Efficient End-to-End Failure Probing Matrix Construction in Data Center Networks

Zequn Jia, Qiang Liu, Ying He, Qianqian Wu, Ren Ping Liu, and Yantao Sun

***Abstract*—Data centers play an essential role in the functioning of modern society. However, failures are unavoidable in data center networks (DCN) and will lead to negative impact on all applications. Therefore, researchers are interested in the rapid detection and localization of failures in DCNs.**

In this paper, we present a theoretical model to analyze the end-to-end failure detection methods in data center networks. Our numerical results verify that the proposed theoretical model is accurate. In addition, we propose an algorithm to construct probing matrices based on an enhanced probing path selection indicator. We also introduce deep reinforcement learning (DRL) method to solve the problem and propose a DRL-based probing matrix construction algorithm. Our experimental results show that both of the proposed algorithms for constructing probing matrices achieve better performance in detection accuracy than existing methods. We discussed different scenarios that the algorithms are applicable to that can improve detection accuracy or construction speed performance.

***Index Terms*—Data center network, deep reinforcement learning, failure detection, probing matrix construction.**

I. Introduction

DATA centers are warehouses that host many servers, provide data processing services, and enable communication between a large number of computing resources [1]. Massively distributed services in data centers like microservices [2], [3] and Spark [4] supports various user-end application. High-speed data center networks connect separated servers together to provide more reliable and scalable computing capability.

With the growth of the DCN scale and the emerging application of new network devices, failures in DCNs have become the norm rather than occasional events. Failures in DCNs usually have more severe consequences than in general networks, from disruption of services to loss of critical data [5]. Meanwhile, the complexity of large-scale DCN management makes it difficult for network administrators to find and locate failures in a timely manner. As a result, rapid detection and localization of failures are crucial in DCN research.

Manuscript received January 5, 2023 revised June 13, 2023; approved for publication by Jeongyeup Paek, Division 3 Editor, June 20, 2023.

Z. Jia is with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, 100044, Beijing, China, and with the School of Electrical and Data Engineering, University of Technology Sydney, Broadway, Ultimo, Sydney, 2007, NSW, Australia. email: zachary@bjtu.edu.cn.

Q. Liu, Q. Wu, and Y. Sun are with the School of Computer and Information Technology, Beijing Jiaotong University, Beijing, 100044, Beijing, China. email: {liuq, qianqianwu0921, ytsun}@bjtu.edu.cn.

Y. He and R. P. Liu are with the School of Electrical and Data Engineering, University of Technology Sydney, Broadway, Ultimo, Sydney, 2007, NSW, Australia. email: {Ying.He, RenPing.Liu}@uts.edu.au.

Q. Liu is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000029

Failure detection is essential to data transmission reliability and system security. Passive detection methods (e.g., SNMP and NetFlow) were utilized in the past, which find and locate failures by periodically collecting port statistics of transmitted bytes and packets. Due to the unpredictability of the data traffic characteristics in DCN, the results of the analysis are imperfect [6], [7]. Nowadays, active probing is a common failure detection technique in DCNs. It will send one or more probes periodically to other nodes to identify any connection problems. This technique can adapt to frequently changing networks. Therefore, active network probing solutions like Microsoft's PingMesh [8] have been developed. By deploying end-to-end probing agents on the edge servers, PingMesh measures the latency between servers and analyzes the results to identify the network's current problem status.

However, there are two main problems with active detection methods such as PingMesh. Firstly, it is challenging for conventional probing methods to use deterministic probing paths due to the abundance of equal-cost multipaths (ECMP) in DCNs. Second, a reasonable set of probing paths must be carefully chosen in order to achieve high detection accuracy with low overhead.

The emergence of new technologies such as software-defined networks, programmable switches, etc., leads to the development of in-band network telemetry (INT)-based failure detection methods [9], [10], [11]. As a result, it is possible to ascertain the probes' forwarding paths, which addresses the issue of too many ECMPs.

Unfortunately, generating a collection of probing paths with high accuracies remains challenging. Most existing research solutions have poor detection accuracy because of the data center's huge scale, high-reliability requirements, and high-application diversity. Moreover, the network load could potentially increase due to the measurement traffic introduced into the network [12]. The reliability of the active failure detection is dependent on the probing path of the agents [13]. PingMesh tries to cover more failures by generating three tiers of ping lists: Intra-racks, inter-racks, and inter-datacenters. These strategies are developed based on the expertise of network specialists. While it is difficult for this approach to generate probing path sets with higher failure-locating accuracy. DeTector [14] tries to improve the accuracy of failure localization by proposing probing matrix construction (PMC) and packet loss localization (PLL) algorithms. While failure detection matrix generation and failure localization accuracy still have growth potential. Besides, there is still limited theoretical analysis on failure detection and localization so far.

In this paper, we propose a formal representation of the

1229-2370/23/$10.00 © 2023 KICS

accuracy of a given failure probing matrix. The design of the probing matrix is then formulated as an optimization problem to minimize the overhead for a given accuracy. We derive an iterative probing matrix construction algorithm with an alternative indicator for evaluating the probing matrices. This algorithm selects a path with the largest accuracy increment from the available detection paths and iterates repeatedly until the accuracy reaches a given value. In doing so, the proposed algorithm will find the optimal path under the current path number limitation at each iteration. Experimental results show that when the detection path is not saturated, the selected probing matrix has higher accuracy than previous methods.

Furthermore, we would like to design an algorithm with higher detection accuracy than conventional heuristics methods. Deep reinforcement learning has generated interest in a variety of research applications. In recent years, the DRL model has been widely used in failure detection, and the automatic feature learning process of its architecture offers excellent potential for solving the problem of failure detection [15]. Therefore, the rational use of reinforcement learning methods can help achieve better failure detection and localization.

So in this paper, we also proposed another DRL-based solution for probing matrix construction. We focus on constructing a probing matrix using the DRL model that performs better than traditional heuristics methods at a given number of probing paths. The experimental results show that our approach is effective, i.e., DRL-based methods are able to find better matrices with higher accuracy than all traditional methods.

Specifically, we have made the following contributions.

- We theoretically analyzed the end-to-end active failure detection and localization method. The failure localization process was analyzed, and we derived the theoretical accuracy for different probing matrices and failure rates.
- We proposed a heuristic approach for probing matrix construction. A novel evaluation was proposed based on the formula above. Then we proposed a heuristic PMC algorithm using the new indicator to get higher failure localization accuracy.
- We propose a DRL-based probing matrix construction method. DRL was incorporated into the PMC problem, and a proximal policy optimization (PPO)-based solution was proposed. The probing matrix generated by this method can achieve higher accuracy.

Our proposed solutions have the potential to make a meaningful impact. By improving the accuracy and efficiency of failure detection, our work can help data center operators enhance network reliability, minimize downtime, and reduce costs. This can eventually enable a seamless user experience for various data center applications and services.

Furthermore, our research can serve as a foundation for future studies in the field of data center network optimization and resilience, ultimately contributing to the development of more robust and efficient data center infrastructures. Our work provides new insights into failure detection in data center networks and has the potential to improve the reliability and availability of data center services.

The rest of this paper is organized as follows. The details of related work are presented in Section II. Section III outlines the system model and defines the probing matrix construction problem. Section IV and Section V describe the motivation and details of our proposed PMC algorithm and DRL-based PMC algorithm, respectively. The evaluation results are presented and discussed in Section VI. Finally, Section VII concludes this article.

