PVD-TD3: A Latency-Oriented Multi-Agent Reinforcement Learning Algorithm for Multipath Routing in DetNet

Ying Yang, Yang Xiao, and Jun Liu
School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing, China
Email: {ynot, zackxy, liujun}@bupt.edu.cn

***Abstract*—The growing demand for ultra-reliable low-latency communication (URLLC) in real-time and mission-critical applications necessitates more rigorous deterministic networking (DetNet) in network traffic engineering. However, DetNet still struggles with routing optimization ensuring efficient resource allocation under growing quality-of-service (QoS) requirements. To address this issue, we propose a multi-agent deep reinforcement learning (MADRL)-based multipath routing method for effective flow scheduling in DetNet. Specifically, we design a novel partially value decomposition (PVD) and twin delayed deep deterministic policy gradient (TD3) algorithm, termed PVD-TD3. The proposed algorithm decomposes the joint action-value function within the neighborhood of each agent, thus enhancing scalability and minimizing communication overhead. Additionally, we introduce a latency-oriented weight calculation mechanism to assess the importance of each agent, enabling the weighted aggregation of value functions and improving adaptability to DetNet. Experimental results demonstrate that our proposed PVD-TD3 algorithm significantly outperforms benchmark methods. Compared to the average performance of the baseline methods, PVD-TD3 reduces the end-to-end (E2E) delay by 48.03% and the packet loss rate by 22.91%.**

***Index Terms*—Deterministic networking, multi-agent deep reinforcement learning, multipath routing.**

I. Introduction

Critical applications such as autonomous driving and Industrial Internet of Things (IIoT) pose huge challenges for ultra-reliable low-latency communication (URLLC) [1]. To meet the strict real-time requirements, deterministic networking (DetNet) emerges as the key solution. Traditional DetNet can guarantee deterministic flow transmission and meet ultra-low latency demands for time-sensitive applications, but it requires carefully optimized routing methods. This is because routing optimization plays a critical role in mitigating end-to-end (E2E) delays caused by deterministic flow congestion, dynamic traffic, and link failures, ensuring the network meets stringent quality-of-service (QoS) requirements, including delay constraints and bandwidth constraints [2]. Delay constraints ensure that data packets are transmitted within an acceptable time frame, which is crucial for time-sensitive applications such as real-time communications and video streaming. Bandwidth constraints, on the other hand, guarantee that the network can handle the required data rates, preventing congestion and ensuring sufficient capacity for high-throughput services [3]. However, routing optimization in DetNet requires precise resource allocation under strict latency and bandwidth constraints, while maintaining consistent service quality across the network. Although traditional protocols such as open shortest path first (OSPF) [4] and routing information protocol (RIP) [5] offer predefined solutions, they often face challenges in adapting to dynamic network conditions, leading to suboptimal routing performance.

To address these challenges, deep reinforcement learning (DRL) has emerged as a promising approach for routing optimization [6]. Through continuous interaction with the environment, DRL enables the network to dynamically learn and adapt optimal routing strategies. Recent studies demonstrate that DRL is highly effective in optimizing routing and improving network performance under dynamic conditions. For instance, Mai et al. [7] applied DRL for in-network load balancing, where distributed switches optimize load distribution in real-time using local observations and a centralized critic. Similarly, Wang et al. [8] investigated DRL-based probabilistic cognitive routing in software-defined networks (SDN), utilizing OMNeT++ and P4 for simulation and real-world evaluation. In addition, Xiao et al. [9] introduced a general problem formulation and the RL4Net framework, enabling rapid prototyping and evaluation of DRL algorithms to minimize network delay.

Building on these advancements, several studies have further applied DRL techniques to optimize routing in DetNet, where the focus shifts to meeting stringent constraints such as low latency, high reliability, and minimal packet loss. To address multiobjective routing and scheduling in DetNet, Yang et al. [10] proposed a multipolicy DRL framework with multistrategy optimization and graph convolutional networks to improve schedulability, resource utilization, and generalization. Chen et al. [11] focused on packet survival time in IP networks to meet delay-sensitive networking requirements. Similarly, Liu et al. [12] introduced the OdIR framework, which combines real-time network state awareness with an improved deep deterministic policy gradient (DDPG) algorithm to reduce overhead and enhance QoS in deterministic networks.

Building on the advancements in DRL-based routing optimization for DetNet, the above studies have proposed various algorithmic and structural improvements to enhance network performance. However, most of these approaches rely on single-path routing, which overlooks key factors like load balancing and congestion control. To address these limitations,

This work was supported by the National Natural Science Foundation of China under the Grant No. 62371057. *(Corresponding author: Yang Xiao.)*

ISBN 978-3-903176-72-0 © 2025 IFIP

multi-path routing has been suggested as an effective strategy in DetNet, allowing for better distribution of traffic across multiple paths and thus improving network robustness and performance [13]. As multi-path routing gains attention in DetNet research, recent studies have begun to explore its integration with DRL to further optimize routing decisions. In this context, Yu et al. [14] proposed a DRL-based flow scheduler for mixed-criticality DetNet flows, using a branching dueling Q-network (BDQN) to optimize routing and scheduling. The method improves scheduling scalability and efficiency, with evaluations showing better performance of multi-path scheduling over single-path scheduling in terms of schedulability. However, lacking multi-agent deep reinforcement learning (MADRL), their approach cannot support the decentralized decision-making needed to coordinate agents in large-scale networks, which limits its scalability. Although another work by Xiao et al. [15] pioneered advancements in multipath routing with MADRL by introducing mechanisms for scalability and QoS awareness, their method overlooks several constraints specific to DetNet, such as delay bounds and packet loss tolerance.

In this work, we propose a novel MADRL-based approach for multipath routing in DetNet. Specifically, we first formulate a multipath routing problem to transmit deterministic flows in DetNet under QoS constraints, which can optimize resource allocation and improve load balancing. For agent design, we introduce a constraints-aware reward function to effectively schedule flows of varying types. To facilitate flow transmission in DetNet, we propose a MADRL-based partially value decomposition (PVD) and twin delayed deep deterministic policy gradient (TD3) algorithm, termed PVD-TD3. The PVD method decomposes the joint action-value function within the neighborhood of each agent, enhancing scalability and reducing communication overhead. The integration of TD3 helps reduce overestimation bias, further improving learning stability. Additionally, we introduce a latency-oriented weight calculation mechanism to assess the importance of each agent, improving adaptability to DetNet. We evaluate the performance of PVD-TD3 compared to two MADRL-based methods and two conventional methods. Extensive experiments demonstrate that the proposed algorithm consistently outperforms these benchmark approaches. Compared to the average performance of the baseline methods, PVD-TD3 reduces the E2E delay by 48.03% and the packet loss rate by 22.91%.

The rest of this paper is organized as follows. Section II introduces our system model. Afterwards, Section III describes our problem formulation for routing optimization and the agent design. Section IV explains our proposed algorithm. Then, Section V evaluates the performance of our proposed algorithm. Finally, Section VI concludes this paper.

II. System Model

We first introduce the multipath routing model in DetNet. Then we give the delay and bandwidth constraints for deterministic flows. The key notations in this section are listed in Table I.

