applied sciences

MDPI

Article

Multi-Path Routing Algorithm Based on Deep Reinforcement Learning for SDN

Yi Zhang [1], Lanxin Qiu [1], Yangzhou Xu [1], Xinjia Wang [1], Shengjie Wang [2], Agyemang Paul [2] and Zhefu Wu [2,*]

[1] Information Communication Branch of State Grid Zhejiang Electric Power Co., Hangzhou 310007, China; zhangyi_zjdl@163.com (Y.Z.); qiulx44520@163.com (L.Q.); xuyangzhou_zjdl@163.com (Y.X.); 402413123@163.com (X.W.)

[2] College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China; 221122030266@zjut.edu.cn (S.W.); 221122030325@zjut.edu.cn (A.P.)

* Correspondence: wzf@zjut.edu.cn

Abstract: Software-Defined Networking (SDN) enhances network control but faces Distributed Denial of Service (DDoS) attacks due to centralized control and flow-table constraints in network devices. To overcome this limitation, we introduce a multi-path routing algorithm for SDN called Trust-Based Proximal Policy Optimization (TBPPO). TBPPO incorporates a Kullback–Leibler divergence (KL divergence) trust value and a node diversity mechanism as the security assessment criterion, aiming to mitigate issues such as network fluctuations, low robustness, and congestion, with a particular emphasis on countering DDoS attacks. To avoid routing loops, differently from conventional 'Next Hop' routing decision methodology, we implemented an enhanced Depth-First Search (DFS) approach involving the pre-computation of path sets, from which we select the best path. To optimize the routing efficiency, we introduced an improved Proximal Policy Optimization (PPO) algorithm based on deep reinforcement learning. This enhanced PPO algorithm focuses on optimizing multi-path routing, considering security, network delay, and variations in multi-path delays. The TBPPO outperforms traditional methods in the Germany-50 evaluation, reducing average delay by 20%, cutting delay variation by 50%, and leading in trust value by 0.5, improving security and routing efficiency in SDN. TBPPO provides a practical and effective solution to enhance SDN security and routing efficiency.

Keywords: software-defined network; deep reinforcement learning; multiple routing algorithm; network security

Citation: Zhang, Y.; Qiu, L.; Xu, Y.; Wang, X.; Wang, S.; Paul, A.; Wu, Z. Multi-Path Routing Algorithm Based on Deep Reinforcement Learning for SDN. *Appl. Sci.* **2023**, *13*, 12520. https://doi.org/10.3390/app132212520

Academic Editor: Christos Bouras

Received: 19 October 2023
Revised: 9 November 2023
Accepted: 12 November 2023
Published: 20 November 2023

1. Introduction

The advancement of communication networks has witnessed swift development in mobile communication, the Industrial Internet of Things (IIoT), and emerging internet technologies. The result is the emergence of a varied spectrum of network types molding the current communication panorama. The interplay and coexistence of these networks have grown crucial due to their deployment in unattended or hostile settings and their inherent openness, rendering them vulnerable to potential intrusions by malicious actors. Moreover, the proliferation of mobile devices connecting to networks has challenged traditional network access methods and policies, necessitating the refinement of routing strategies to meet dynamic information transmission needs. One promising solution to these evolving challenges is Software-Defined Networking (SDN). SDN is a flexible and efficient networking approach that offers a global view of the network and enables the selection of optimal paths based on real-time network conditions [1]. Its fundamental characteristic is the separation of the control plane from the data plane, where a centralized control function maintains the network's state and issues instructions to data plane devices [2]. This architecture provides clear security advantages, allowing for real-time analysis and correlation of network feedback. Also, due to the special programmability of SDN, programmable

routing based on SDN has also become quite popular in recent years [3–6]. However, it also presents concerns related to open programmability and trust between network elements, particularly when using technologies like OpenFlow. OpenFlow, the most commonly used SDN technology, is a focal point of security analysis. The STRIDE threat analysis method applied to OpenFlow reveals vulnerabilities, including susceptibility to Denial of Service (DoS) attacks [7]. DoS attacks on SDN can exhaust controller resources by flooding it with fake flow table requests and disrupt communication by targeting the southbound interface between the controller and network devices. To address these security threats, various technologies like intrusion detection, authentication, and access control have been used [8]. However, these are typically passive measures against specific vulnerabilities. It is crucial to include proactive security measures in network system design to enhance immunity against threats.

Recently, a trust management-based security mechanism has been proposed [9–12]. Trust levels are assigned to network nodes based on direct and prior observations, guiding decisions in future interactions [13]. This approach enhances network security by avoiding nodes with security vulnerabilities during route discovery [14–17]. Despite trust-based methods showing potential, their complexity, when coupled with deep learning, hinders real-time applications. The traditional Dijkstra shortest path algorithm [17] faces issues like slow convergence and responsiveness, causing network congestion, especially in dynamic environments with increasing traffic [18].

Machine learning, particularly deep reinforcement learning, has gained prominence due to its exceptional performance in data processing, classification, and intelligent decision making. This has led to the development of several popular algorithms, including the Deep Q Network (DQN) [19], Actor–Critic (AC) [20], Deep Deterministic Policy Gradient (DDPG) [21], Trust Region Policy Optimization (TRPO) [22], etc. More recently, researchers have explored integrating deep reinforcement learning into SDN routing specifications to achieve intelligent routing and fine-grained management [23–26]. In their pioneering work, Casas et al. [25] proposed a DQN-based deep reinforcement learning (DRL) scheme to generate dynamic traffic changes for SDN network routing. Likewise, Alkhalaf et al. [26] introduced a Proximal Policy Optimization (PPO)based deep reinforcement learning technique to improve SDN network routing, allowing for real-time intelligent control and administration. However, these efforts often neglect security considerations in real-time communication. Therefore, there is a pressing need for an efficient deep learning algorithm that offers strong privacy protection, real-time communication capabilities, and high efficiency.

In this study, a novel multi-path routing approach called Trust-Based Proximal Policy Optimization (TBPPO) is introduced for SDN. TBPPO leverages a trust value mechanism based on Kullback–Leibler divergence (KL divergence) [27] and a node diversity assessment mechanism to enhance SDN robustness, address congestion problems, and notably fortify defenses against Distributed Denial of Service (DDoS) attacks. Furthermore, an improved PPO algorithm is tailored for optimizing multi-path routing in SDN, while also considering security, network delay, and variations in multi-path delays. This advancement is crucial for improving SDN routing optimization. To enhance computational efficiency, an enhanced Depth-First Search (DFS) algorithm is incorporated, simplifying the action space. The experimental results validate TBPPO's superior performance in terms of convergence and overall effectiveness when compared to the traditional methods. The contribution of this work can be summarized as follows:

1. We present TBPPO, a multi-path routing algorithm designed specifically for SDN. This innovative approach integrates a trust value mechanism based on KL-divergence and node diversity as a key security assessment criterion. TBPPO addresses network fluctuations, enhances robustness, and mitigates congestion issues, with a primary focus on countering DDoS attacks.
2. We present an enhanced PPO algorithm to optimize security and efficiency in SDN routing. This novel algorithm involves the optimization of multi-path routing, con-

sidering factors like security, network delay, and variations in multi-path delays. To address the recurrent gradient explosion issues observed during the experimental process, we introduced learning rate decay and layer normalization. Furthermore, we incorporated a trust value-based routing selection approach, resulting in enhanced security stability and a reduced delay performance.