II. Related Work

Researchers have been working on failure detection and developed various failure detection techniques for different applications [16]. PTPmesh [17] network monitoring tool uses precision time protocols to estimate one-way latency and quantify packet loss, and it is a viable tool for cloud tenants to obtain network performance statistics. Similar to PingMesh, PTPmesh does not prioritize improving failure localization precision by choosing more appropriate failure detection paths. Pandey *et al.* [18] proposed a novel networked multi-agent system diffusion protocol based on network structure and node priority. Nodes interact with neighboring nodes to diffuse information based on the weighted difference of available resources on the neighboring nodes. While this solution still relies on point-to-point probing, which increases the expense of replacing network devices. Liang *et al.* [19] proposed a heuristic algorithm to optimize the computed backup path, formalizing the backup path selection as an integer programming problem. However, this scheme does not allow for flexible modeling of failure modes, and detection accuracy is inadequate. Xie *et al.* [20] proposed a new chunked matrix completion algorithm to detect link failures. Compared with the existing matrix completion algorithms, the proposed chunking algorithm reduces the sampling complexity. Nevertheless, this method includes a centralized scheduling mechanism to periodically choose the probe paths, which increases the system's costs and complexity. DeTector [14] is a topology-aware failure detection system for DCNs. The authors gave more attention to the significance of failure probing paths and proposed an algorithm to identify suitable probing paths in order to improve the accuracy of failure localization.

However, the theoretical analysis of failure detection is still insufficient, and there is still room for further research in constructing a more efficient and accurate path selection algorithm based on the analysis. We proposed a scheme to address this issue and included a comparison of our approach to deTector in this paper.

With artificial intelligence, failure detection methods based on machine learning have been introduced [21]. Hou *et al.* [22] use an unsupervised two-stage method to detect and characterize the faults in networks. They collect massive time series data of probing results and provide valuable insights on anomaly mining. Mahmood *et al.* [23] proposed a routing method based on reinforcement learning for intelligent failure detection that finds the best route with minimum end-to-end delay. In addition, it also demonstrates that the proposed fault-tolerant strategy contains highly trusted computational capabilities that successfully improve network efficiency and

thus reduce the risk of network failure. Karthik *et al.* [24] proposed an automatic link failure prediction and detection method that improves existing techniques with ML to detect and predict failures with automatic troubleshooting capabilities. Such prediction helps the network to react before a failure occurs. Traffic loss during failures is almost zero and can be achieved by predicting failures in advance. Deep learning methods such as graph neural networks (GNNs) are also applied to failure detection tasks within data centers. Jiang *et al.* [25] proposed APGNN, which extracts features and learns alarm-fault mappings using GNN. Directed graph convolutional neural networks (DGCNN) [26] are applied to locate link failures in data center simulations. The DGCNN model uses the structure of a directed line graph to represent the second-order structure of its underlying graph, providing a new approach to modeling the domain represented by the directed graph.

DRL methods may be more appropriate for this issue than CNNs because it is more adaptive to diverse network topologies. But there are currently rare applicable DRL-based solutions for this problem. We applied deep reinforcement learning to this problem and obtained better results than with conventional approaches.

III. System Overview

In this section, we present the system model for end-to-end failure detection. We then analyze the formulation of failure detection accuracy given the network topology and the failure probing matrix and propose the optimization problem.

A. System Model

Fig. 1 depicts a system model for failure detection via the end-to-end probe mechanism. In this model, agents are deployed on the edge servers of the DCN to send and receive probes, and the results of the probes are analyzed to identify network issues. This model consists of the following components.

- *Probing matrix* is a matrix used to represent a set of failure detection paths in a DCN. Each row of it represents a probe path, whereas each column represents a DCN link.
- *Probing matrix construction (PMC)* algorithm is employed to generate probing matrices. According to the structure of the network and the detection objectives, the algorithm can construct a probing matrix that meets the requirements.
- *Packet loss localization (PLL)* algorithm is designed to analyze the collected probing results to identify and locate failures in the network.
- *Agent* is a lightweight software that is installed on the edge servers of a DCN and consists of senders and receivers. The sender receives the probing matrix from the PMC and constructs probes to transmit to the specified peers according to the probing matrix. Receiving the probes from the sender, the receiver logs the results and reports them to the PLL for analysis routinely.

TABLE I
Table of Notation.

System model	
n	Number of links in the network
$\boldsymbol{L} \in \mathbb{I}^n$	Current link status of the network
k	Number of all available probing paths
$R \in \mathbb{I}^{k\times n}$	Matrix of candidate probing paths
m	Number of probing paths in the probing matrix
$\boldsymbol{P}$	Probing path
$D \in \mathbb{I}^{m\times n}$	The chosen probing matrix
$\mathcal{F}_p$	The PMC algorithm
$\boldsymbol{s}$	Probing result of D
$\boldsymbol{L}'$	Inferred link status
$\mathcal{F}_m$	The PLL algorithm
Problem formulation	
$\mathcal{A}(D)$	Failure probing accuracy
$\mathcal{P}(\boldsymbol{L})$	Power set of $\boldsymbol{L}$
$Q \in \mathbb{I}^{2^n\times n}$	Failure matrix
$\mathcal{S} \in \mathbb{I}^{2^n\times m}$	Probing results of Q
$\hat{S}_{N\times m}$	Compressed $\mathcal{S}$ that removes duplicated rows
N	Number of unique row vectors in $\mathcal{S}$
$U_{N\times 1}$	Mapping matrix for $\hat{S}$ and $\mathcal{S}$
$\hat{Q}_i$	The set of candidate link failure situations of ith probing result.
P	The probability of accurately localizing all link failures can be achieved by the basic-policy PLL algorithm
P'	The probability of accurately localizing all link failures can be achieved by the enhanced-policy PLL algorithm
Probing matrix construction algorithm	
$\mathcal{G}_t$	The set of all indistinguishable failures sets in step t
C	The proposed indicator to evaluate the result of the set division
σ	Standard deviation
β	Identifiability
DRL-based probing matrix construction	
π	Reinforcement policy
S	State vector
A	Action space
W	Termination reward

This architecture is highly compatible with current data center networks, especially the software-defined data center networks, as it leverages the flexibility and programmability of these networks. The PMC and PLL algorithms can be integrated into the SDN controllers. The PMC algorithm, when deployed on the controller, can construct corresponding probing matrices based on the network topology collected by the controller. Similarly, the probing statistics collected in the edge servers can also be aggregated to the controller through the SDN control plane and analyzed by the PLL algorithm in the controller. This integration allows for a more efficient and rapid response to network issues, as well as more effective remediation.

The sender and receiver agents can be implemented as

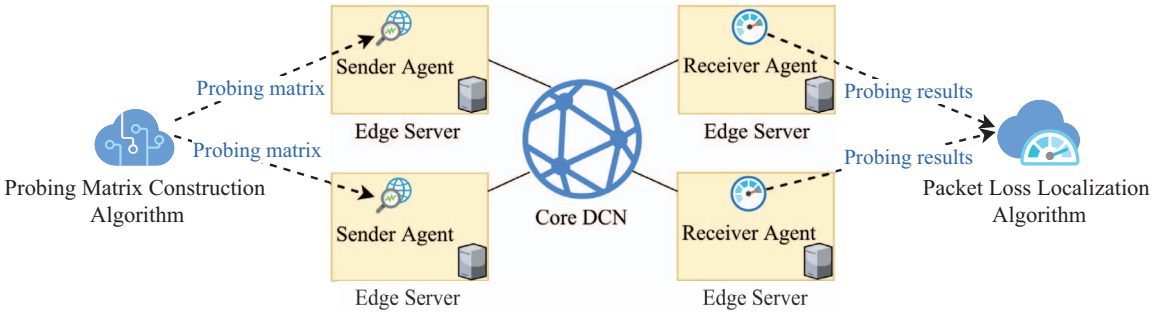


Fig. 1. System architecture of end-to-end probing solutions.

lightweight software modules running on the edge servers of the data center network. As their functions are straightforward, they can be designed to operate with minimal overhead, ensuring that their impact on the data center's primary services is very minimal.

Assuming that there are n links in the whole network topology, we use a binary vector $\boldsymbol{L} \in \mathbb{I}^n$ to indicate the current status of the network. L_i is the ith element of $\boldsymbol{L}$ that represent the status of link i. $L_i = 0$ for link failure, $L_i = 1$ for link is in a normal state.