TABLE I
Key Notations in Section II

Parameters	Description
N, M	The number of switches and links
V, E, W	The set of switches, links, and link weights
v_n, e_m, w_{e_m}	The n-th switch, m-th link, and the weight of e_m
G	The network topology graph
P_{v_i,v_j}	The set of reachable paths from v_i to v_j
$p^p_{v_i,v_j}$	The p-th reachable path from v_i to v_j
f^t_h	The h-th deterministic flow at t
F^t	The set of deterministic flows in network G at t
$v^t_{h,src}, v^t_{h,dst}$	The source and destination nodes of f^t_h
λ^t_h	The arrival rate of f^t_h
$\xi^t_h \in \{\xi_H, \xi_S\}$	The deterministic flow type of f^t_h(ξ_H for HRT flows, ξ_S for SRT flows)
μ_n	The service rate of router v_n
K	The number of candidate paths
P'_{v_i,v_j}	The set of candidate paths of flow from v_i to v_j
X^t_h	The split ratio for f^t_h
$\chi^t_{h,k}$	The split ratio for f^t_h on $p'^{t}_{v^t_{h,src},v^t_{h,dst},k}$
$\lambda^t_{v_n}$	The aggregate arrival rate at v_n at t
s_n	The system capacity of router v_n
$P^t_{v_n}$	The packet loss probability on v_n
ρ^t_n	The utilization of v_n at t
$O^t_{v_n}$	The average queuing occupancy of v_n at t
$D_{f^t_h,k}$	The E2E delay of k-th sub-flow for flow f^t_h
$D_{f^t_h}$	The total E2E delay of deterministic flow f^t_h
D_{min}	The infimum of $D_{f^t_h}$ for any $f^t_h \in \mathcal{F}^t$
D_{max}	The supremum of $D_{f^t_h}$ for HRT flows
D^S_{max}	The supremum of $D_{f^t_h}$ for SRT flows
B_{e_m}	The bandwidth capacity of link e_m

A. Multipath Routing Model

To meet the stringent requirements of URLLC, we design a deterministic flow transmission method incorporating multi-path routing. In the considered DetNet system, we assume a network with N switches and M links, each serving as both an ingress and an egress node. Those switches are denoted by $V = \{v_1, \cdots, v_n, \cdots, v_N\}$, among which lies a set of directed links denoted by $E = \{e_1, \cdots, e_m, \cdots, e_M\}$. Then, the network topology is represented as $G = (V, E, W)$, while $W = \{w_{e_1}, \cdots, w_{e_m}, \cdots, w_{e_M}\}$ represents the set of link weights for each $e_m \in E$. Besides, we denote the shortest path from switch v_i to switch v_j by $P_{v_i,v_j} = \{p^1_{v_i,v_j}, \cdots, p^p_{v_i,v_j}, \cdots, p^{|P|}_{v_i,v_j}\}$, sorted in ascending order based on the cost of $p^p_{v_i,v_j}$. Formally, the cost of $p^p_{v_i,v_j}$ is defined as the sum of the link weights in W. We assume that the communication network G is a strongly connected graph where any pair of switches $v_i, v_j \in V (i \neq j)$ exist at least one forwarding path. For simplicity, we assume that G does not contain loops, meaning there is exactly one directed link between any pair of switches.

At each time step t, there is always one deterministic flow f^t_h arriving at each switch $v_n \in V$, thus the total number of flows is N. We denote the set of deterministic flows in G as $F^t = \{f^t_1, \cdots, f^t_h, \cdots, f^t_N\}$. Here, flow f^t_h is defined as being a source-destination pair from $v^t_{h,src}$ to $v^t_{h,dst}$, characterized by its arrival rate λ^t_h and deterministic flow type ξ^t_h. Based on criticality in terms of delay requirement, we classify flows in DetNet into two types, i.e., hard real-time (HRT, denoted by ξ_H) and soft real-time (SRT, denoted by ξ_S). Both HRT and

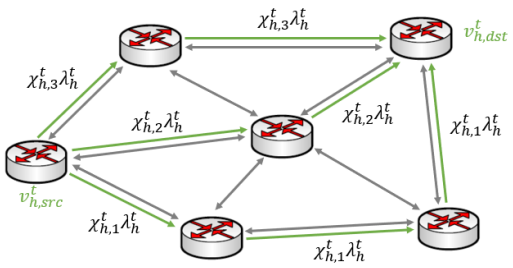


Fig. 1. The multipath routing pattern of deterministic flows in DetNet. For clarity, only one deterministic flow f^t_h is depicted. The flow f^t_h has $K = 3$ candidate paths and is split into three sub-flows (shown as green arrows) according to the split ratios $X^t_h = \{\chi^t_{h,1}, \chi^t_{h,2}, \chi^t_{h,3}\}$.

SRT flows are subject to delay bounds. However, the delay bound for HRT flows is rigid, and any violation of this bound could lead to catastrophic consequences. Consequently, the scheduling policy must ensure that all HRT flows are transmitted within their respective delay constraints. In contrast, the delay bound for SRT flows is flexible, meaning that while missing the deadline does not result in a failure, it may lead to performance degradation. The service rate of router v_n is denoted by μ_n, and the link weight w_{e_m} is defined as the reciprocal of the service rate μ_n at the link terminus.

At the beginning of deterministic flow transmission from v_i to v_j, we choose the top-K shortest paths from P_{v_i,v_j} as the candidate paths, which is denoted as $P'_{v_i,v_j} = \{p'_{v_i,v_j,1}, \cdots, p'_{v_i,v_j,k}, \cdots, p'_{v_i,v_j,K}\}$. Here, K is defined as the number of candidate paths. When transmitting flow f^t_h, it is split into K sub-flows across candidate paths $P'_{v^t_{h,src},v^t_{h,dst}}$ based on the split ratio at $v^t_{h,src}$. The split ratio is represented as $X^t_h = \{\chi^t_{h,1}, \cdots, \chi^t_{h,k}, \cdots, \chi^t_{h,K}\}$. Here, $0 < \chi^t_{h,k} < 1$ and $\sum_{k=1}^{K} \chi^t_{h,k} = 1$. Therefore, the arrival rate of each sub-flow on the path $p'_{v^t_{h,src},v^t_{h,dst},k}$ is obtained by $\chi^t_{h,k}\lambda^t_h$. Particularly, when $K > |P_{v^t_{h,src},v^t_{h,dst}}|$, we have $\chi^t_{h,k'} = 0$, for $k' \in (|P_{v^t_{h,src},v^t_{h,dst}}|, K]$. Note that flows split only once at the source node, and no further splitting occurs at transit routers during a time step. The multipath routing pattern of deterministic flows in DetNet is illustrated in Fig.1.

We denote the aggregate arrival rate at v_n at t by $\lambda^t_{v_n}$, and the system capacity of router v_n is denoted by s_n. Then, the packet loss probability on v_n at t is expressed as follows

$$P^t_{v_n} = \frac{(1-\rho^t_n)(\rho^t_n)^{s_n}}{1-(\rho^t_n)^{s_n+1}}, \tag{1}$$

where ρ^t_n is the utilization of v_n obtained by $\lambda^t_{v_n}/\mu_n$. The average queuing occupancy of v_n at t is calculated as

$$O^t_{v_n} = \begin{cases} \frac{\rho^t_n}{1-\rho^t_n} - \frac{(s_n+1)(\rho^t_n)^{s_n+1}}{1-(\rho^t_n)^{s_n+1}} & , \text{if } \rho^t_n < 1, \\ \frac{s_n}{2} & , \text{if } \rho^t_n = 1. \end{cases} \tag{2}$$