3. To avoid routing loops, we abandoned the traditional Next Hop routing mechanism and adopted a path selection approach. The improved DFS uses a path selection method to choose a set of promising routes, which are called routing groups. Then, we search for the best multi-path route within these routing groups. This enhancement significantly reduces packet loss and improves the overall efficiency and practicality of the algorithm.
4. Finally, we conducted experiments using the NSFNet-14 and Germany-50 network topologies. The results demonstrate that our TBPPO technique outperforms traditional approaches in terms of both convergence and overall performance. Additionally, the obtained findings emphasize TBPPO's potential as an effective solution for improving SDN security and routing efficiency.

The subsequent sections of this paper will be organized as follows: Section 2 provides an overview of related works. Section 3 offers a detailed description of the system model, encompassing crucial elements such as the SDN network environment. Section 4 delves into the design and specific details of the TBPPO algorithm. This includes its operational principles and key algorithms. Section 5 presents simulation results and corresponding analyses to validate and evaluate the performance of the TBPPO algorithm in various scenarios. Section 6 summarizes the experimental conclusions.

2. Related Works

2.1. Trust-Based Security Mechanisms

Trust-based security mechanisms have gained popularity in Wireless Sensor Networks (WSN) and the Internet of Things (IoT) [28–33]. These mechanisms focus on ensuring the privacy and security of network nodes. Several trust-based algorithms have been introduced in recent studies. Ashwin et al. [34] introduced a weighted clustering trust model algorithm that demonstrated significant improvements in identifying malicious nodes. Rajeswari et al. [35] proposed a trust-based next-hop node selection algorithm, which improved the network performance in terms of the data packet transmission, delay, and error rates. Zhang et al. [36] presented a cloud-based trust evaluation approach that is sensitive to various attacks and capable of improving malicious node detection accuracy. Mingwu et al. [37] introduced trust entropy and a standard structural entropy mechanism for detecting malicious behavior in sensor systems. Subhash et al. [38] utilized machine learning to include parameters such as friendship and community interest, shedding light on the evolution of trust in an entity over time. In another study, Subhash et al. [39] employed a heuristic approach based on machine learning to amalgamate trust-related attributes, successfully distinguishing between trustworthy and untrustworthy nodes within the network. Claudio et al. [40] introduced an incremental Support Vector Machine (iSVM) method for simulating various attack patterns, outperforming other methods. Besat et al. [41] used the K-Nearest Neighbors (KNN) algorithm to detect selfish behavior in entities effectively. Wafa et al. [42] employed machine learning methods for trust parameter aggregation and a mixed propagation approach to classify users and detect attack types. However, many of these methods have high computational complexity and are unsuitable for real-time communication scenarios when combined with deep learning. They did not further investigate the application of security mechanisms in route planning. Currently, there is not a market solution that simultaneously considers low delay, high security, and computational efficiency for multi-route planning. Therefore, our work intends to fill this blank.

2.2. Reinforcement Learning in SDN

SDN has shifted traditional network operations from hardware to software, both simplifying and integrating the functionality of the network control plane while also enhancing the reliability of network hardware devices. With the continuous growth in network complexity and traffic demands, traditional shortest path algorithms suffer from drawbacks such as slow convergence and network congestion. Researchers have turned to machine learning, specifically deep reinforcement learning, to optimize SDN route selection. Compared to traditional methods, DRL may introduce some additional overhead. Deep learning models have a large number of parameters, requiring more storage space to save and load the model. DRL models need to be trained, which might require a significant amount of time and data. Traditional methods are usually based on fixed algorithms and do not require a training process. DRL might experience unstable phases when learning and adapting to new environments, which could lead to short-term instability in routing policies. Some DRL approaches use an experience replay mechanism, necessitating the storage and processing of vast amounts of historical data. Despite these overheads, DRL remains attractive for SDN routing decisions because it can learn and adapt to complex network environments and often outperforms traditional methods in many scenarios. Various studies have explored the application of deep reinforcement learning in SDN route optimization, focusing on intelligent routing, customization, and fine-grained management. Table 1 presents a succinct overview of recent research employing DRL in SDN.

The above table summary elucidates the learning techniques applied, the formulation of actions, and the criteria used for assessment. In the studies [43,44], the authors employed the RL method Q-Learning, which relies on Q-tables and demands significant memory, data, and time resources. Conversely, refs. [25,45,49] used the DRL method DQN, and [46] employed the Dueling Double DQN method, which optimizes DQN. These methods utilize deep learning networks to approximate values instead of Q-tables, making them more practical and scalable than Q-Learning. Additionally, refs. [26,47] respectively used the Advanced AC (A2C) and PPO methods, which employ policy gradients instead of value approximation. These methods can enhance convergence compared to DQN. In studies [26,43–46,48], the choice of the next hop is used as the action. These studies reveal that this approach fulfills end-to-end performance requirements only when the appropriate next hop is selected. An incorrect choice can lead to significant performance degradation and even routing loops. In contrast, ref. [47] employs the adjustment of graph neural network parameters as the action, which requires reinforcement learning assisted by a specific graph neural network, making it less generalizable. Furthermore, the concept of the takeover decision of the switches was employed as the action in [48] to adaptively mitigate the propagation of attacks, thereby enhancing the resilience of Software Defined Industrial Networks (SDIN). The authors in [25] select routes from a preselected route group, which eliminates the possibility of routing loops and provides stable connections. However, as the number of nodes increases, the count of preselected paths may grow rapidly, introducing security vulnerabilities. Therefore, there is a compelling need to devise high-efficiency DRL algorithms capable of addressing these concerns. Contrary to what is described in Section 2.1, these studies only considered single-route optimization for the entire route without taking into account multi-route performance. More critically, they did not consider the security performance under a DDoS attack environment, thereby overlooking the potential adverse impacts, such as network congestion, that the crucial factor of security might bring to the network. In contrast to prior research, our emphasis is on redressing the imbalance in recent studies, which prioritized network performance but overlooked vital aspects such as privacy, security, and real-time communication delay. We introduce an enhanced PPO algorithm tailored for optimizing multi-path routing in SDN, with a significant focus on security, network delay, and variations in multi-path delays.

Table 1. Reinforcement Learning in SDN related research.