A forwarding path consists of multiple connected links. Probing paths are forwarding paths in the network, whose source and destination nodes both belong to the set of edge servers. The matrix of candidate probing paths is defined as $R \in \mathbb{I}^{k \times n}$, where k is the number of all available probing paths. Each row of this matrix denotes the related link to this probing path.

The *PMC* algorithm is expected to choose m forwarding paths from candidate paths, i.e., probing paths $\boldsymbol{P} = \{p_1, p_2, \cdots, p_m\}$. And we construct the probing matrix $D \in \mathbb{I}^{m \times n}$ by extracting $R_{p_1}, R_{p_2}, \cdots, R_{p_m}$ from R. In other words, D is a subset of R, i.e.,

$$D = \mathcal{F}_p(R), \tag{1}$$

where $\mathcal{F}_p$ indicates the *PMC* algorithm. The value of D_{ij} is either 0 or 1, implying that link l_j is involved in path p_i.

The probing results will demonstrate whether or not each link in the probing paths works well. A path will be reported as failed if any link in the probing path are in failure. And in reverse, a failed probing path means at least one related link fails. The probing result of D is defined as vector $\boldsymbol{s}$ with length m, where $\boldsymbol{s}_i = 1$ means that path p_i is probed to be working, while $\boldsymbol{s}_i = 0$ means p_i is out-of-order.

Then we use the PLL algorithm to infer L according to $\boldsymbol{s}$. The resulting inferred link status, denoted as L', are shown below.

$$\begin{aligned} L' &= \mathcal{F}_m(R, D, \boldsymbol{s}) \\ &= \mathcal{F}_m(R, \mathcal{F}_p(R), \boldsymbol{s}) \end{aligned} \tag{2}$$

Thus, the key problem of failure probing is to design appropriate *PMC* ($\mathcal{F}_p$) and *PLL* ($\mathcal{F}_m$) algorithms to find the minimized *Diff* (L', L). We could use the ratio of detected link failure number and total link number, i.e., *accuracy*, to indicate *Diff* (L', L).

B. Problem Formulation

In this subsection, we model the detection accuracy. According to the analytical model, one can calculate the detection accuracy value of the failure detection matrix for any network topology under a specific failure localization policy.

Problem: Given topology matrix R and probing matrix D, find failures accuracy $\mathcal{A}(D)$.

The set of all possible link failures is the power set of $\boldsymbol{L}$, i.e., $\mathcal{P}(\boldsymbol{L})$, and $|\mathcal{P}(\boldsymbol{L})| = 2^n$. We put all values of $\boldsymbol{L}$ into the failure matrix $Q \in \mathbb{I}^{2^n \times n}$. Each row of Q is a unique status of $\boldsymbol{L}$ and represents a failure situation of the topology. Q_{ij} means in the ith failure situation, L_j is failed or normal.

As mentioned above, the probing result relies on each involved link, therefore the probing result matrix of all failure situation $\mathcal{S} \in \mathbb{I}^{2^n \times m}$ is a binary matrix as shown below.

$$\mathcal{S} = \neg\mathcal{B}(Q \cdot D^T) \tag{3}$$

Here $\mathcal{B}$ is the binarization function. $\mathcal{S}_{ij}$ is the probing result of the jth probing path under the ith failure situation. $S_{ij} = 1$ means that all links in p_j are reported normal, while $S_{ij} = 0$ means one or more links are out of order.

We now turn our attention to the failure localization algorithm $\mathcal{F}_m$ and present its theoretical accuracy formulation. The algorithm $\mathcal{F}_m$ is designed as a reverse mapping that retrieves the failure situation from the probing results, as depicted in (2). Due to the constraints of probing complexity, this mapping is not always one-to-one. Consequently, multiple failures may lead to the same probing outcome for a given probing result. In such cases, there could be several candidate failure situations $\boldsymbol{Q}_i$ that satisfy (3), making it impossible for the PLL algorithm to determine the expected result.

We assume that all link failures in the network follow a binomial distribution with probability p, i.e., $X \sim B(n, p)$. Let $\hat{S}_{N \times m}$ represent the compressed $\mathcal{S}$ matrix with duplicate rows removed, where N denotes the number of unique row vectors in $\mathcal{S}$. We then define the mapping matrix $U_{N \times 1}$ as follows:

$$U = \{U_i\},\ \forall U_i \in U, U_i = |\{\mathcal{S}_j\}|, \text{where } \forall j, \mathcal{S}_j = \hat{\boldsymbol{S}}_{\boldsymbol{i}}.$$

In this case, U_i corresponds to the number of times the row vector $\hat{\boldsymbol{S}}_{\boldsymbol{i}}$ appears in $\mathcal{S}$. We also define $\hat{Q}_i$ as the set of candidate link failure situations, meaning that all elements within it result in the same probing outcome $\hat{\boldsymbol{S}}_{\boldsymbol{i}}$:

$$\hat{Q}_i = \{\boldsymbol{Q}_{\boldsymbol{j}}\}, \text{where } j \in [1, U_i], \text{ and } \forall j, \boldsymbol{\mathcal{S}_j} = \hat{\boldsymbol{S}}\boldsymbol{i}.$$

Each $\hat{\boldsymbol{Q}}\boldsymbol{ij}$ is a vector representing a failure situation.

We now consider the accuracy of inferring $\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}$ from $\boldsymbol{Si}$. If the inferred $\hat{\boldsymbol{Q}}\boldsymbol{ij}$ is chosen randomly from the candidate failure situations $\hat{\boldsymbol{Q}}_{\boldsymbol{i}}$, the probability that the algorithm can correctly identify all failed links is given by the following expression, where ΣR_j represents the length of the probing paths.

$$P = \sum_{i=1}^{N}\left(\frac{1}{U_i}\cdot\sum_{j=1}^{U_i}\left(p^{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\cdot(1-p)^{\Sigma R_j-\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\right)\right) \tag{4}$$

But we would like to find a better policy for choosing $\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}$ from $\hat{\boldsymbol{Q}}_{\boldsymbol{i}}$ to improve the detection accuracy. Therefore, we proposed to choose the $\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}$ with the minimum number of failure occurrences from $\hat{\boldsymbol{Q}}_{\boldsymbol{i}}$, i.e., $j^* = \arg\min_j \Sigma\hat{Q}_{ij}$. Then the probability that the algorithm can identify all the failed links become as follows.

$$\begin{aligned} P' &= \sum_{i=1}^{N}\left(p^{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}}\cdot(1-p)^{\Sigma R_j-\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}}\right) \\ j^* &= \text{argmin}\ \Sigma\hat{Q}_{ij} \end{aligned} \tag{5}$$

Here, p is the probability of link failure, this value is usually less than 20% in commercial data center networks [27]. Therefore we could assume that $p < 0.5$. It could be proved that (5) is higher than (4) for $p < 0.5$ as follows.

1) Let

$$P_{1j} = p^{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\cdot(1-p)^{\Sigma R_j-\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}},$$

$$P_2 = p^{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\cdot(1-p)^{\Sigma R_j-\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}.$$

To prove that (5) > (4), we need to prove that for a given U_i, $P_2 > avg(P_{1j})$.

2) Let

$$K = \frac{P_{1j}}{P_2} = \left(\frac{p}{1-p}\right)^{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}-\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}}.$$

In (5), we let $j^* = \text{argmin}\ \Sigma\hat{Q}_{ij}$, and as a result, $\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}} - \Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}} \geq 0,\ \forall j \in U_i$.

3) We get $K < 1$ when $p < 0.5$, i.e., $P_{1j} < P_2, \forall j \in U_i$ That means $avg(P_{1j}) < P_2$, and the proof is finished.