Notably, the propagation delay is neglected for E2E delay calculations due to its negligibly small value. Based on the packet loss probability $P^t_{v_n}$ and the average queuing occupancy $O^t_{v_n}$, the E2E delay of k-th sub-flow for flow f^t_h is defined as the sum of delay of all the routers on path $p'_{v^t_{h,src},v^t_{h,dst},k}$, expressed as

$$D_{f^t_h,k} = \sum_{v_n \in p'_{v^t_{h,src},v^t_{h,dst},k}} \frac{O^t_{v_n}}{\lambda^t_{v_n}(1-P^t_{v_n})}. \tag{3}$$

Thus, the total E2E delay of deterministic flow f^t_h is calculated by the weighted sum of end-to-end delays of each path $p'_{v^t_{h,src},v^t_{h,dst},k} \in P'_{v^t_{h,src},v^t_{h,dst}}$, given by

$$D_{f^t_h} = \sum_{k=1}^{K} \chi^t_{h,k} D_{f^t_h,k}. \tag{4}$$

B. Constraints for Deterministic Flows

In DetNet, both E2E delay and link load are critical constraints that must be carefully managed to meet the performance requirements of time-sensitive applications [14]. These two factors play a crucial role in ensuring that the network operates within the desired bounds, thereby maintaining the reliability and efficiency of the system. Therefore, to account for these constraints, we define the following two conditions for flow f^t_h as follows:

1) E2E delay constraints: The delay of a single flow is limited. For HRT flow f^t_h with $\xi^t_h = \xi_H$, the E2E delay is strictly confined within a specified range. In contrast, for SRT flow f^t_h with $\xi^t_h = \xi_S$, the latency requirement is comparatively less stringent. Formally, we have

$$\mathbf{C1}: D_{min} \leqslant D_{f^t_h} \leqslant D_{max}, \quad (\xi^t_h = \xi_H), \tag{5}$$

$$\mathbf{C2}: D_{min} \leqslant D_{f^t_h} \leqslant D^S_{max}, \quad (\xi^t_h = \xi_S), \tag{6}$$

where D_{min} is the lower bound of the E2E delay for all deterministic flows. Additionally, D_{max} and D^S_{max} are the upper bounds of the E2E delay for HRT and SRT, respectively, with $D_{max} < D^S_{max}$.

2) Bandwidth constraints: The traffic load on any link e_m must not exceed the allocated bandwidth capacity, which is expressed as

$$\mathbf{C3}: \lambda^t_{e_m} \leqslant B_{e_m}, \tag{7}$$

where $\lambda^t_{e_m}$ denotes the aggregate arrival rate on link e_m, and B_{e_m} is the bandwidth capacity of link e_m.

III. Problem Formulation and Agent Design

We introduce a customized QoS utility function to assess the performance of multipath routing in DetNet. The utility function U^t comprehensively considers the two metrics of E2E delay and packet loss rate, expressed as

$$U^t = \alpha_d \left(1 - \frac{\sum_{h=1}^{N} \lambda^t_h D_{f^t_h}}{\hat{D}^t \sum_{h=1}^{N} \lambda^t_h}\right) + \alpha_p \left(1 - \frac{\sum_{n=1}^{N} \lambda^t_{v_n} P^t_n}{\sum_{n=1}^{N} \lambda^t_{v_n}}\right). \tag{8}$$

α_d and α_p are scaling factors for the two main QoS metrics, with $\alpha_d + \alpha_p = 1$. $\hat{D}^t$ is denoted as the maximum E2E delay

of all flows at t. Then, the joint flow split ratio of all flows is denoted by $X^t = \{X_1^t, \cdots, X_h^t, \cdots, X_N^t\}$, forming the optimization objective as

$$\max_{X^t} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} U^t, \tag{9}$$

$$\text{s.t. } \mathbf{C1}, \mathbf{C2}, \mathbf{C3},$$

where T denotes the duration of the whole routing task.

The routing optimization problem is modeled as a partially observable Markov decision process (POMDP), represented by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \mathcal{R}, \gamma \rangle$. $\mathcal{N} = \{1, \cdots, n, \cdots, |\mathcal{N}|\}$ represents the set of agents, $\mathcal{S}$ denotes the global state space, and $\mathcal{A} = \{a_1^t, \cdots, a_n^t, \cdots, a_{|\mathcal{N}|}^t\}$ is the joint action space. The state transition probability is defined as $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. $\mathcal{O} = \{o_1^t, \cdots, o_n^t, \cdots, o_{|\mathcal{N}|}^t\}$ represents the joint observation space, and $\mathcal{R} = \{r_1^t, \cdots, r_n^t, \cdots, r_{|\mathcal{N}|}^t\}$ denotes the reward function. The discount factor is γ. In our setting, each switch is instantiated as an agent, with $|\mathcal{N}| = N$, and agents collaborate using a distributed learning approach to achieve the optimization goal. At each time step t, upon receiving the local observation o_n, agent n selects an action a_n and receives a reward r_n from the environment. The local observation then transitions to o_n' at the next time step. We denote the discounted cumulative reward for agent n at t by G_n^t, which is expressed as

$$G_n^t = \lim_{T\to\infty} \sum_{t'=0}^{T} \gamma^{t'} r_n^{t+t'}, \tag{10}$$

where γ balances the immediate reward and the future rewards. The detailed observation, action, and reward function are defined as follows.

A. Observation

The observation of agent n at time step t is denoted by o_n^t, which can be expressed as

$$o_n^t = \left\{ v_{h,dst}^t, \lambda_h^t, H_h^t, L_h^t, \xi_h^t \right\} |_{v_{h,src}^t = v_n}, \tag{11}$$

where $v_{h,dst}^t$ represents the destination node of flow f_h^t, and λ_h^t denotes the arrival rate of f_h^t (with $v_{h,src}^t = v_n$). The first two components inform agent n where the current flow f will be forwarded and how much traffic will be on the candidate paths. The third component, $H_h^t = \{H_{h,1}^t, \cdots, H_{h,k}^t, \cdots, H_{h,K}^t\}$, is the set of hop counts for each candidate path of f_h^t, guiding preference of the agent for flow distribution on candidate paths. $L_h^t = \{L_{h,1}^t, \cdots, L_{h,k}^t, \cdots, L_{h,K}^t\}$ represents the average link utilization for $p'_{v_{h,src}^t, v_{h,dst}^t, k}$ at the previous time step. This incorporates historical information to enhance network performance by considering previous link utilization patterns. Formally, we have $L_{h,k}^t = \sum_{e_m \in \mathcal{E}_{p'_{h,k}}} \lambda_{e_m}^t / B_{e_m}$, where $\mathcal{E}_{p'_{h,k}}$ is the set of all links on path $p'_{v_{h,src}^t, v_{h,dst}^t, k}$. Finally, ξ_h^t indicates the deterministic flow type of f_h^t, enabling the agent to formulate different strategies for different types of flows. Thus, the cardinality of o_n^t is $2K + 3$.

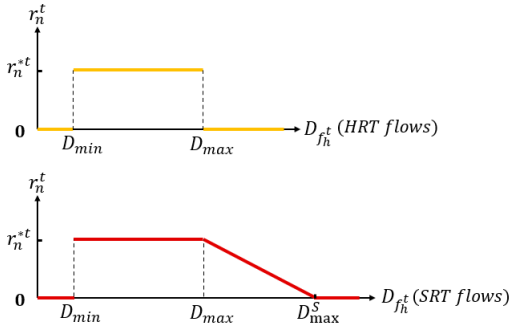


Fig. 2. Illustration of the constraints-aware reward function r_n^t variations for HRT and SRT flows.