Paper	Description	Learning Approach	Action	Performance Metrics
[25]	DRSIR: A DQN-based solution for intelligent routing in SDN based on path-state metrics.	DQN	Path Selection	Delay, Loss and Throughput
[26]	A DRL-based method, which can maximize numerous objectives to dynamically update the routing strategy.	PPO	Next Hop	Average Delay, Maximum Delay
[43]	RLBeep: A RL-based method to increase the lifetime in wireless sensor networks.	Q-Learning	Next Hop	First node death time
[44]	RLMR: A Q-Learning based method performs routing for different flows, based on the real-time information of network state and flow characteristics.	Q-Learning	Next Hop	Forwarding Efficiency, Loss rate
[45]	NetworkAI: A network architecture using network monitoring technologies and artificial intelligence for generating control policies.	DQN	Next Hop	Delay
[46]	RLRouting: a reinforcement learning routing algorithm solving traffic engineering (TE) problem of SDN in terms of throughput and delay.	Dueling DDQN	Next Hop	Utilization rate, Delay and CPU Usage
[47]	DRL-GS: A DRL-based solution to optimize topology in SDN	A2C,PPO,GNN	Parameter of GNN	Entropy Loss, Training Accuracy and Testing Accuracy
[48]	DQSP: A DRL-based solution concerning QoS in SDN-IoT	DDPG	Next Hop	Packet Delivery Ratio, Delay, Probablity of passing attacked nodes
[49]	An attack mitigation scheme based on DRL to adaptively prevent the spread of attacks.	DQN	Takeover Decision of the Switches	Request Arrival Rate

3. System Model

3.1. Network Module

The overall objective of the system is to find a given number of paths within a given network topology while also achieving optimal performance in terms of delay, delay variation, security, and other comprehensive factors. Specifically, for a given network, its topology is represented as $G =< V, E >$, where $V = \{v_1, v_2, \ldots\ldots, v_N, \}$ represents the set of all vertices, and $E = \{e_1, e_2, \ldots\ldots, e_M\}$ represents the set of all edges. The network topology G is represented using two adjacency matrices, M_{topo} and $M(\tau)$. M_{topo} is the initial adjacency matrix that includes delay information and is used for preliminary exploration in the depth-first algorithm. $M(\tau)$ is an adjacency matrix that represents the connectivity of links and their corresponding delay. It serves as the input state for the deep reinforcement learning component. For nodes v_i and v_j, if there is a connected edge e_x with a current delay of d, then $e_x = e_{i,j} = d$. Conversely, if there is no connected edge or $i = j$, then $e_{i,j} = -1$. In this paper, the policy-based network topology optimization problem is transformed into a maximization problem of the objective function:

$$\max_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] \tag{1}$$

where θ represents the parameters of the deep reinforcement neural network, $J(\theta)$ represents the objective function of the neural network, and $\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$ is the expected reward R at time τ, following the distribution of the neural network policy π_θ.

3.2. Attack Module

The system aims to assess the vulnerability of a network over a series of time windows T in the presence of external attacks characterized by the strategy π_{att}. Each node in the network, denoted as v_i, is assigned a probability of being attacked, represented as

$p(v_i) = \pi_{att}(v_i)$. Within each time window τ, node v_i generates a set of send–receive records that, to some extent, indicate the node's susceptibility to attacks:

$$Generator_{\pi_{att}}(\tau, v_i) = Group(\tau, v_i) \tag{2}$$

where $Generator_{\pi_{att}}$ represents the mapping of the external environment generating attack records using the strategy π_{att}. $Group(\tau, v_i)$ represents the records of node v_i at time τ, and $Group(\tau)$ represents the record group of all nodes at time τ. When defining the time window τ and using $M(\tau)$ to represent the delay matrix, the changes in the delay matrix caused by $Group(\tau, v_i)$ are described by the mapping relationship $Trans$ as follows:

$$Trans_{Group(\tau)}(M(\tau - 1), \tau) \rightarrow M(\tau) \tag{3}$$

3.3. Deep Reinforcement Learning Module

Deep reinforcement learning is a deep learning algorithm that involves two main components, including the environment and the agent. The agent selects an action according to the environment state and receives the reward and the next state from the environment. The main goal of the DRL algorithm is to train the agent to select the actions in the environment that maximize its rewards. The environment for DRL is designed as a Markov decision process, so it can only solve Markov decision process problems. DRL is divided into off-policy methods and on-policy methods. Off-policy methods use different strategies to collect experiences and use these experiences to improve other target strategies. On-policy methods directly collect experiences based on the current policy and use these experiences to improve the policy. Compared to off-policy methods, on-policy methods, because they cannot use the experiences of old policies, need to interact with the environment more frequently to collect experiences, which leads to a decrease in learning efficiency. However, because they are updated according to a consistent policy, the updates obtained are more effective and the decision-making performance stability of the new policy is higher. Typical off-policy methods for SDN include DQN [19]. Typical on-policy methods for SDN are, for example, the AC [20], DDPG [21], and PPO [26], which are called policy gradient algorithms (PG). Compared to DQN, PPO has greater stability, which makes PPO perform better on complex problems than DQN. Transitioning from traditional policy gradient algorithms like DDPG and AC to PPO is driven by the quest for stability and efficiency. While DDPG excels in continuous action spaces, it is sensitive to hyperparameters; AC leverages asynchronous updates for diverse data but faces challenges in distributed training. PPO, on the other hand, employs a clipping mechanism to prevent large policy updates, offering a more stable and consistent learning experience. This makes PPO often outperform DDPG and AC without intricate tuning or handling asynchronous complexities (response to Reviewer 4, comment 10). Additionally, PPO usually requires less hyperparameter tuning, making it more convenient than DQN. In this work, we chose the on-policy PPO algorithm as our decision-making module and propose a TBPPO. We will provide specific descriptions of the algorithm in Section 4.5.

4. Trust-Based Proximal Policy Optimization (TBPPO)

This section introduces the TBPPO algorithm, which is a multi-objective, multi-path routing planning algorithm providing a secure multi-route scheme designed to provide low delay for real-time communication. The algorithm includes a preprocessing module based on DFS, a trust value calculation module using KL divergence, a Markov process transition mechanism, and a deep reinforcement learning decision module based on PPO. Figure 1 illustrates the overall workflow of the algorithm.