But the probabilities in (5) means the located failures situations $\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}$ are exactly the same with the actual failures. Whereas the accuracy only counts the number of located failure links. Therefore we also present the probability for different statistic calibers.

$$P(f) = \sum_{i=1}^{N}\sum_{j=1}^{U_i}\left(p^{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\cdot(1-p)^{\Sigma R_j-\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\right)\cdot\frac{f(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}},\hat{\boldsymbol{Q}}_{\boldsymbol{ij}})}{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}, \tag{6}$$

where $j^* = \text{argmin}\ \Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}$, and f indicates the statistical functions for different statistic calibers. There are different $f(\hat{\boldsymbol{Q}}_{\boldsymbol{ik}},\hat{\boldsymbol{Q}}_{\boldsymbol{ij}})$ in (7), (8) and (9) for accuracy, false negative rate (FNR) and false positive rate (FPR) .

For $P(\frac{\text{\# of detected failures}}{\text{\# of total failures}})$, i.e., the accuracy $\mathcal{A}$,

$$f(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}},\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}) = \Sigma\left(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}\bigcap\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}\right). \tag{7}$$

For $P(\frac{\text{\# of undetected failures}}{\text{\# of total failures}})$, i.e., FNR,

$$f(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}},\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}) = \Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}} - \Sigma\left(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}\bigcap\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}\right). \tag{8}$$

For $P(\frac{\text{\# of misreported normal link}}{\text{\# of total failures}})$, i.e., FPR,

$$f(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}},\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}) = \Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}} - \Sigma\left(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}\bigcap\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}\right). \tag{9}$$

The size of the probing matrix D affects how much probing overhead is required. This means more probing agents need to be deployed in DCNs and more extra bandwidth is occupied. Therefore we would like to achieve a given detection accuracy $\mathcal{A}$ with minimum probing overhead, i.e., the smallest number of probing paths. We model the optimization problem as shown in (10).

$$\begin{aligned} \min\quad & \text{rows}(D) \\ \text{s.t.}\quad & \sum_{i=1}^{N}\sum_{j=1}^{U_i}\left(p^{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\cdot(1-p)^{\Sigma R_j-\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}}\right)\cdot\frac{f(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}},\hat{\boldsymbol{Q}}_{\boldsymbol{ij}})}{\Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}} \geq \mathcal{A}, \\ & j^* = \text{argmin}\ \Sigma\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}, \\ & f(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}},\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}) = \Sigma\left(\hat{\boldsymbol{Q}}_{\boldsymbol{ij^*}}\bigcap\hat{\boldsymbol{Q}}_{\boldsymbol{ij}}\right) \end{aligned} \tag{10}$$

IV. Probing Matrix Construction Algorithm

The above optimization problem of failure detection via round-trip probing is NP-hard, as proven in [28]. In order to obtain the available probing matrix in a reasonable time, we propose an iterative probing matrix calculation algorithm, which chooses one path from the sub-optimal probing paths that maximizes the accuracy increment $\Delta\mathcal{A}$ and iterates until the accuracy reaches the target.

However it remains a complex problem to calculate the accuracy of the new probing matrix P' after each iteration. According to (6), the complexity of computing the probing accuracy for a given probing matrix will be $O(2^n)$, where n is the number of physical links. While the scale of a modern data center network is enormous, e.g., Microsoft hosts over one million servers in over 100 data centers globally [29]. Therefore, we need an approximate indicator with lower enough computational complexity to reflect the accuracy variation.

As we mentioned in the previous section, one given state

$\mathcal{S}$ may correspond to multiple $\boldsymbol{Q}_i$ values. In (6), there are multiple probing methods to choose a value from multiple possible values $\hat{\boldsymbol{Q}}_{\boldsymbol{i}}$ as our predicted value. For both probing methods 1 and 2, the less optional values, i.e., $|\hat{\boldsymbol{Q}}_{\boldsymbol{i}}|$, the higher the probability that the predicted result is the same as the actual result, which means higher probing accuracy.

Define $\mathcal{G}_t$ as the set of all indistinguishable failures sets in step t.

$$\mathcal{G}_t = \{G_1, G_2, \cdots, G_k\} \tag{11}$$

At the start of constructing the probing matrix, we put all possible failures into $\mathcal{G}$. After a new probing path p is added to the probing matrix D, the set can be divided according to whether the link in the failure exists on the path. The set division from step t to $t+1$ is defined as follows.

$$\mathcal{G}_{t+1} = \{G_1', G_1'', G_2', G_2'', \cdots, G_k', G_k''\}, \tag{12}$$

where

$$G_i' = \{g | g \in G_i \text{ and } g \times p^T > 0\},$$
$$G_i'' = \{g | g \in G_i \text{ and } g \times p^T = 0\}.$$

We can increase t by adding more paths to D, until the accuracy constraint is met. Here each G_i is corresponding to a $\hat{\boldsymbol{Q}}_{\boldsymbol{i}}$. We would like to decrease $|\hat{\boldsymbol{Q}}_{\boldsymbol{i}}|$ by finding a more reasonable solution of set division, i.e., constructing the probing matrix.

On the basis of the above observations, we propose a new indicator C to evaluate the result of set division. There are two key requirements for set division: 1) each G_i should contain as fewer elements as possible to make failures easier to identify; 2) each $|G_i|$ should be evenly distributed to avoid lower accuracy due to too many failures reside in some particular G_i. Therefore the proposed indicator C is defined as follows.

$$C(\mathcal{G}) = \frac{1 + \sigma(\mathcal{L}(\mathcal{G}))}{|\mathcal{G}|}, \tag{13}$$

where $\mathcal{L}(\mathcal{G}) = \{|G_i|\}, \forall G_i \in \mathcal{G}$ and σ is the standard deviation of it. Smaller C means fewer elements in each subset of $\mathcal{G}$ and/or elements are more balanced divided, eventually leading to better accuracy.

However, the number of failures initially being divided i.e., $\sum \mathcal{L}$, is still up to 2^n. Considering the identifiability mentioned in [14], we can use the failure set of maximum β simultaneous failures to reduce the complexity. As a result, the total number of failures will fall to $\sum_{i=1}^{\beta} \binom{n}{i}$.

The proposed PMC algorithm is shown in Algorithm 1, where C is the criterion when adding a new probing path. This algorithm is a greedy method, which will iterate until given number of probing paths are selected or a given accuracy is achieved. At each iteration, the algorithm calculates the C for each candidate path after adding it to the current set of detected paths, and selects the path that minimizes C to the set. The overall time complexity will be $O(m \times k \times n)$, where m is the size of probing matrix and k is the number of candidate probing paths.

C will remain valid until set division of $\mathcal{G}$ is complete, i.e., each subsets in $\mathcal{G}$ contains only one element. As Algorithm 1 is an iterative algorithm, in each iteration, the algorithm evaluates the candidate set division solutions according to C. Thus it is necessary to ensure that C is valid, i.e., $|\mathcal{G}| \leq \mathcal{L}(\mathcal{G})$. In the subsequent experiments, we found that generally when $\beta = 1$, $|\mathcal{G}|$ will reach $\mathcal{L}(\mathcal{G})$ too fast, making the iteration fail to continue. While $\beta = 2$ or 3 is large enough to achieve the expected accuracy.

Algorithm 1 Proposed probing matrix contruction algorithm

Input: $R, \beta, \mathcal{A}$
Output: Probing matrix $P_{m\times n}$
  $Q \leftarrow LINKOR(R, \beta)$
  $\mathcal{G} \leftarrow \{Q\}$
  $D = \Phi$
  while $Accuracy(P) < \mathcal{A}$ **do**
    for $p \in R$ **do**
      $\mathcal{G}_p \leftarrow$ divide $\mathcal{G}$ by p
      $C_p \leftarrow \frac{1+\sigma(\mathcal{L}(\mathcal{G}))}{|\mathcal{G}|}$
    end for
    $p \leftarrow \text{argmin}_{p' \in R}\, C_{p'}$
    $D \leftarrow D \cup \{p\}$
    $R \leftarrow R/\{p\}$
    $\mathcal{G} \leftarrow \mathcal{G}_p$
  end while

V. DRL-based Probing Matrix Construction

Algorithm 1 focuses on finding a better probing path in each step. Therefore, the final probing matrix is likely to be the local optimum and there is no guarantee that the global optimum will be obtained.