B. Action

The action of agent n is the split ratio of flow f_h^t entering the DetNet at v_n, which is expressed as

$$a_n^t = X_h^t = \{\chi_{h,1}^t, \cdots, \chi_{h,k}^t, \cdots, \chi_{h,K}^t\}|_{v_{h,src}^t = v_n}. \tag{12}$$

Then, the cardinality of a_n^t is K, where K is the number of candidate paths as defined in Section II. Since any $\chi_{h,k}^t \in a_n^t$ is a real number within the range $[0, 1]$, the action space remains continuous.

C. Reward Function

The raw reward of agent n at time t is decoupled from the overall utility function U^t, which is expressed as

$$r_n^{*t} = \alpha_d r_{n,d}^t + \alpha_p r_{n,p}^t, \tag{13}$$

where $r_{n,d}^t = 1 - D_{f_h^t} / \hat{D}^t$ denotes the normalized E2E delay of agent n, and $r_{n,p}^t = 1 - P_n^t$ denotes the normalized packet loss rate. We design the constraints-aware reward function r_n^t based on the E2E delay constraints for HRT and SRT flows. For all deterministic flows, when the delay constraint is satisfied, $r_n^t = r_n^{*t}$; otherwise, $r_n^t = 0$. Specifically, for SRT flows, when $D_{f_h^t}$ falls between D_{max} and D_{max}^S, r_n^t decreases linearly until it reaches 0. Fig. 2 illustrates how the value of r_n^t varies. Since our routing problem involves distributed communication among agents, we model the optimization framework as a general-sum Markov process [16].

IV. Algorithm Design

A. Workflow of the Proposed Algorithm

To balance multi-agent collaboration and independence, we introduce the value decomposition (VD) method [17] to decompose the global value function into local ones. However, current VD methods face challenges in large-scale systems due to the increased computational complexity of global value aggregation and communication overhead. Toward this problem, we design a novel PVD method that integrates value functions within the neighborhood of each agent, reducing both communication and computational burdens. Despite its advantages, PVD struggles to handle continuous action spaces and still faces challenges with learning stability and value overestimation in large-scale systems. To solve the issues, we

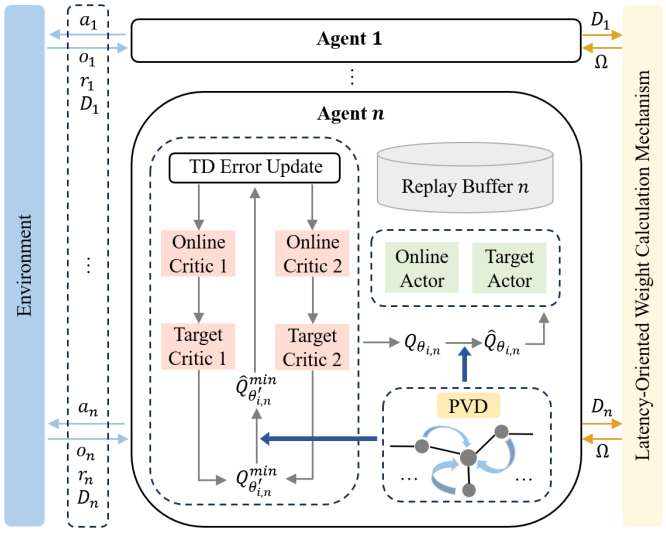


Fig. 3. Illustration of the designed MADRL-based PVD-TD3 algorithm for multipath routing in DetNet.

integrate the TD3 [18] algorithm with the PVD method. TD3 is a deterministic reinforcement learning algorithm designed for continuous action spaces, incorporating twin critics to reduce overestimation bias and implementing delayed policy updates to enhance learning stability. This combination allows PVD to handle continuous actions more effectively while improving the overall learning robustness in large-scale systems.

Building on the TD3 algorithm and PVD method, we design a MADRL-based PVD-TD3 algorithm, as illustrated in Fig. 3. In DetNet, each router is modeled as an agent with two actor and four critic networks. Specifically, the online critic networks $Q_{\theta_{1,n}}$ and $Q_{\theta_{2,n}}$, and the online actor network π_{ϕ_n} are initialized with parameters $\theta_{1,n}$, $\theta_{2,n}$, and ϕ_n, respectively. The corresponding target networks are initialized as $\theta'_{1,n} \leftarrow \theta_{1,n}$, $\theta'_{2,n} \leftarrow \theta_{2,n}$, and $\phi'_n \leftarrow \phi_n$. For critic networks, $Q_{\theta_{i,n}}(o_n^t, a_n^t)$ approximate the expected cumulative reward, which represents the expected return after taking a particular action in a given state. Formally, $Q_{\theta_{i,n}}(o_n^t, a_n^t)$ can be defined as:

$$Q_{\theta_{i,n}}(o_n^t, a_n^t) = \mathbb{E}[G_n^t]. \tag{14}$$

By learning the Q-value, the objective of PVD-TD3 algorithm is to maximize these expected cumulative rewards, thereby optimizing the policy selection. Our method adopts a mixed cooperative approach, leveraging distributed learning through inter-agent communication during the training phase.

At the start of each episode, agents interact with the environment, forming transition tuples $(o_n^t, a_n^t, r_n^t, o_n^{t+1}, D_n^t)$, which are stored in the replay buffer $\mathcal{B}_n$. Here, D_n^t represents the E2E delay $D_{f_h^t}$ (with $v_{h,src}^t = v_n$), and o_n^{t+1} denotes the observation at the next time step. After collecting sufficient experience, agents update their networks at the end of each episode. The training process consists of N_e episodes, each comprising N_s time steps. During training, agent n samples a mini-batch of B transitions from $\mathcal{B}_n$, and evaluates the current policy with the online critic networks $Q_{\theta_{i,n}}(o_n^t, a_n^t)$ for $i = 1, 2$. Then, the target critic networks compute the

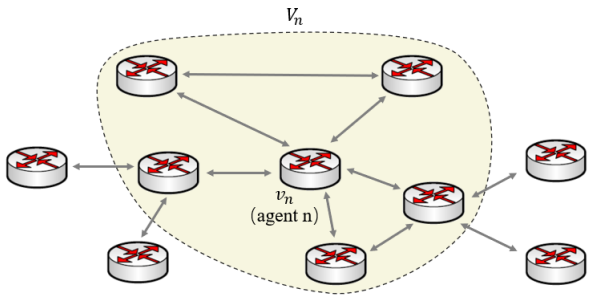


Fig. 4. The region V_n of value function decomposition for agent n (shown in yellow area).

target action-value function $Q_{\theta'_{i,n}}(o_n^{t+1}, a_n^{t+1})$, where a_n^{t+1} is produced by the target actor network with o_n^{t+1}. Subsequently, agent n computes the minimum target action-value function $Q_{\theta'_{i,n}}^{min}(o_n^{t+1}, a_n^{t+1})$ within the two target critics.

Before updating the neural network parameters, we first apply the PVD method. Unlike the conventional VD method, which decomposes the value function for all agents, this approach decomposes the value function within the one-hop neighborhood of each agent. Formally, the region of value function decomposition for agent n is denoted as V_n. The region V_n is illustrated in Fig. 4. Since the behavior of an agent is typically influenced by the other agents within its neighborhood, the PVD method decomposes the action-value function within a local neighborhood, which not only more accurately reflects the local environment but also enhances computational efficiency and system scalability.

