Falkner N. et al. The internet topology zoo. *IEEE J Sel Areas Commun* 2011; **29**:1765–75. https://doi.org/10.1109/JSAC.2011.111002.
46. Barabási A-L. Network science. *Philos Trans R Soc London, Ser A* 2016; **371**.
47. Chiu C.-C., He T. Stealthy DGoS attack: degrading of service under the watch of network tomography. *IEEE Conference on Computer Communications (INFOCOM)*, Toronto, ON, 06–09 July, 2020, pp. 367–376. IEEE, Piscataway.
48. Chiu C-C, He T. Stealthy DGoS attack: degrading of service under the watch of network tomography. *IEEE/ACM Trans Netw* 2021; **29**:1294–307. https://doi.org/10.1109/TNET.2021.3058230.