But in some specific scenarios that are extremely sensitive to overhead, we would like to find a probing matrix that is closer to the optimal result, allowing higher system accuracy on the same probing matrix size. Reinforcement learning methods have achieved remarkable results in solving complex combinatorial optimization problems [30]. The PPO algorithm [31] has been reported to be more efficient and stable compared to other reinforcement learning algorithms such as DQN in solving these combinatorial optimization problems [32]. Therefore, we apply the PPO algorithm to the probing path construction problem. We provide feedback to the policy model to enable it to choose more reasonable failure detection paths. The PPO algorithm uses an actor-critic model and limits the change of the policy through the hyperparameter ϵ so that the update of the policy is within a limited range, making it more stable.

The above probing matrix construction problem is abstracted into an environment in order to employ reinforcement learning algorithms. To model this problem with a DRL model, we use the selected set of probing paths as the output of the reinforcement learning model, i.e., actions. But the solution space of this problem is exponential. Thus the action space will become overwhelmingly large, and it will grow rapidly with the increase of the network size. Excessive action space will have a negative impact on the performance of the DRL model, which is not conducive to find an appropriate policy [33].

In order to manage and reduce the complexity of the action space, we have designed our reinforcement learning

environment to treat the probing matrix construction problem as an episode, which is divided into multiple steps. This approach allows us to break down the problem into smaller, more manageable subproblems. At each step, the DRL policy selects a single probing path and adds it to the existing probing path set. By doing so, we effectively reduce the action space at each step, as the DRL model only needs to consider a single probing path rather than the entire solution space. This step-by-step approach not only simplifies the action space but also enables the DRL model to learn more efficiently, as it can focus on making incremental improvements to the probing path set. As a result, the performance of the DRL model is less likely to be negatively impacted by the exponential growth of the action space, allowing it to find an appropriate policy more effectively.

Additionally, it is worth noting that while this method of reducing complexity may have some potential negative impact on the final results, our experimental results demonstrate that this approach still achieves optimization outcomes that surpass those of traditional methods. This indicates that the trade-off between managing the complexity of the action space and the quality of the final solution is well-balanced in our proposed method. By effectively handling the action space complexity, our DRL model is able to learn more efficiently and ultimately outperform conventional techniques in solving the probing matrix construction problem.

State space: The state space includes the information that the DRL policy needs to know when making path selections in each step. In this environment, we feed the set of selected probing paths as the state to the policy model π. The policy model π will choose new probe paths to be added to the set based on the current state. Therefore, we define the state vector $S \in \mathbb{I}^{1\times k}$ as follows, where S_i indicates whether the candidate probing path i exists in the set of detection paths and k is the number of all candidate probing paths.

$$S = (S_1, S_2, \cdots, S_k),\ S_i \in \{0, 1\}$$

Action Space: The action space of each step is the new probing path selected by policy π in the current step. We directly use the ordinal number of the probe path to represent the corresponding path. Formally, the action space A can be defined as a discrete integer.

$$A = i,\ i \in \{1, 2, \cdots, m\}$$

Reward: The reward provides feedback to the policy π and must be carefully designed to ensure that qualified probing paths are chosen. We divide the reward in the whole environment into two parts: 1) the reward after each action step is completed (step reward), and 2) the reward at the end of an episode when the compliant set of probing paths is obtained (termination reward).

- **Step reward:** After each step is completed, we first penalize the repeatedly selected path, as it provides no help to the probing. We directly give a large negative reward and terminate this episode. This penalty discourages the DRL policy π from selecting the same path multiple times. For each distinct path, we also give a small penalty to encourage the DRL policy π to continue exploring until the size of the probing matrix meets the requirement. This step reward design aims to balance exploration and exploitation during the learning process.
- **Termination reward:** For the reward at the end of each episode, we use the proposed indicator C. We design the reward W to increase in proportion to the performance, as follows:

$$W = \frac{1}{C} = \frac{|\mathcal{G}|}{1 + \sigma(\mathcal{L}(\mathcal{G}))}. \tag{14}$$

This termination reward design encourages the DRL policy to select the set of probing paths with a higher final reward W. By using the indicator C, we ensure that the reward is directly related to the detection accuracy, thus guiding the policy towards better solutions.

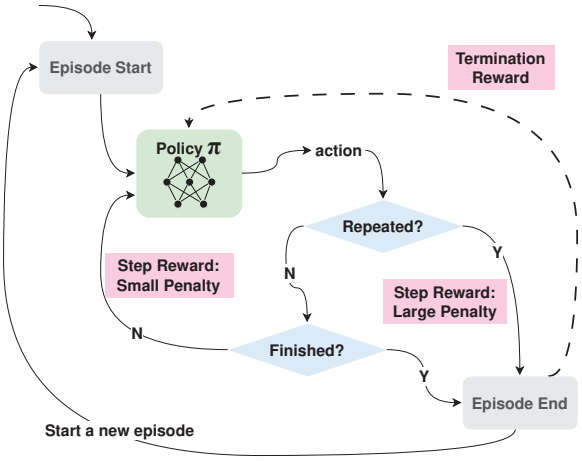


Fig. 2. The training process of the PPO-based PMC algorithm.

As shown in Fig. 2, the reward mechanisms are adopted to train the DRL policy. We do not include the termination reward W in the intermediate step and introduce a discount, as we are more concerned about the final result rather than the intermediate process for this model. This design choice helps the policy avoid converging to a local optimum, as might happen with a greedy algorithm.

VI. Evaluation and Discussion

A. Experimental Setup

To evaluate the performance of the two algorithms proposed in this paper, we constructed simulation platforms for experimental assessment.

First, we developed a Python-based simulation platform to assess the detection accuracy of failure probing matrices produced by the PMC algorithm. This platform initially generates a fat-tree network topology with faults, taking into account the specified link failure rate q. Subsequently, it executes the provided PMC algorithm to create the corresponding probing matrix and deploys the relevant agents to the appropriate edge servers. Then detection agents run on the edge server nodes, transmitting probing data. The PLL algorithm from [14] is employed for failure localization using collected probing data.

Finally, the identified link failures are compared with the actual link failures to compute metrics such as detection accuracy.

Additionally, to emulate a more realistic probing topology, we constructed a similar experimental environment utilizing the ns-3 network simulator. The probe transmission and reception processes are simulated on the edge servers within the network topology established in ns-3. The forwarding model of ns-3 more closely resembles the actual data center network nodes, thereby providing a better representation of our algorithms' performance in real-world scenarios.

We compare our proposed algorithms with deTector [14] as it is currently one of the leading algorithms for host-based failure localization. It builds upon the well-known PingMesh [8]. Our work improves upon deTector in several critical aspects, including theoretical analysis, a superior indicator, and further more, a DRL-based solution. By comparing against deTector, we aim to highlight the strengths of our proposed algorithms.

The fat-tree topology [34] is one of the most prevalent network structures in data center networks, with many commercial data centers employing fat-tree and its variants [35]. It utilizes a hierarchical structure where every level is fully interconnected. This design guarantees multiple pathways between any two nodes within the network, thereby circumventing bottlenecks and augmenting overall network performance. Therefore, we used a 4-ary fat-tree network topology in our experiments, consisting of a total of 36 nodes.

The simulation platform and other related experiments in this paper are run on a workstation with an Intel Core i7-8700 processor and 32 GB of memory.

B. Theoretical Accuracy

We first investigate the consistency between the proposed accuracy model and the simulation accuracy results. The experiments are conducted in a 4-ary fat-tree network topology with two pods, where 128 candidate probing paths are involved in this network. Algorithm 1 was executed with different link failure rates to generate the probing matrices at each given link failure rate. Then we construct probing matrices with different sizes from 1 to 20, and run the PLL algorithm respectively based on them. The theoretical and simulation accuracies are shown in Fig. 3.