To achieve this, we first design a latency-oriented weight calculation mechanism to assess the importance of each agent based on E2E delay D_n^t. This approach ensures better adaptation to the DetNet environment from a global network perspective. The agent weights are denoted as $\Omega_n^t = \{\omega_1^t, \cdots, \omega_n^t, \cdots, \omega_{|\mathcal{N}|}^t\}$, where the weight ω_n for each agent is computed as

$$\omega_n^t = 1 - \beta \cdot \frac{D_n^t - \bar{D}_n^t}{D_{max} - D_{min}}. \tag{15}$$

Here, $\bar{D}_n$ represents the average E2E delay of agents in V_n. The factor $\beta \in (0, 1)$ controls the balance between agents, where a smaller β reduces disparities in weights, while a larger β promotes equal weighting among agents. The main assumption we make and exploit in PVD is the joint action-value function for the region V_n can be additively decomposed into value functions across agents:

$$\hat{Q}_n^t = \sum_{v_n \in V_n} \omega_n^t Q_n^t, \tag{16}$$

where Q_n^t is the action-value of agent n, and $\hat{Q}_n^t$ is the processed action-value after applying the PVD method.

Following the PVD method, the minimum processed action-value function $\hat{Q}_{\theta'_{i,n}}^{min}(o_n^{t+1}, a_n^{t+1})$ from the two target critics is

Algorithm 1 The Proposed PVD-TD3 Algorithm

Initialize the online critic networks $Q_{\theta_{1,n}}$ and $Q_{\theta_{2,n}}$, the online actor networks π_{ϕ_n} with parameters $\theta_{1,n}$, $\theta_{2,n}$, and ϕ_n. Initialize the target networks $\theta'_{1,n} \leftarrow \theta_{1,n}$, $\theta'_{2,n} \leftarrow \theta_{2,n}$, $\phi'_n \leftarrow \phi_n$, and the replay buffers $\mathcal{B}_n$ for all agents;
for episode $i = 1$ to N_e **do**
  for time step $t = 1$ to N_s **do**
    for agent $n = 1$ to $|\mathcal{N}|$ **do**
      Sample mini-batch of B transitions from $\mathcal{B}_n$;
      Compute $Q_{\theta_{i,n}}(o_n^t, a_n^t)$ and $Q_{\theta'_{i,n}}(o_n^{t+1}, a_n^{t+1})$;
      $Q^{min}_{\theta'_{i,n}}(o_n^{t+1}, a_n^{t+1}) = \min_{i=1,2} Q_{\theta_{i,n}}(o_n^t, a_n^t)$;
    end for
    for agent $n = 1$ to $|\mathcal{N}|$ **do**
      Calculate $\Omega_n^t = \{\omega_1^t, \cdots, \omega_n^t, \cdots, \omega_{|\mathcal{N}|}^t\}$ by (15);
    end for
    for agent $n = 1$ to $|\mathcal{N}|$ **do**
      Calculate $\hat{Q}^{min}_{\theta'_{i,n}}(o_n^{t+1}, a_n^{t+1})$ and $\hat{Q}_{\theta_{i,n}}(o_n^t, a_n^t)$ by (16), and calculate y_n^t by (17);
      Update online critic networks by (19);
      if i mod d **then**
        Update online actor network by (20);
        Update target networks by (21);
      end if
    end for
  end for
end for

utilized to compute the target action-value function y_n^t, which can be expressed as:

$$y_n^t \leftarrow \hat{r}_n^t + \gamma \hat{Q}^{min}_{\theta'_{i,n}}(o_n^{t+1}, a_n^{t+1}). \tag{17}$$

It should be noted that we learn action-value function by backpropagating gradients from the Q-learning rule using the weighted local reward $\hat{r}_n^t$ through the summation, without relying on any reward specific to the individual agent. Here, $\hat{r}_n^t$ is the weighted summation of the rewards for all agents within V_n, which is expressed as

$$\hat{r}_n^t = \sum_{v_n \in V_n} \omega_n r_n^t. \tag{18}$$

After obtaining the target action-value function y_n^t, we utilize it to update the online critic networks, thereby refining its estimation of the action-value function. The temporal difference (TD) error is used to quantify the discrepancy between the current action-value function $\hat{Q}_{\theta_{i,n}}(o_n^t, a_n^t)$ and the target action-value function y_n^t. By minimizing the TD error, we update the parameters of the online critic networks. Specifically, the critic networks are optimized by minimizing the mean squared error (MSE), which is formulated as follows:

$$\theta_{i,n} \leftarrow \text{argmin}_{\theta_{i,n}} |\mathcal{N}|^{-1} \sum \left(y_n^t - \hat{Q}_{\theta_{i,n}}\left(o_n^t, a_n^t\right)\right)^2. \tag{19}$$

Upon updating the online critic networks, the online actor network is then updated by utilizing the online critic networks estimates. The objective of optimizing the actions selected by the online actor is to maximize the action-value function predicted by the online critic networks. Periodically, the online actor is updated with a delay of d time steps relative to the online critics, using the deterministic policy gradient:

$$\nabla_{\phi_n} J(\phi_n) = |\mathcal{N}|^{-1} \sum \nabla_{a_n^t} \hat{Q}_{\theta_i,n}(o_n^t, a_n^t)\Big|_{a_n^t} \nabla_{\phi_n} a_n^t. \tag{20}$$

Subsequently, all the target networks are updated. Specifically, the parameters of the target networks are updated towards the parameters of the online networks with a certain ratio, typically using a small update step size:

$$\begin{aligned} \theta'_{i,n} &\leftarrow \tau\theta_{i,n} + (1 - \tau\theta'_{i,n}), \\ \phi'_n &\leftarrow \tau\phi_n + (1 - \tau\phi'_n), \end{aligned} \tag{21}$$

where τ is the soft update coefficient, controlling the balance between fast adaptation and stability. After the target networks are updated, the agent continues to interact with the environment, progressing to the next time step. Through this process, the agent continually collects new experiences, which are then used to optimize the performance of both the actor and critic networks. Simultaneously, the agent refines its policy through ongoing exploration and learning, aiming to enhance overall performance. The pseudocode for the PVD-TD3 algorithm is presented in **Algorithm 1**.

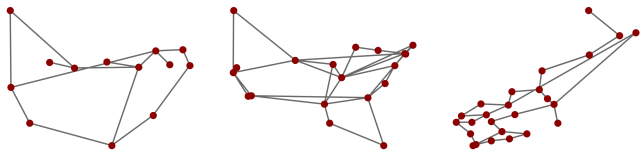
B. Computational Complexity

In our proposed PVD-TD3 algorithm, the computational complexity mainly depends on the optimization of critic and actor networks. As shown in Section III, the input and output dimensions of the actor network π_{ϕ_n} are $2K + 3$ and K, respectively. For the two critics $Q_{\theta_{i,n}}$, the input dimension is $3K + 3$, while the output dimension is 1. Considering that fully connected layers are employed for all hidden layers in the networks, we define the number of hidden layers in the actor and the two critics as H_{ϕ_n} and $H_{\theta_{i,n}}$, respectively. Furthermore, the number of neurons in the h-th hidden layer of the actor and the two critics are denoted as $Y_{\phi_n,h}$ and $Y_{\theta_{i,n},h}$, respectively. According to the proposed algorithm PVD-TD3, the computational complexity of each actor network at any training time step t is $O(C_{\phi_n}) = O((2K + 3)Y_{\phi_n,1} + \sum_{h=1}^{H_{\phi_n}-1} Y_{\phi_n,h}Y_{\phi_n,h+1} + KY_{\phi_n,H_{\phi_n}})$. Also, the computational complexity of both two critics for each agent at any training time step is $O(C_{\theta_{i,n}}) = 2O((3K + 3)Y_{\theta_{i,n},1} + \sum_{h=1}^{H_{\theta_{i,n}}-1} Y_{\theta_{i,n},h}Y_{\theta_{i,n},h+1} + Y_{\theta_{i,n},H_{\theta_{i,n}}})$. Finally, the computational complexity of the optimization phase is $O(N_e(N_s \sum_{n=1}^{|\mathcal{N}|} C_{\theta_{i,n}} + \frac{N_s}{d} \sum_{n=1}^{|\mathcal{N}|} C_{\phi_n}))$. The linear scaling of the computational complexity, with respect to the number of agents $|\mathcal{N}|$, the number of episodes N_e, and the episode duration N_s demonstrates its scalability to large-scale networks.