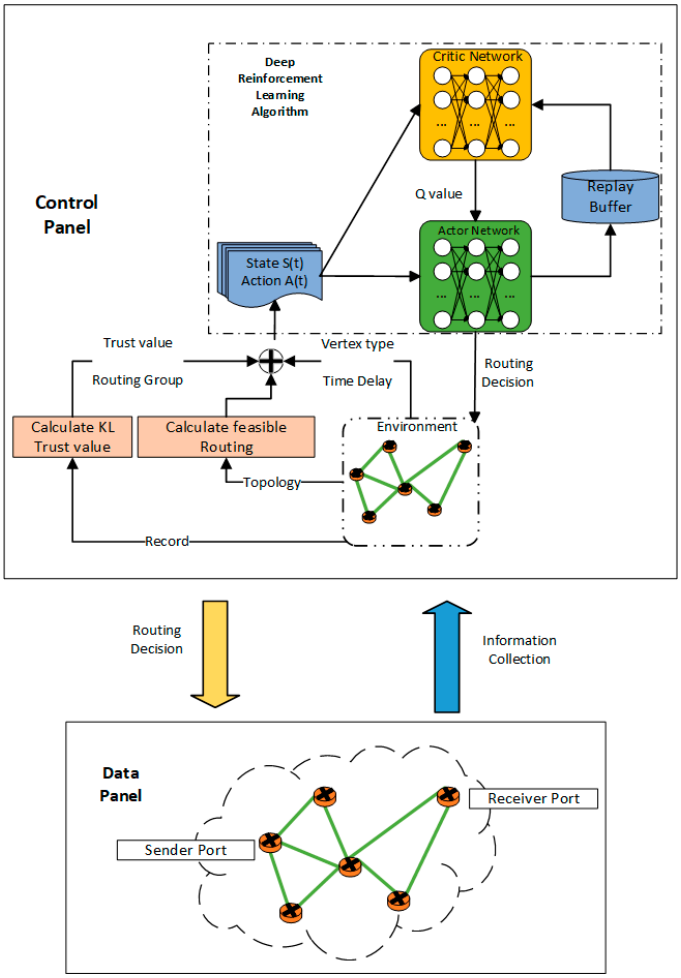


Figure 1. Schematic structure of the TBPPO algorithm.

In this algorithm, the data is divided and transmitted over multiple routes, thereby dispersing traffic across several paths to increase the effective bandwidth of the network, allowing multiple connections to work in parallel. If one path fails, the traffic can automatically switch to another, thereby enhancing network reliability. This method also takes into account security considerations. By dispersing traffic across multiple routes, it enhances network security, as it requires an attacker to compromise several paths at once, thereby raising the complexity and difficulty of network attacks. Our strategy accounts for real-time communication and security along each route, crafting a balanced approach that weighs various metrics including average delay, delay variations, KL trust value, and node diversity.

Figure 1 depicts the structure of our algorithm. For each time interval, the control panel gathers the current network topology, DDoS attack logs, and node types from the data panel. After processing via the trust value module and the DFS module, these features are concatenated to form a state that enters the DRL module. Within this module, the critic network calculates Q-values while the actor network determines the action. This action is then executed on the data panel and the experience is stored in the replay buffer. Upon completion of an episode, experiences are extracted from the replay buffer in sample batches to update the critic network before proceeding to the next episode.

4.1. The Improved DFS Module

Most deep reinforcement learning algorithms are designed based on Next Hop routing policies, as indicated in [44–47]. However, it has been shown that such policies are prone to leading to routing loops, causing longer delays and some degree of packet loss [50]. These

issues have not been adequately addressed in the aforementioned studies. To tackle these challenges, we introduce a path selection approach. The number of available paths in the network increases exponentially with the network's scale, which can be computationally prohibitive. Therefore, we introduce the concept of limiting the number of paths. We use the path selection approach in conjunction with the DFS algorithm to pre-select the K best paths, effectively sorting and selecting the optimal K paths from the available options. The initial topology structure, M_{topo}, containing delay information, is used as the input to the network. K best paths are computed to form the preselected routing group. The delay values associated with these preselected routes effectively replace the delay matrix $M(\tau)$ as part of the state representation. This preprocessing step results in a considerable reduction in the computational complexity of our algorithm. This preprocessing step can be expressed as:

$$\mathbb{L} = DFS_K(M_{topo}) \tag{4}$$

where, M_{topo} is the matrix topological structure, L represents the set of all possible paths generated after the application of the DFS algorithm, and its cardinality is denoted as L_0. DFS_K signifies the utilization of an enhanced DFS algorithm to select the shortest K routes.

4.2. Security Module

DDoS attacks represent a prevalent and disruptive network threat. These attacks involve malicious nodes intentionally discarding messages from legitimate nodes, causing severe disruptions to the data transmission. What makes these attacks particularly insidious is that these malicious nodes can mimic legitimate behavior when not receiving data packets, making them hard to detect and highly destructive [51–53]. To address the challenge of DDoS attacks and establish trust levels for network nodes, we introduce a KL divergence as a trust mechanism to counter network attacks. Initially, specific features are carefully selected as the KL benchmark, drawing from the normal network operation records. The choice of these features can be tailored to suit specific circumstances. The records are configured to combat DDoS attacks and take the following form:

$$Group(0, v_i) = (\text{SIP}, \text{DIP}, \text{SYN}, \text{DP}) \tag{5}$$

where SIP, DIP, SYN, and DP represent the frequency of the source IP, the destination IP, the SYN packets, and the different destination ports in the records of node v. At the end of each predefined time interval denoted as τ, we compute the KL divergence between the records of the node within that time frame and the baseline. A higher KL divergence value serves as an indicator of a higher probability that the node is currently under attack, consequently leading to a reduced security level. The formula for computing this divergence is given as follows:

$$trust(\tau, v_i) = KL(Group(0, v_i), Group(\tau, v_i)) \tag{6}$$

where $Group(0, v_i)$ represents the KL baseline, $Group(\tau, v_i)$ corresponds to the records of node v_i within the time window τ, $Trust(\tau, v_i)$ signifies the trust value of node v_i at time τ, and $Trust(\tau)$ denotes the trust values of all nodes at time τ. The term KL refers to the enhanced iterative KL divergence calculation formula, which is enhanced from the equation in [23] and defined as follows:

$$KL\left(p_{\|}q\right) = \sum_{i=1}^{I} \beta_i p(x_i) \left(log \frac{p(x_i)}{q(x_i)} \right) \tag{7}$$

where I represents the total data volume in the records. It is worth noting that the size of I may differ among nodes due to the varying number of destination ports. Moreover, β_i signifies the weights assigned to different parameters, and these weights are allocated based on specific mechanisms tailored to different attack models. Next, we introduce a novel node-type mechanism to enhance the management of various nodes within the system. Currently, different Internet ISP (ISP) are responsible for handling distinct nodes.

However, depending heavily on a single ISP can lead to critical security issues, including server outages, attacks on ISPs, and potential data breaches by ISPs [54]. In response to these challenges, we present a solution that focuses on diversifying the selection of multiple paths to reduce dependence on a single ISP. This approach aims to minimize security vulnerabilities associated with ISP-related issues. We design a metric Node Diversity presented as the ISP variance of an individual route to describe the node diversity of this route. In this context, the variable t is used to represent different ISP, and the Node Diversity in a single path is formulated as:

$$var(a_l) = \sum_{t \in T} \left(Count_{a_l}(t) - \overline{Count_{a_l}} \right)^2 \tag{8}$$

where t represents the node type currently under consideration. For all t belonging to the set T, $Count_{a_l}(t)$ is used to signify the number of nodes of this ISP along route a_l. Here, a_l represents the l-th route within the currently selected route group $A(\tau)$. T denotes the set encompassing all node types, expressed as follows:

$$T = [ISP\,1, ISP\,2, ISP\,3] \tag{9}$$

In the following, we use the notation δ_{v_i} to represent the node type of any given node v_i, and it is expressed as $\delta_{v_i} = t$. Collectively, we refer to the overall node-type configurations for all nodes as δ.