It is shown that the simulated values (detected rates, undetected rates, and misreported rates) are closely approximated to the theoretical values given by (6) with different link failure rates ($q = 0.05$ and $q = 0.1$). This indicates our proposed model in Section III is able to accurately reflect the failure detection accuracy given the network topology and probing matrix.

C. Heuristic PMC Algorithm

We propose an indicator C in (13) as an alternative to the absolute accuracy with high time complexity. We first compare the probing accuracies using different evaluation indicators ($\Delta\mathcal{A}$ and C) with our proposed Algorithm 1.

As shown in Fig. 4, We ran the failure detection algorithm using $\Delta\mathcal{A}$ and C as evaluation metrics respectively. It is shown that with the same link failure rate, the accuracies produced by the two approaches are close to each other. This demonstrates that the indicator C we proposed for choosing better probing paths during each iteration is reasonable and can depict the contribution of each probing path to the final accuracy.

We evaluate the theoretical values in (7), (8), and (9) of the proposed PMC algorithm and deTector. The results shown in Fig. 5 demonstrate that our proposed PMC algorithm outperforms the PMC algorithm in deTector. Specifically, the proposed algorithm gains higher detected rates as well as lower undetected and misreported rates.

In addition, we evaluate the simulated failure detection accuracies of our algorithm. The experiments are conducted in both numerical and ns-3 simulation environments with deTector as a comparison.

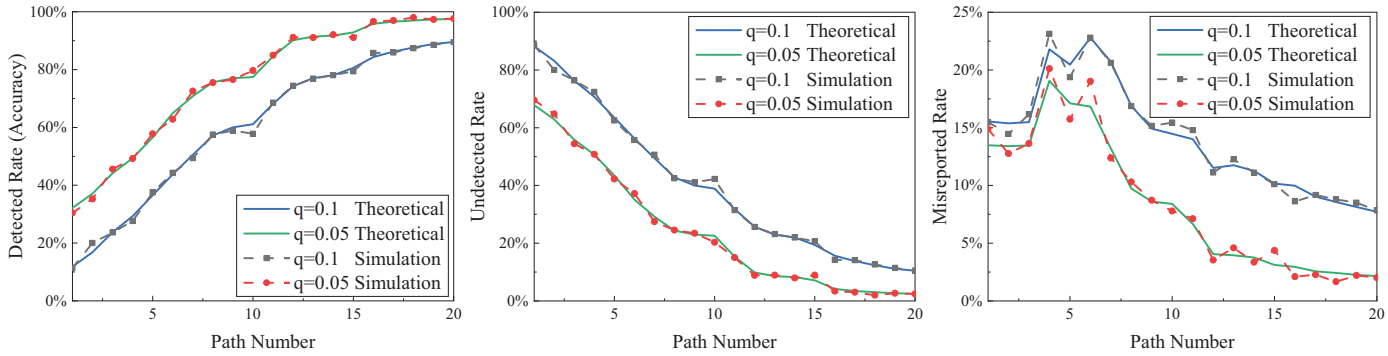
The PLL algorithm proposed in [14] is a failure localization algorithm that is efficient and could be deployed in data center networks. Therefore, we also performed an accuracy evaluation using the PLL algorithm. We generated the probing matrices using the same parameters and sent their probing results to the PLL algorithm as inputs. The accuracy of the link failures inferred by the PLL algorithm according to the probing results is shown in Fig. 6(a).

To validate the actual effectiveness of the algorithm in real data center networks, we implemented the whole probing matrix construction, probing agents and failure localization algorithms in a popular network simulator ns-3. We use ns-3 to generate the experimental network topology, and obtain the corresponding probing results on the receiver agents by sending actual probe packets from the sender agents. The probing results are aggregated at the controller to calculate the failure localization using the PLL algorithm. Results in Fig. 6(b) indicate that the proposed algorithm also gains better accuracies in the ns-3 simulator than the deTector algorithm.

These several experimental data above show that our proposed algorithm has a significant improvement in the accuracy of failure detection compared with deTector. Our method's accuracy increases more rapidly with the number of probing paths and can be accomplished with fewer probing paths compared with deTector, which could reduce the overall probing overhead.

We can also find that with the different values of β, the accuracies of $\beta = 2$ and $\beta = 3$ are almost the same in the first half of the curve, whereas the curve of $\beta = 2$ tends to flatten out while $\beta = 3$ continues to grow in the rest. This is due to the failures of $\beta = 2$ is already divided completely when the path number is greater than 15, i.e., there is only one element in each set. As a result, the set number cannot be used to enhance the subsequent probing path selection. While there are more elements in $\beta = 3$ and they can keep being split as the number of paths increases.

This provides insight for selecting an appropriate β value. The value of β can be determined in accordance with the accuracy requirement. With a given accuracy, a smaller β is preferred to reduce the overhead of deploying agents and probing bandwidth.



(a) Detected failure rate (Accuracy) (b) Undetected failure rate (c) Misreported failure rate

Fig. 3. The comparison of theoretical and simulation values of (6) with different link failure rates.

TABLE II
COMPARISON OF ACCURACIES OF DIFFERENT SCHEMES IN DIFFERENT β AND PROBING PATH NUMBER PARAMETERS.

Schemes	**Accuracy (β = 2, path=10)**	**Accuracy (β = 3, path=15)**
deTector PMC algorithm [14]	63.68%	83.19%
Proposed PMC algorithm in Section IV	72.13%	85.87%
Proposed DRL-based PMC algorithm in Section V	73.01%	86.68%

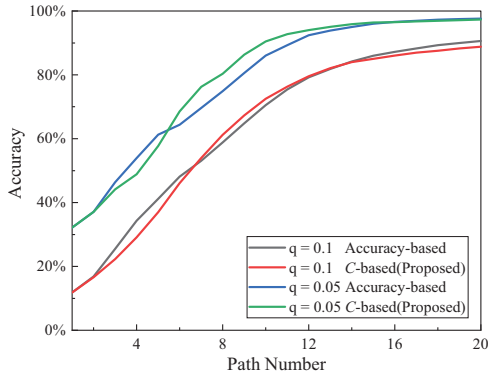


Fig. 4. The accuracy results when choosing different indicators in Algorithm 1.

D. DRL-based PMC Algorithm

We also evaluated the accuracy of the failure probing matrix obtained using the DRL-based method. As the main objective of using reinforcement learning methods is to improve the final accuracy as closely to the global optimal as possible, the reinforcement learning method we designed focuses on improving the final C, and not on the values in the intermediate process.

The comparison of the accuracies of the three methods is shown in table Table II. This table demonstrates that our reinforcement learning scheme is able to find a probing matrix that is closer to the optimal solution than the heuristic algorithm and deTector with the same given number of probing paths and β value.

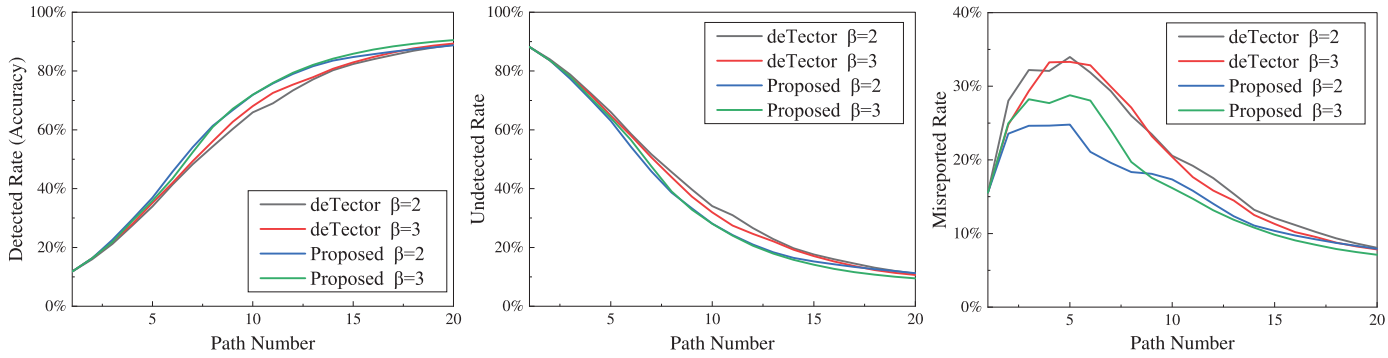
It is worth noting that although the DRL-based method can find a better probing matrix, it takes longer training time than the greedy heuristic algorithms. However, considering the fat-tree network topology with 128 candidate paths as an example, the complexity of finding the optimal probing matrix with 10 paths is $\binom{128}{10} \approx 2 \times 10^{14}$, while the complexity of selecting 15 or 20 paths is even higher. This means the DRL-based algorithm is able to find a suboptimal matrix that is closer to the optimal one after several hours of training even for such a complex problem.