V. Performance Evaluation

A. Experimental Setting

The experiment is conducted using the NetworkX library [19] for network topology generation. As illustrated in Fig. 5(a), NSFNET (13 nodes, 30 links) is the default topology to form the network in the experiment. At each time step t, the arrival rate λ_h^t of flow f_h^t follows a Zipf distribution with



(a) NSFNET (b) InternetMCI (c) Sunet

Fig. 5. Illustration of the NSFNET, InternetMCI, and Sunet topologies.

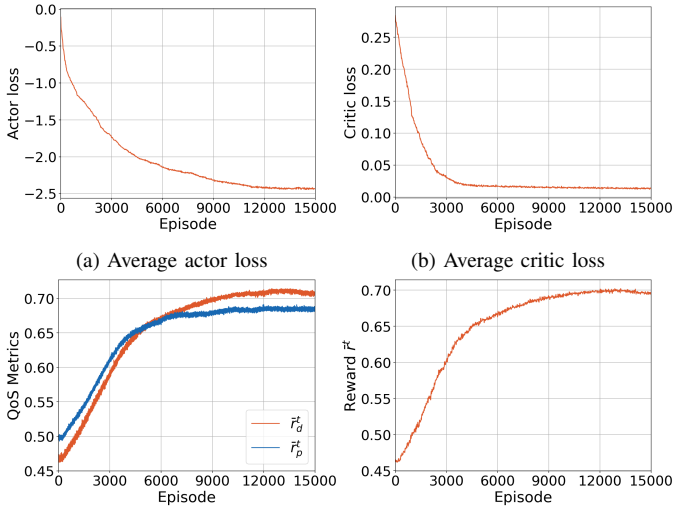
rates $\{500, 550, 600\}$ packets per second (packets/s), where the average traffic arrival rate, denoted by $\bar{\lambda}_h^t$, is 550 packets/s. Here, the packet size is fixed at 100 bytes. The service rate μ_n of each router n is sampled from a discrete distribution of $\{1000, 2000, 3000\}$ packets/s. Each router has a system capacity s_n of $10,000$ packets, and the number of candidate paths K is set to 3. The E2E delay bounds for different flow types are set to $D_{min} = 0$, $D_{max} = 46$, and $D_{max}^S = 50$ ms, respectively. These delay requirements are fine-tuned based on the preliminary QoS metrics in the default scenario. The bandwidth B_{e_m} for all links is uniformly set to 2.4 Mbps.

The training process consists of $N_e = 15$k episodes, each containing $N_s = 50$ time steps. The scaling factor α_d and α_p are set to 0.7 and 0.3, respectively. The importance factor β is set to 0.4 by default. The soft update coefficient τ is set to 0.005, and the delay of the soft update is set to $d = 2$ time steps. The learning rates for the actor and critic networks are set to $l_a = 2 \times 10^{-6}$ and $l_c = 5 \times 10^{-5}$, respectively. Both networks have two hidden layers with 64 and 32 neurons. The discount factor γ is set to 0.9, and a dropout rate of 0.3 is applied to mitigate overfitting. The replay buffer size $|\mathcal{B}_n|$ is 10000 and the mini-batch size B is set to 256.

We compare the performance of the proposed PVD-TD3 algorithm against two MADRL-based algorithms and two traditional routing protocols, detailed as follows:

- Multi-Agent TD3 (MA-TD3) [18]: A deterministic MADRL approach employing decentralized training and execution (DTDE). In this method, agents update policies based only on local information, without inter-agent communication.
- Multi-Agent Deep Deterministic Policy Gradient (MA-DDPG) [20]: Another DTDE-based MADRL method, built upon the DDPG algorithm.
- Equal-Cost Multi-Path (ECMP) [21]: A traditional routing solution that splits traffic evenly across shortest paths with equal costs.
- Load Balancing (LB) [22]: A method that distributes traffic evenly across all available candidate paths.

For all benchmark methods, the environment settings remain consistent with the default configuration. The learning rates for the actor and critic networks are adjusted to $l_a = 1 \times 10^{-6}$ and $l_c = 5 \times 10^{-6}$ for MA-TD3, and $l_a = 5 \times 10^{-7}$ and $l_c = 1 \times 10^{-6}$ for MA-DDPG. To demonstrate the applicability of the PVD-TD3 method, we employ two additional topologies, i.e., InternetMCI (19 nodes, 66 links) and Sunet (26 nodes, 62 links), depicted in Fig. 5(b) and 5(c). The corresponding E2E delay bounds are set to $D_{max} = 33$ ms and $D_{max}^S = 36$ ms for InternetMCI, and $D_{max} = 57$ ms and $D_{max}^S = 65$ ms for Sunet, reflecting their distinct network requirements, while $D_{min} = 0$ in both cases.



(a) Average actor loss (b) Average critic loss

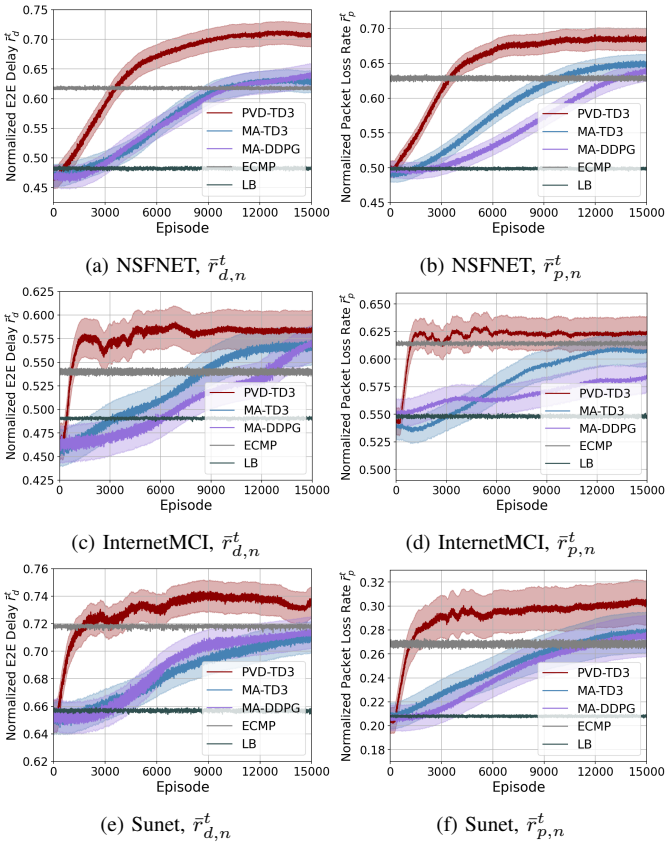
(c) Average QoS metrics (d) Average reward

Fig. 6. Convergence performance of the proposed PVD-TD3 algorithm in terms of (a) actor loss, (b) critic loss, (c) QoS metrics, and (d) reward on NSF. The results are averaged over 5 independent runs.