4.3. Markov Process Transition Module

After employing the trust value mechanism relying on the KL divergence for transformation, the SDN control plane becomes capable of presenting the network's real-time status, which encompasses information like the delay matrix, node categories, and node trust values. This issue is subsequently structured as a Markov Decision Process (MDP) and is characterized by a four-tuple (S, A, P, R); here, S represents the state space, A is the action space, P is the probability distribution function, and R represents the reward function, which is composed of five components: delay reward, delay variation reward, KL trust value reward, node diversity reward, and node redundancy reward. The state, denoted as S, is composed of three components at time τ: the delay values of the selected route group $\mathbb{L}$, represented as $D_{\mathbb{L}}(\tau)$; the trust values of each node at time τ, denoted as $Trust(\tau)$; and the types of each node, referred to as δ. This can be formulated as:

$$\mathrm{S}(\tau) = [D_{\mathbb{L}}(\tau), Trust(\tau), \delta] \tag{10}$$

where the action space A has a size of L_0, which corresponds to the size of the preselected route group $\mathbb{L}$. $A(\tau)$ is a subset of $\mathbb{L}$, and it consists of $\mathbb{L}$ routes, where the l-th route is expressed as a_l:

$$A(\tau) = [a_1, a_2, \ldots, a_l, \ldots, a_L] \tag{11}$$

where the probability distribution P is based on the current policy $\pi_{\theta_{\tau-1}}$, representing the probabilities of different actions for different states S(τ). For any given time window τ, the current policy $\pi_{\theta_{\tau-1}}$ based on parameters θ will generate an action $A(\tau)$ according to the probabilities in P:

$$A(\tau) = \pi_{\theta_{\tau-1}}(S(\tau)) \tag{12}$$

4.4. Reward Module

The Reward Mechanism's role is to assign rewards to the network with respect to multiple objectives. By adjusting the weighting factors α_o and the rewards r_o associated with different objectives, the system can guide the network's learning process. The formula

for calculating the reward $R(\tau)$ obtained by the network at the τ-th time window is represented as:

$$R(\tau) = \sum_{o \in O} \alpha_o r_o(\tau) \tag{13}$$

here α_o represents the weight of reward type o, and $r_o(\tau)$ represents the reward value of type o obtained during the τ time window. O represents the set of reward types, including the delay reward, delay variation reward, KL trust value reward, node diversity reward, and node redundancy reward. These can be expressed as:

$$O = \{delay, variation, diversity, safety, repeat\} \tag{14}$$

Here, *delay* represents the average delay, *variation* represents the delay variation, *diversity* represents the node diversity, $safety$ represents the route trust value, and *repeat* represents the node redundancy. The delay reward is defined as the average delay of the current path group $A(\tau)$ and it is defined as:

$$r_{delay}(\tau) = \frac{1}{L}\sum_{l=0}^{L} D_{M(\tau)}(a_l) = \frac{1}{L}\sum_{l=0}^{L}\sum_{e \in a_l} d_{M(\tau)}(e) \tag{15}$$

Here, $r_{delay}(\tau)$ represents the delay reward at time τ, $D_{M(\tau)}(a_l)$ is the delay of route a_l under $M(\tau)$, and $d_{M(\tau)}(e)$ is the delay of edge e under $M(\tau)$. The delay variation reward is expressed as the average delay variation for each path in the current path group $A(\tau)$ and is formulated as:

$$r_{variation}(\tau) = mean\left[\left|D_{M(\tau)}(a_i) - D_{M(\tau)}(a_j)\right| \,\middle|\, i < j \wedge i, j < L\right] \tag{16}$$

Here, $r_{variation}(\tau)$ represents the delay variation reward at time τ, and *mean* indicates the calculation of the mean value. The KL trust reward is defined as the mean of the minimum safety node trust values for each path in the current path set $A(\tau)$ and it can be expressed as:

$$r_{safety}(\tau) = \frac{1}{L}\sum_{l=0}^{L} \max_{v_i \in a_l}(Trust(\tau, v_i)) \tag{17}$$

Next, the node diversity reward is defined as the average of the variance of the number of nodes of each type in each path within the current path group $A(\tau)$ as follows:

$$r_t(\tau) = \frac{1}{L}\sum_{l=0}^{L} var(a_l) = \frac{1}{L}\sum_{l=0}^{L}\sum_{t \in T}\left(Count_{a_l}(t) - \overline{Count_{a_l}}\right)^2 \tag{18}$$

where the node redundancy reward is represented by the number of nodes that appear repeatedly in the current routing group $A(\tau)$.

4.5. Enhanced PPO Module

In traditional policy gradient algorithms, policy weights are typically updated by calculating the gradient of the objective function and applying it with a step size. However, this update process may encounter problems like overshooting or undershooting. To address these issues, we adopt the PPO algorithm. PPO is a policy gradient method in reinforcement learning, which enhances the policy by optimizing a surrogate objective function using stochastic gradients obtained by sampling the data from interactions with the environment. It allows for multiple small-batch updates, as opposed to a single gradient update for each data sample. The specific PPO variant employed in this research is the PPO-clip algorithm, which relies on a clipping mechanism. To prevent the importance sampling function from exceeding the predefined upper or lower bounds, a truncation function denoted as $J_{PPO}^{\theta^k}(\theta)$ is applied. This function automatically limits the importance

sampling values when they go beyond the specified upper or lower limits. This can be expressed as Equation:

$$J_{PPO}^{\theta^k}(\theta) \approx \sum_{(s_\tau, a_\tau)} \min\left(\frac{p_\theta(a_\tau \mid s_\tau)}{p_{\theta^k}(a_\tau \mid s_\tau)} A^{\theta^k}(s_\tau, a_\tau),\ clip\left(\frac{p_\theta(a_\tau \mid s_\tau)}{p_{\theta^k}(a_\tau \mid s_\tau)}, 1-\varepsilon, 1+\varepsilon \right) A^{\theta^k}(s_\tau, a_\tau) \right) \tag{19}$$

Here, $J_{PPO}^{\theta^k}(\theta)$ is used to assess the expected cumulative reward (performance) of the policy, θ is the current policy's parameter, and θ^k is the policy parameters at a previous time step or iteration step k. s_τ represent $S(\tau)$. a_τ represents the action taken at time step τ, and $a_\tau \in A(\tau)$. $clip\left(\frac{p_\theta(a_\tau|s_\tau)}{p_{\theta^k}(a_\tau|s_\tau)}, 1-\varepsilon, 1+\varepsilon \right)$ is a clipping function used to limit the ratio between $1-\varepsilon$ and $1+\varepsilon$, where ε is a small positive value usually employed to ensure that policy updates are bounded. The function A^{θ^k} represents the function, which provides an estimate of the advantage when taking action a_τ with the parameters set θ^k:

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} \left(\gamma\lambda \right)^l \delta_{t+l}^V = \delta_t^V + (\gamma\lambda)\delta_{t+1}^V + (\gamma\lambda)^2 \delta_{t+2}^V + \cdots + (\gamma\lambda)^{T-t+1} \delta_{T+1} \tag{20}$$

Here, γ is the discount factor, λ *is* the GAE parameter, and δ_t^V is the temporal difference function:

$$\delta_t^V = r_t + \gamma V_\omega(s_{t+1}) - V_\omega(s_t) \tag{21}$$

We utilize its gradient as the loss function for parameter θ:

$$\bigtriangledown J^{\theta'}(\theta) = E_{\pi_{\theta'}} \left[\frac{\pi_\theta(s,a)}{\pi_{\theta'}(s,a)} R(s,a) \bigtriangledown log \pi_{\theta'}(s,a) \right] \tag{22}$$

In our experiments, we observed that fully connected layers exhibit relatively weak fitting capabilities for the relationship between states and actions and they can often lead to the problem of gradient explosion. Therefore, we made some improvements to the PPO algorithm. We introduced a learning rate decay technique, which can enhance the stability of training in the later stages and improve the training effectiveness. Here, we employed linear learning rate decay, where the Actor Network's learning rate decreases linearly from an initial value of 1×10^{-4} to 0 as the training steps progress. This is formulated as:

$$\alpha_{new} = \alpha_{initial} \times (\text{decay factor})^{\frac{\text{iteration}}{\text{decay step}}}. \tag{23}$$

where α presents the parameter operated. By default, PPO uses the ReLU activation function, but experimental findings suggest that PPO performs more effectively with the Tanh activation function. Therefore, we replaced ReLU with the Tanh activation function.

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{24}$$

In the feedforward neural network (FNN), we added a layer normalization layer (LN), following the formula [55]:

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2} \tag{25}$$

After these adjustments, the issue of gradient explosion has been significantly alleviated. The pseudocode of the algorithm is given in Algorithm 1 and a flow chart is given in Figure 2.

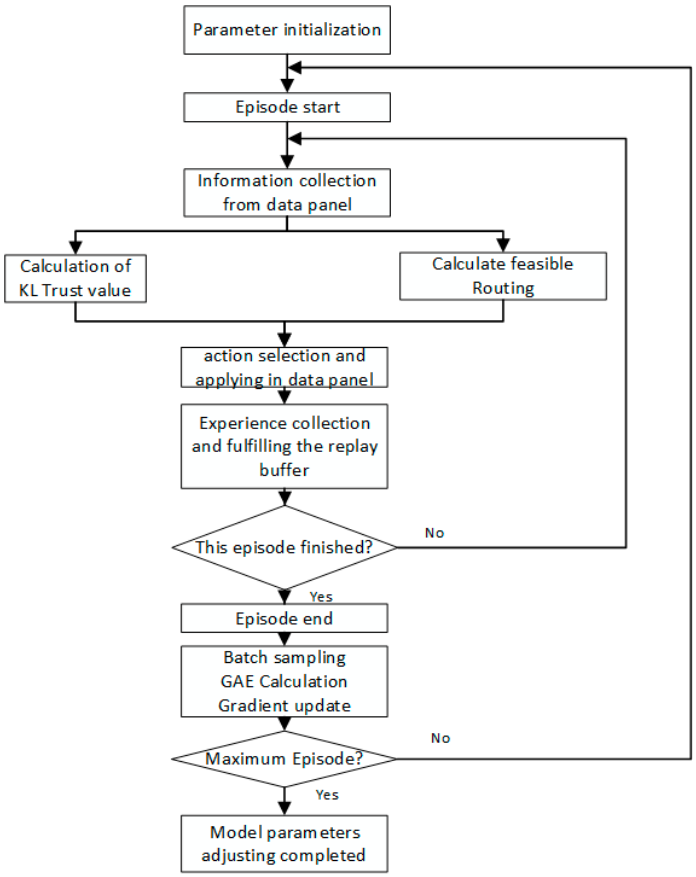


Figure 2. Flowchart of parameter tuning and model establishing.

Algorithm 1: TBPPO Routing Process

Input: M_{topo}, $Group(0)$.
Output: Paths for current network.
Initialization: θ_{actor}, θ_{critic}.
Using M_{topo} as input, perform the DFS algorithm to obtain the preselected path set $\mathbb{L}$.
For $\tau = 1, \ldots\ldots, \mathrm{T}$ **do**
  Obtain $Group(\tau, v_i)$, $M(\tau)$, $S(\tau)$ from the environment under the policy π_{att}.
  Calculate trust values $Trust(\tau, v_i)$ for all nodes using $Group(\tau, v_i)$ and $Group(0, v_i)$ as input.
Start DRL algorithm:
    While next state is not final state **do**
      Actor network selects action a_l
      Calculate the reward $R_{a_l}(\tau)$ for this action.
      Store the current experience in the experience replay buffer.
    End
    Sample a batch from the experience replay buffer.
    Calculate the TD error.
    Calculate the advantage function $\hat{A}_t^{GAE(\gamma,\lambda)}$
    For epoch = 1, ……, 10 **do**
      Perform proximal policy optimization $J_{PPO}^{\theta^k}(\theta)$
      Update gradients for Actor and Critic networks.
    End
  End
End

5. Results and Validation

In this section, we thoroughly examine the details of the experiment and the results to assess the reliability of our proposed system. We compare the performance of the TBPPO algorithm with three other algorithms: DRSIR, PPO, and Dijkstra. Additionally, we explore

the performance of TBDRSIR, which integrates the DRSIR algorithm with our trust value mechanism within the network topology.

5.1. Experiment Setup

Our experiments were executed on hardware consisting of an Intel (R) Core (TM) i5-9400 CPU running at 2.90 GHz and an NVIDIA GeForce GTX 1660Ti for training the TBPPO agent. We implemented the AC model using PyTorch and optimized the training process using the Adam Optimizer as the loss function. In our experimental model, the state representation includes several components, such as the current KL trust values for each node, node types, node utilization, the current average delay, and the number of remaining steps. The action space in our experiments is constructed from a set of feasible routes calculated using an improved DFS algorithm. The Actor network is designed as a six-layer fully connected neural network, with each layer consisting of 256 neurons and layer normalization applied between these layers. The output size of the network matches the action space size, which is set to 5 in our experimental model. Similarly, the Critic network is constructed as a six-layer fully connected network with 256 neurons in each layer, featuring layer normalization between layers. The output of the Critic network corresponds to the Q-value, resulting in an output size of 1. For a comprehensive understanding of the remaining parameters of our experimental model, please refer to Table 2.