To further justify our choice of the PPO algorithm, we conducted a comparative experiment between PPO and DQN, two popular DRL algorithms. As shown in Fig. 7, the PPO algorithm demonstrates faster growth, smaller fluctuations, and ultimately achieves a higher normalized reward compared to DQN. In contrast, the DQN algorithm exhibits slower growth, larger fluctuations, and its final reward is only about 60% of the PPO's reward. These results provide empirical evidence supporting our choice of the PPO algorithm for solving the complex combinatorial optimization problem in our study. The superior performance of PPO in terms of both convergence speed and final reward makes it a more suitable choice for our application.

The entire training process of 1000K PPO steps took approximately 2 hours. The training was performed on an Intel Core i7-8700 workstation with 32 GB of RAM, utilizing only the CPU. Because reinforcement learning typically employs smaller neural networks, so the benefits of using a GPU would be counterbalanced by the communication overhead between the CPU and GPU.

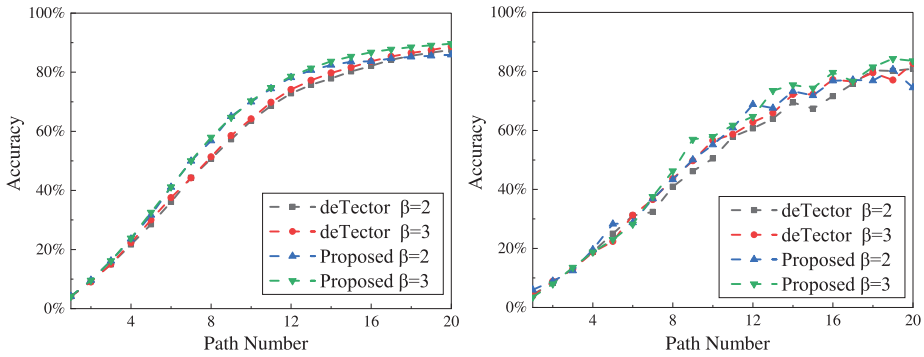
E. Limitations and Future Work

Our experimental results demonstrate the effectiveness of the proposed solutions. The results show that our theoretical model is consistent with actual scenarios, and both our heuristic algorithm and DRL-based PMC algorithm outperform other methods.



(a) Detected failure rate (Accuracy) (b) Undetected failure rate (c) Misreported failure rate

Fig. 5. Theoretical values of different methods with different β.



(a) Accuracies in the PLL numerical simulation (b) Accuracies in the ns-3 simulator

Fig. 6. Accuracies of generated probing matrices using different methods with different βs.

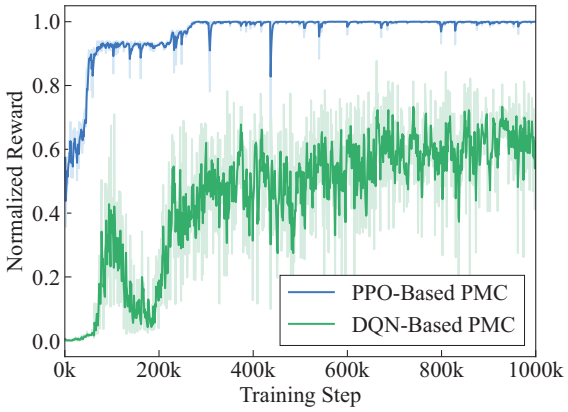


Fig. 7. Training performance comparison of PPO and DQN-based PMC algorithms.

Nevertheless, our algorithms exhibit certain limitations and drawbacks. The DRL-based algorithm delivers superior detection performance, achieving higher accuracy with equivalent probing overhead or reduced probing overhead with the same target accuracy. In contrast, the heuristic algorithm provides expedited construction speed but slightly lower detection performance. As a result, these two algorithms cater to distinct application scenarios. When probing overhead is highly sensitive and a smaller probing matrix is desired, the DRL-based algorithm should be employed given the same target accuracy. On the other hand, when the construction speed of the probing matrix is prioritized and its size is less critical, the heuristic algorithm should be selected.

Additionally, scalability remains a significant limitation for both algorithms. As the size of data center networks increases, the matrix construction speed of these methods still has considerable constraints. Further research is warranted in this area.

Fortunately, recent studies such as Bernardez *et al.* [36] suggest that trained GNNs can yield favorable results on unseen topologies in the training data, indicating a degree of generality. Our work establishes the groundwork for utilizing deep reinforcement learning in failure probing matrix construction. Building upon this foundation, it might be feasible to develop a universal reinforcement learning PMC solution employing GNN-based reinforcement learning techniques. This approach would enable the construction of probing matrices on any network topology following a single training session. We leave this exploration for future work.

Besides, our proposed algorithms are anticipated to perform optimally in data centers with physical infrastructure instead of virtualized networks. This is mainly due to physical data center networks' regular and predictable topologies. Conversely, virtualized data center networks present a more diverse range of topologies. These virtualized networks may not always offer multiple equivalent paths, which could potentially diminish the effectiveness of our solution. We acknowledge that further

research is necessary to optimize probing matrix construction in data centers with virtual infrastructure.

VII. Conclusion

This paper firstly models the end-to-end failure detection method and presents the theoretical accuracy of failure detection under a given network topology and probing matrix. Secondly, we design an alternative indicator for evaluating the effectiveness of a probing matrix, and propose a new probing matrix construction algorithm. The evaluation results show that the proposed algorithm using the new indicator focuses on improving the accuracy at each step of selection. As a result, the selected probing matrix can achieve the target accuracy sooner than previous methods. Furthermore, we propose to use the deep reinforcement learning algorithm to construct probing matrices. We provide feedback to the reinforcement learning policy via balanced set division result. The proposed algorithms are evaluated in both numerical and ns-3 simulations. Experimental results show that the DRL-based probing matrix construction is able to find a probing matrix that is closer to the optimal. Finally, we discussed the different application scenarios of the two proposed methods that are aiming to achieve better detection accuracy or construction speed.

References

[1] K. Wang, Q. Zhou, S. Guo, and J. Luo, "Cluster frameworks for efficient scheduling and resource allocation in data center networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3560–3580, 2018.

[2] I. K. Aksakalli, T. Çelik, A. B. Can, and B. Tekinerdoğan, "Deployment and communication patterns in microservice architectures: A systematic literature review," *J. Syst. Software*, vol. 180, p. 111014, 2021.

[3] T. Wang, W. Zhang, J. Xu, and Z. Gu, "Workflow-aware automatic fault diagnosis for microservice-based applications with statistics," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 4, pp. 2350–2363, 2020.

[4] Y. Xia *et al.*, "A tale of two topologies: Exploring convertible data center network architectures with flat-tree," in *Proc. ACM SIGCOMM*, Aug. 2017.

[5] H. Wang *et al.*, "Grano: Interactive graph-based root cause analysis for cloud-native distributed data platform," *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 1942–1945, 2019.

[6] Z. Chen, J. Xu, T. Peng, and C. Yang, "Graph convolutional network-based method for fault diagnosis using a hybrid of measurement and prior knowledge," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9157–9169, 2022.