B. Experimental Results and Analysis

To provide a comprehensive evaluation of the proposed PVD-TD3 algorithm, we consider several key performance indicators (KPIs), including actor loss and critic loss discussed in Section IV, reward and the two QoS metrics outlined in Section III. These KPIs are assessed based on the average values of their single-agent counterparts across all agents at each time step. Specifically, the average reward and the two normalized QoS metrics (i.e. normalized end-to-end delay and normalized packet loss rate) at time t are denoted as $\bar{r}^t$, $\bar{r}_d^t$ and $\bar{r}_p^t$. We evaluate PVD-TD3's performance using two general QoS metrics: normalized E2E delay reward $\bar{r}_d^t$ and normalized packet loss rate reward $\bar{r}_p^t$. A higher delay reward $\bar{r}_d^t$ reflects better network delay optimization, while a higher packet loss reward $\bar{r}_p^t$ reflects better optimization of packet loss rate. All the results are averaged over five independent runs for each of the five methods.

1) Convergence: Fig. 6(a)-(b) illustrate the average actor loss, critic loss, QoS metrics and reward achieved by our proposed PVD-TD3 algorithm on NSFNET with 5 independent runs. As illustrated in Fig. 6(a), the actor loss exhibits a rapid decrease during the early stages of training, signifying the agent's swift enhancement in selecting actions associated with higher Q-values. The subsequent slight fluctuations reflect the ongoing trade-off between exploration and exploitation as the agent fine-tunes its policy. Eventually, the actor loss stabilizes and converges to a value of approximately -2.4, indicating that the policy has reached a confident and near-optimal state. In Fig. 6(b), a similar trend is observed for the critic loss, which experiences a sharp decline during the initial training phase, followed by a reduction in variance as the loss



(a) NSFNET, $\bar{r}_{d,n}^t$ (b) NSFNET, $\bar{r}_{p,n}^t$

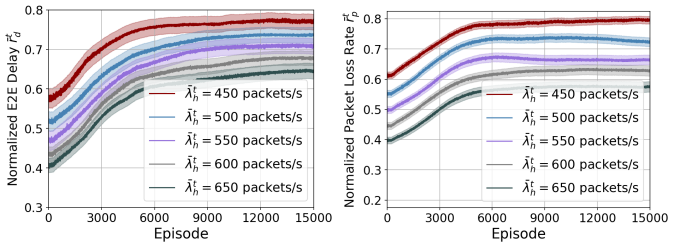
(c) InternetMCI, $\bar{r}_{d,n}^t$ (d) InternetMCI, $\bar{r}_{p,n}^t$

(e) Sunet, $\bar{r}_{d,n}^t$ (f) Sunet, $\bar{r}_{p,n}^t$

Fig. 7. The E2E delay $\bar{r}_{d,n}^t$ and packet loss rate $\bar{r}_{p,n}^t$ performance of PVD-TD3 and baseline methods on three typical topologies.

approaches zero. This behavior aligns with the expectation derived from (19), where the critic loss is minimized to a small positive value, suggesting an accurate evaluation of the policy. As depicted in Figs. 6(c) and 6(d), the trends of the two QoS metrics are consistent with the overall reward. In the early stages of training, the overall reward increases rapidly as the agent learns effective policies, which is reflected in substantial improvements across both two metrics. Specifically, both $\bar{r}_d^t$ and $\bar{r}_p^t$ show the most significant improvement, ultimately stabilizing at high levels. This suggests that the agent prioritizes delay reduction while maintaining a well-balanced trade-off between reliability and delay.

2) Scalability: Fig. 7(a)-7(f) present the results of three different topologies in terms of $\bar{r}_d^t$ and $\bar{r}_p^t$. The shaded areas of PVD-TD3, MA-TD3, and MA-DDPG indicate the standard deviation among agents. Overall, PVD-TD3 consistently outperforms the four benchmark methods. The relative E2E delay is calculated as $1 - \bar{r}_d^t$, and the relative packet loss rate is calculated as $1 - \bar{r}_p^t$. Considering the average of all the experiments in this part, PVD-TD3 approximately reduces the relative E2E delay by 48.03% and the relative packet loss rate by 22.91% compared to the average performance of the baseline methods. Particularly, it demonstrates more stable convergence compared to the two MADRL-based benchmark algorithms. Then, the two traditional methods maintain relatively stable performance with slight fluctuations, as they do



(a) $\bar{r}_d^t$ (b) $\bar{r}_p^t$

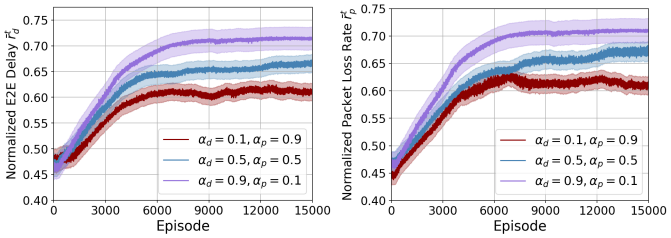
Fig. 8. PVD-TD3's normalized E2E delay $\bar{r}_d^t$ and normalized packet loss rate $\bar{r}_p^t$ performance under different average arrival rates on NSFNET.

not involve a training process. Furthermore, across different topology structures, PVD-TD3 continues to achieve superior convergence values. This phenomenon indicates that even on larger-scale topologies, PVD-TD3 provides better QoS performance than other benchmark methods. To evaluate the generalizability of the MADRL-based learning schemes, we apply a consistent set of environment and agent configurations across experiments involving our PVD-TD3 algorithm, MA-TD3, and MA-DDPG, across three distinct topologies. In this context, our proposed algorithm demonstrates superior generalizability, performing effectively across topologies of varying scales and structures. It is anticipated that, under more complex environmental settings, particularly with larger topological scales, our proposed algorithm will outperform other MADRL-based benchmark methods in addressing the multipath routing task for deterministic flows.

3) Stability: It is crucial to assess whether our proposed PVD-TD3 algorithm can maintain its performance for deterministic flows under varying network conditions. To this end, we focus on evaluating the performance including delay and packet loss rate of PVD-TD3 under different average arrival rates, scaling factors, and E2E delay bounds.

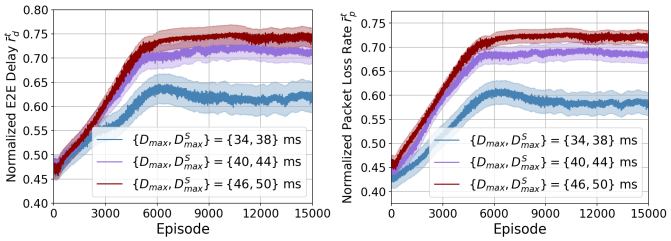
Fig. 8(a) and 8(b) illustrate the QoS metrics $\bar{r}_d^t$ and $\bar{r}_p^t$ achieved by PVD-TD3 under different average traffic arrival rate over 5 independent runs on NSFNET. We adjust the average traffic arrival rate $\bar{\lambda}_h^t$ as $\{450, 500, 550, 600, 650\}$ packets/s. Network traffic intensity is influenced by user demand and tends to fluctuate significantly over extended time scales. As we expected, lower levels of $\bar{\lambda}_h^t$ lead to an increase in $\bar{r}_d^t$ and $\bar{r}_p^t$ due to reduced traffic load. The results of all five experiments show stable convergence in the later stages, demonstrating that PVD-TD3 can effectively transmit deterministic flows under stringent constraints across varying levels of $\bar{\lambda}_h^t$, as reflected by low standard deviation and consistent convergence. Although an increase in traffic volume leads to some performance degradation, as anticipated, our proposed algorithm consistently maintains satisfactory performance for deterministic flows. This is demonstrated by its successful convergence and low standard deviation throughout the training process, even under varying traffic intensities.