Table 2. Parameters of neural network and reward function.

Parameter Name	Value
Total Episodes	3000
Discount Factorγ	0.96
Learning Rate of Actor	1×10^{-5}
Learning Rate of Critic	1×10^{-4}
Advantage function scaling factor λ	0.90
Truncation parameter ε	0.2
Parameter of delay reward α_{delay}	0.2
Parameter of variation $\alpha_{variation}$	4
Parameter of trust value α_{trust}	-400
Parameter of diversity α_{type}	3
Parameter of repeat α_{repeat}	-100

5.2. Validation of the KL Trust Mechanism

To assess the effectiveness of the KL trust value mechanism, we conducted experiments using the CICIDS2017 dataset [56], which offers insights into packet exchange patterns within a network over a specific time frame. This validation process involved simulating a botnet network with three key stages: probing, propagation, and launching DDoS attacks. These stages encompassed diverse characteristics, such as creating packets with fixed IP addresses but varying TCP destination port numbers, generating packets with a multitude of uniform source addresses but with distinct destination addresses and port numbers, introducing a significant number of unique source addresses accessing a specific destination IP address. Furthermore, our SDN was inundated with SYN packets, but the number of ACK packets did not align with these SYN packets.

The decision to employ KL divergence as a metric for the trust value was driven by our objective to evaluate the impact of attack interference on a node's communication capability, as opposed to classifying the type of attack. KL divergence serves as a suitable measure for gauging the degree to which a node's communication capabilities are affected, making it a pragmatic choice for our purposes. Our methodology involves identifying records in which a node functions as both the source and destination IP without being subjected to any attacks, deeming these records as valid for that particular node. We randomly sampled 3000 such records for each test node. Out of the 84 distinctive features within the CICIDS2017 dataset, we classify a node as malicious if it falls into any of the

following categories: an attacker node itself, a node with identified port vulnerabilities, a node affected by worm infestations, or a node subject to DDoS attacks. As a result, we established four key features as our criteria for classification: source IP frequency, destination IP frequency, SYN packet frequency, and the frequency of distinct destination ports. These criteria were utilized as benchmarks for calculating the KL divergence. It is worth noting that the data volume for the number of distinct ports recorded (DP) may vary across different nodes. Therefore, we combined attack records with a portion of normal communication records in varying proportions and subsequently calculated their trust values. The weightings assigned to the parameters, namely SIP (Source IP), DIP (Destination IP), SYN, and DP (Distinct Ports), are detailed in Table 3.

Table 3. β of advanced KL-Divergence.

SIP Parameter β_1	DIP Parameter β_2	SYN Parameter β_3	DP Parameter β_4
0.8	1.0	2.0	0.0001

In the experiment, several nodes were tested, including one with the IP address 172.16.0.1. The test dataset included a mix of DDoS attack records and regular communication, with proportions of 20%, 40%, and 60%, respectively. The results are presented in Table 4. As can be observed, there is a positive correlation between the KL trust value and the proportion of DDoS attacks. This result demonstrates that the KL trust value can reflect the security performance of a given node.

Table 4. The relationship between KL trust value and DDoS attack ratio of tested node.

DDoS Attack Proportion	KL Trust Value
20%	0.069
40%	0.167
60%	0.298

5.3. Performance Comparison

5.3.1. Baselines

The TBPPO algorithm was evaluated in comparison to the DRSIR [21], PPO [22], and Dijkstra [13] algorithms. Furthermore, we integrated the trust mechanism introduced in our paper with DRSIR to form TBDRSIR and conducted comparative evaluations.

(1) Dijkstra Algorithm [13]: The Dijkstra algorithm is based on the weights of a graph to select paths. This algorithm is simple and feasible and is an important component of the OSPF protocol. We hope to use this algorithm to demonstrate the deviation of our algorithm's delay portion from the theoretical optimal value and to evaluate its security performance.

(2) DRSIR Algorithm [21]: The DRSIR algorithm is based on DQN's deep reinforcement learning algorithm and its effectiveness has been thoroughly demonstrated in [21]. However, it does not explore multiple objectives. We aim to use this algorithm to showcase our algorithm's optimization ability for multiple objectives.

(3) TBDRSIR (Trust-Based DRSIR): TBDRSIR is a variant of our ablation experiment introduced to explore the performance of TBPPO. It draws inspiration from DRSIR and combines the dynamic security assessment capability we designed. We aim to test the performance of our security mechanism on other algorithms.

(4) PPO Algorithm [22]: The PPO algorithm has been used to solve routing optimization problems in SDNs in [22]. Its advantage lies in its adaptability to diverse environments and good performance in terms of delay. However, the authors lacked consideration for security capabilities. Additionally, we have made improvements to PPO, enabling it to achieve a superior performance. We hope to use this algorithm as a baseline to

demonstrate our algorithm's superior security performance and better performance in terms of delay variation.

5.3.2. Network Topologies

These experiments were performed in two distinct network topologies: NSFNet-14 [57] and Germany-50 [58]. NSFNet-14 consists of 14 nodes and 19 undirected edges, with an initial route count of 61 determined using the DFS algorithm. In contrast, the Germany-50 network comprises 50 nodes and 176 undirected edges, with an initial route count (K) capped at 10,000 and calculated using an enhanced DFS algorithm. Figure 3 shows the design and network topology of NSFNet-14 and Germany-50.