[7] X. Qi, J. Li, Z. Wang, and L. Liu, "Probabilistic probe selection algorithm for fault diagnosis in communication networks," *Comput. Netw.*, vol. 198, p. 108365, 2021.

[8] C. Guo *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proc. ACM SIGCOMM*, Aug. 2015.

[9] C. Kim *et al.*, "In-band network telemetry via programmable dataplanes," in *Proc. ACM SIGCOMM*, Aug. 2015.

[10] S. Narayana *et al.*, "Language-directed hardware design for network performance monitoring," in *Proc. ACM SIGCOMM*, Aug. 2017.

[11] D. Bhamare *et al.*, "Intopt: In-band network telemetry optimization framework to monitor network slices using p4," *Comput. Netw.*, vol. 216, p. 109214, 2022.

[12] X. Guo *et al.*, "Graph-based trace analysis for microservice architecture understanding and problem diagnosis," in *Proc. ACM ESEC/FSE*, Nov. 2020.

[13] T. Pan *et al.*, "Int-probe: Lightweight in-band network-wide telemetry with stationary probes," in *Proc. IEEE ICDCS*, Jul. 2021.

[14] Y. Peng *et al.*, "deTector: A topology-aware monitoring system for data center networks," in *Proc. USENIX ATC*, Jul. 2017.

[15] S. R. Saufi, Z. A. B. Ahmad, M. S. Leong, and M. H. Lim, "Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review," *IEEE Access*, vol. 7, pp. 122644–122662, 2019.

[16] X. Zhou *et al.*, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Trans. Softw. Eng.*, vol. 47, no. 2, pp. 243–260, 2021.

[17] D. A. Popescu and A. W. Moore, "Measuring network conditions in data centers using the precision time protocol," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 3753–3770, 2021.

[18] P. K. Pandey, B. Adhikari, and S. Chakraborty, "A diffusion protocol for detection of link failure and utilization of resources in multi-agent systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1493–1507, 2020.

[19] D. Liang *et al.*, "Low interruption ratio link fault recovery scheme for data plane in software-defined networks," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 6, pp. 3806–3819, 2021.

[20] K. Xie, J. Tian, G. Xie, G. Zhang, and D. Zhang, "Low cost sparse network monitoring based on block matrix completion," in *Proc. IEEE INFOCOM* , May 2021.

[21] P. Foroughi, F. Brockners, and J.-L. Rougier, "Adt: Ai-driven network telemetry processing on routers," *Comput. Netw.*, vol. 220, p. 109474, 2023.

[22] B. Hou, C. Hou, T. Zhou, Z. Cai, and F. Liu, "Detection and characterization of network anomalies in large-scale rtt time series," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 793–806, 2021.

[23] T. Mahmood *et al.*, "An intelligent fault detection approach based on reinforcement learning system in wireless sensor network," *J. Supercomputing*, vol. 78, no. 3, pp. 3646–3675, 2022.

[24] K. B. Harichandra Babu, X. Huang, A. Anand, and M. Mishra, "Method and apparatus for automated link failure detection in networks using machine learning," Technical Disclosure Commons, Tech. Rep., Jun. 2020.

[25] W. Jiang and Y. Bai, "Apgnn : Alarm propagation graph neural network for fault detection and alarm root cause analysis," *Comput. Netw.*, vol. 220, p. 109485, 2023.

[26] M. Kenning, J. Deng, M. Edwards, and X. Xie, "A directed graph convolutional neural network for edge-structured signals in link-fault detection," *Pattern Recognition Lett.*, vol. 153, pp. 100–106, 2022.

[27] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM*, Aug. 2011.

[28] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," in *Proc. IEEE INFOCOM*, Mar.–Apr. 2003.

[29] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 640–656, 2017.

[30] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Comput. Operations Research*, vol. 134, p. 105400, 2021.

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

[32] N. Saha, M. Zangooei, M. Golkarifard, and R. Boutaba, "Deep reinforcement learning approaches to network slice scaling and placement: A survey," *IEEE Commun. Mag.*, vol. 61, no. 2, pp. 82–87, 2023.

[33] S. Manna *et al.*, "Learning in continuous action space for developing high dimensional potential energy models," *Nature Commun.*, vol. 13, no. 1, p. 368, 2022.

[34] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, Aug. 2008.

[35] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in *Proc. IEEE OIC*, May 2013.

[36] G. Bernardez *et al.*, "Is machine learning ready for traffic engineering optimization?" in *Proc. IEEE ICNP*, Nov. 2021.

Zequn Jia received his B.E. degree in Computer Science and Technology from Beijing Jiaotong University. He is currently pursuing his PhD degree in School of Computer and Information Technology, Beijing Jiaotong University, China. And he is also a dual degree student at University of Technology Sydney. His research interests include data center networking and software-defined networking.

Qiang Liu received the B.S. and Ph.D. degrees, in Communication and Information System from the Beijing Institute of Technology, in 2002 and 2007. From 2007, he was an Assistant Professor with the School of Computer and Information Technology, Beijing Jiaotong University. Since 2019, he has been an Associated Professor with the School of Computer and Information Technology, Beijing Jiaotong University. He is the author of more than 50 articles and 4 patents. His research interests include mobile ad-hoc networks, UAV ad-hoc networks, wireless media access control, wire-less routing, and swarm intelligence.

Ying He received the B.Eng. degree in Telecommunications Engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and the Ph.D. degree in Telecommunications Engineering from the University of Technology Sydney, Ultimo, NSW, Australia, in 2017. She is currently a Senior Lecturer with the School of Electrical and Data Engineering, University of Technology Sydney. Her research interests are physical-layer algorithms in wireless communication, vehicular communication, spectrum sharing, 5G, and IoT.

Qianqian Wu received the B.S. degree in Network Engineering from the Zaozhuang University, Zaozhuang, China, in 2019, and the M.S. degree in Computer Technology from North China University of Science and Technology, Tangshan, China, in 2022. She is currently working towards the Ph.D. degree with the School of Computer and Information Technology, Beijing Jiaotong University. Her current research interests include computer networks and SDN.

Ren Ping Liu is a Professor and Head of Discipline of Network & Cybersecurity at University of Technology Sydney (UTS). Since joining UTS in 2016, Prof. Liu has received 18 research grants (led 12) totaling over $5million. As a Research Leader, a certified network professional, and a full stack web developer, he has delivered networking and cybersecurity solutions to a number of government agencies and industry customers. His research interests include wireless networking, 5G, IoT, vehicular networks, 6G, cybersecurity, and block-chain. He has supervised over 30 Ph.D. students, and has over 200 research publications. Professor Liu was the founding Research Program Leader of Digital Agrifood Technologies in Food Agility CRC, a $50million initiative to empower Australia's food industry through digital transformation. He was the Co-founder and CTO of Ultimo Digital Technologies Pty Ltd. developing IoT and block-chain. Prior to that he was a Principal Scientist and Research Leader at CSIRO, where he led wireless networking research activities contributing to large scale industrial research projects valued over $100million. Professor Liu was the winner of NSW iAwards 2020 for leading the BeFAQT (blockchain enabled fish provenance and quality tracking) project. He was awarded the Australian Engineering Innovation Award 2012 and CSIRO Chairman's medal for his contribution in the wireless backhaul project. Professor Liu was the founding chair of IEEE NSW VTS Chapter and a Senior Member of IEEE. He served as Technical Program Committee Chairs and Organizing Committee Chairs in a number of IEEE Conferences.

Yantao Sun is now an Associate Professor in the School of Computer and Information Technology. He received his B.S. degree from Shandong University of Technology in 1999, received his M.S. degree from Shandong University in 2002 and Ph. D degree from Institute of Software, Chinese Academy of Science in 2006, respectively. His research interests include cloud computing, data center network, wireless sensor network, the Internet of things, multimedia communication, and network management. He published over 30 papers on international journals and conferences such as WCMC, Mobile Networks and Applications, Journal of Communication, GLOBECOM, etc. He also holds 6 patents.