Fig. 9(a) and 9(b) illustrates PVD-TD3's performance on NSFNET under different scaling factors for two QoS metrics in terms of $\bar{r}_d^t$ and $\bar{r}_p^t$. In this experiment, we select scaling



(a) $\bar{r}_d^t$ (b) $\bar{r}_p^t$

Fig. 9. PVD-TD3's normalized E2E delay $\bar{r}_d^t$ and normalized packet loss rate $\bar{r}_p^t$ performance under different scaling factors on NSFNET.



(a) $\bar{r}_d^t$ (b) $\bar{r}_p^t$

Fig. 10. PVD-TD3's normalized E2E delay $\bar{r}_d^t$ and normalized packet loss rate $\bar{r}_p^t$ performance under E2E delay bounds on NSFNET.

factors from the set α_d $\{0.1, 0.5, 0.9\}$, with greater emphasis on the E2E delay metric as α_d increases. Correspondingly, scaling factors α_p are set to $\{0.9, 0.5, 0.1\}$. The results of $\bar{r}_d^t$ and $\bar{r}_p^t$ consistently converges under different scaling factor settings. Additionally, a larger α_d results in higher convergence outcomes, as the latency-sensitive environment provides positive feedback when greater attention is given to the E2E delay metric for deterministic flows. This phenomenon further highlights the strong compatibility of PVD-TD3 with DetNet environments, thereby demonstrating its effectiveness in optimizing performance under stringent delay constraints. The algorithm's adaptability to DetNet is primarily facilitated by its latency-oriented weight calculation mechanism, which prioritizes the crucial role of E2E delay in DetNet.

Fig. 10(a) and 10(b) present PVD-TD3's performance on NSFNET in terms of $\bar{r}_d^t$ and $\bar{r}_p^t$ across three different delay bounds. The delay constraints $\{D_{max}, D^S_{max}\}$ for HRT and SRT flows are set to $\{34, 38\}$, $\{40, 44\}$ and $\{46, 50\}$ ms, while D_{min} is fixed at 0. As expected, increasing $\{D_{max}, D^S_{max}\}$ leads to higher $\bar{r}_d^t$ and $\bar{r}_p^t$ convergence outcomes, which can be attributed to relaxed delay constraints. Notably, convergence stability remains unaffected by varying latency constraints, demonstrating the robustness of PVD-TD3.

VI. Conclusion

In this paper, we investigated a MADRL-based multipath routing method for deterministic flows with E2E delay and bandwidth constraints in a DetNet scenario. To enhance adaptability, we proposed the MADRL-based PVD-TD3 architecture, which decomposes the joint action-value function within the one-hop neighborhood of each agent, utilizing a latency-oriented weight calculation mechanism. Experimental results showed that the proposed PVD-TD3 algorithm outperforms two MADRL-based benchmark algorithms and two traditional methods regarding E2E delay and packet loss rate. In future research, we plan to conduct larger-scale simulations and explore integrating techniques such as attention mechanisms.

References

[1] M. Almekhlafi, M. A. Arfaoui, C. Assi, and A. Ghrayeb, "Superposition-based URLLC traffic scheduling in 5G and beyond wireless networks," *IEEE Trans. Commun.*, vol. 70, no. 9, pp. 6295–6309, SEP 2022.

[2] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic networking architecture," in *RFC 8655*. IETF, 2019.

[3] J. Miserez, D. Colle, M. Pickavet, and W. Tavernier, "Exploiting queue information for scalable delay-constrained routing in deterministic networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 5, pp. 5260–5272, 2024.

[4] J. Moy, "OSPF version 2," RFC 2328, Apr. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2328

[5] "Routing information protocol," RFC 1058, Jun. 1988. [Online]. Available: https://www.rfc-editor.org/info/rfc1058

[6] Y. Xiao, J. Liu, J. Wu, and N. Ansari, "Leveraging deep reinforcement learning for traffic engineering: A survey," *IEEE Commun. Surv. Tutor.*, vol. 23, no. 4, pp. 2064–2097, 2021.

[7] T. Mai, H. Yao, and Z. Xiong, "Multi-agent actor-critic reinforcement learning based in-network load balance," in *GLOBECOM 2020 - IEEE Global Commun. Conf.*, no. 1, 2020, pp. 1–6.

[8] Y. Wang, Y. Xiao, Y. Song, J. Zhou, and J. Liu, "Deep reinforcement learning based probabilistic cognitive routing: An empirical study with OMNeT++ and P4," in *2023 19th Int. Conf. Netw. and Serv. Manag. (CNSM)*, 2023, pp. 1–7.

[9] Y. Xiao, J. Li, J. Wu, and J. Liu, "On design and implementation of reinforcement learning based cognitive routing for autonomous networks," *IEEE Commun. Lett.*, vol. 27, no. 1, pp. 205–209, 2023.

[10] S. Yang, L. Zhuang, J. Zhang, J. Lan, and B. Li, "A multipolicy deep reinforcement learning approach for multiobjective joint routing and scheduling in deterministic networks," *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17 402–17 418, 2024.

[11] J. Chen, Y. Xiao, G. Lin, G. He, F. Liu, W. Zhou, and J. Liu, "Deep reinforcement learning based dynamic routing optimization for delay-sensitive applications," *GLOBECOM 2023 - IEEE Global Commun. Conf.*, pp. 5208–5213, 2023.

[12] Y. Liu, J. Yin, W. Zhang, and S. Xie, "Deep reinforcement learning for on-demand intelligent routing in deterministic networks," in *GLOBECOM 2023 - IEEE Global Commun. Conf.*, 2023, pp. 1932–1937.

[13] S. K. Singh, T. Das, and A. Jukan, "A survey on internet multipath routing and provisioning," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 4, pp. 2157–2175, 2015.

[14] H. Yu, T. Taleb, and J. Zhang, "Deep reinforcement learning-based deterministic routing and scheduling for mixed-criticality flows," *IEEE Trans. Ind. Informat.*, vol. 19, no. 8, pp. 8806–8816, 2023.

[15] Y. Xiao, Y. Yang, H. Yu, and J. Liu, "Scalable QoS-aware multipath routing in hybrid knowledge-defined networking with multiagent deep reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 11, pp. 10 628–10 646, 2024.

[16] Z. Song, S. Mei, and Y. Bai, "When can we learn general-sum markov games with a large number of players sample-efficiently?" *CoRR*, vol. abs/2110.04184, 2021.

[17] P. Sunehag, G. Lever, and A. Gruslys, "Value-decomposition networks for cooperative multi-agent learning," *CoRR*, vol. abs/1706.05296, 2017.

[18] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018.

[19] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using NetworkX." [Online]. Available: https://www.osti.gov/biblio/960616

[20] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2020. [Online]. Available: https://arxiv.org/abs/1706.02275

[21] C. Hopps, "RFC2992: Analysis of an equal-cost multi-path algorithm," USA, 2000.

[22] S. Prabhavat, H. Nishiyama, N. Ansari, and N. Kato, "On load distribution over multipath networks," *IEEE Commun. Surv. Tutor.*, vol. 14, no. 3, pp. 662–680, 2012.