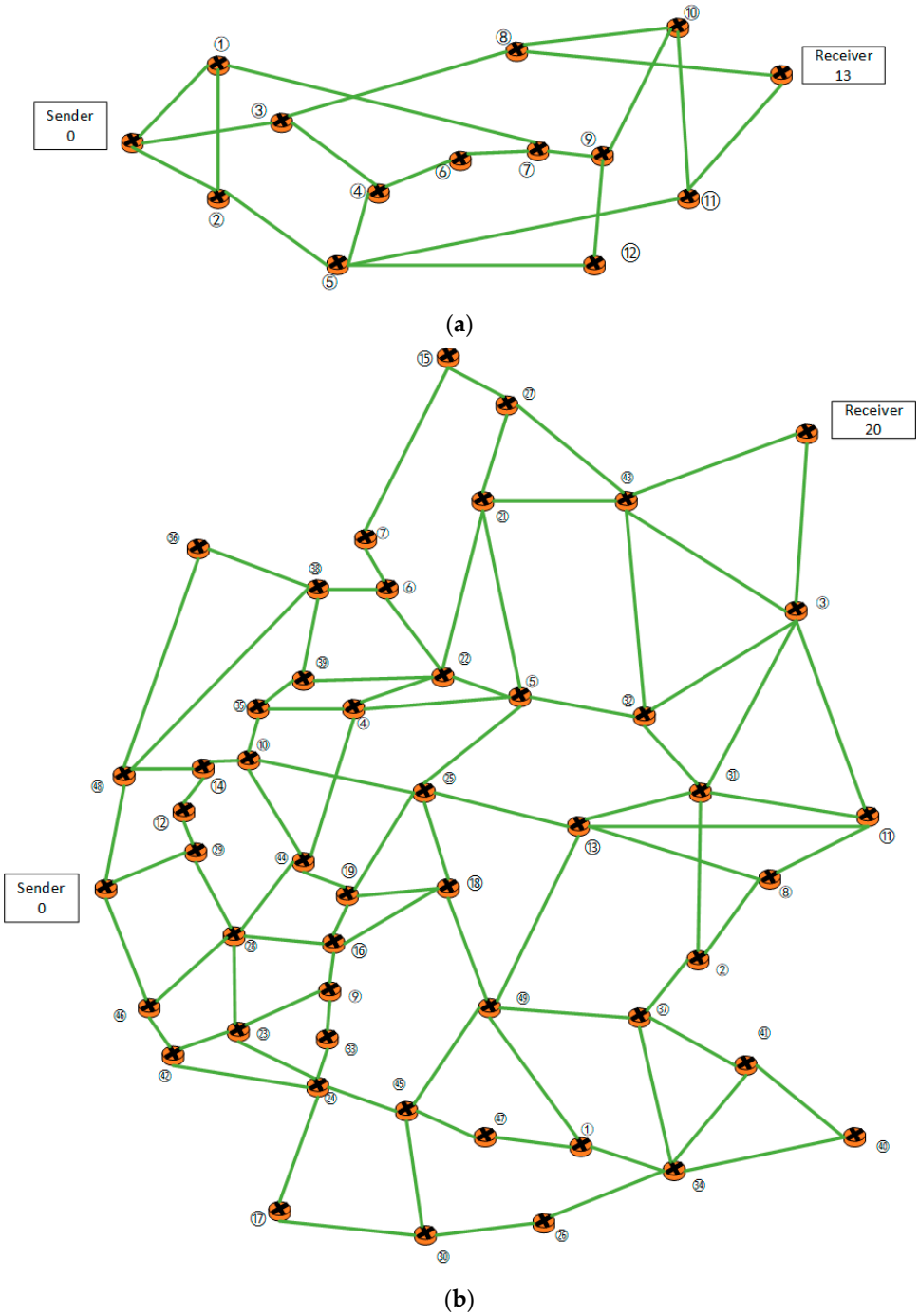


Figure 3. The design and network topology of NSFNet-14 and Germany-50: (**a**) NSFNet-14, and (**b**) Germany-50.

5.3.3. Convergence Comparison

Figure 4 shows the convergence performance comparison in the NSFNet-14 and Germany-50 topologies after 3000-episode iterations of simulation, as represented by the deep reinforcement learning metric of reward convergence. It is worth noting that because PPO and DRSIR do not incorporate our security mechanism, their rewards will lack the corresponding negative values of $r_{safety}(\tau)$ and $r_t(\tau)$, which leads to a certain numerical difference in the performance of TBPPO versus PPO and TBDRSIR versus DRSIR. Figure 4a shows the rewards collected per cycle in the NSFNet-14 topology. After 3000-episode iterations, TBPPO converged to -20 after 800 episodes, while PPO initially reached -20 after 500-episode iterations and later re-converged to -12 after 2700 episodes, both exhibiting good convergence trends. DRSIR showed poorer convergence, quickly rising to -60 in the first 100-episode iterations and fluctuating widely until the end of 3000 episodes. In contrast, TBDRSIR displayed greater volatility and ultimately failed to converge. These findings indicate that PPO is more suited to mechanisms that combine the trust value and the nodes' diversity than DQN is.

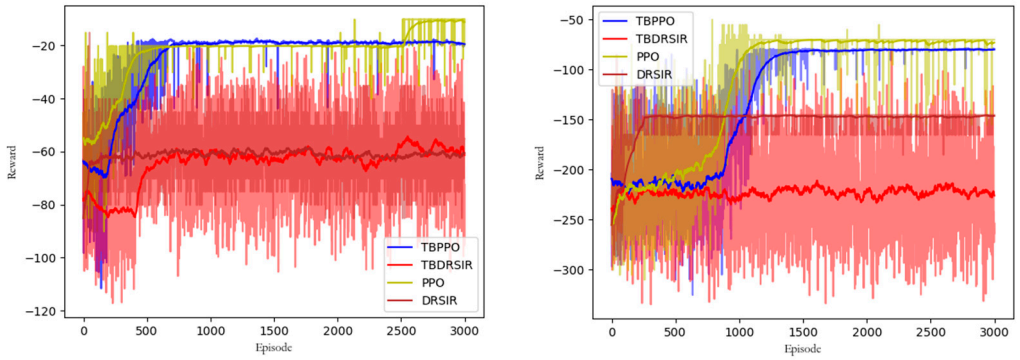


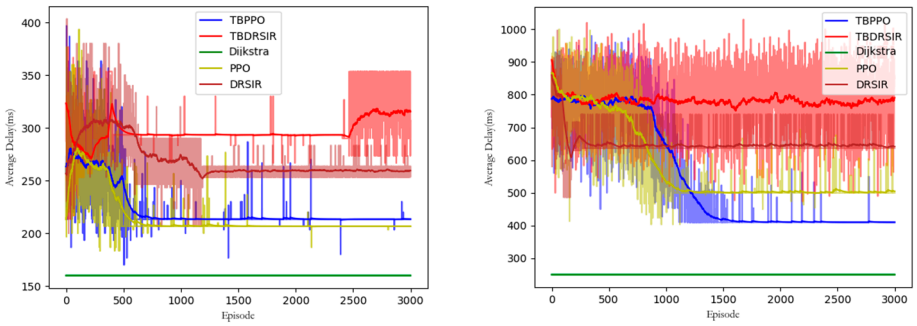
Figure 4. Performance comparison of reward of DRL algorithms: (**a**) NSFNet-14, and (**b**) Germany-50.

In Figure 4b, rewards collected per cycle in the Germany-50 network topology are presented, which offers more potential routes than NSFNet-14. According to the reward comparison in the figure, the convergence patterns of various algorithms can be observed. TBPPO starts to approach -85 after roughly 1400-episode iterations, while PPO achieves convergence at -75 after just 1200-episode iterations. DRSIR rapidly climbs to around -150 within the first 300-episode iterations, but exhibits higher volatility compared to PPO-based algorithms. In contrast, TBDRSIR does not reach convergence. Clearly, TBPPO consistently performs well on larger networks, while TBDRSIR seems to show an inferior performance in our analysis, possibly due to the inclusion of the trust mechanism, which appears to negatively impact DQN's performance. Compared to NSFNet-14, both PPO and TBPPO continue to show superior convergence and optimization capabilities.

5.3.4. Delay Comparison

Figure 5 illustrates the delay performance comparison in the NSFNet-14 and Germany-50 topologies conducted following the simulation of 3000-episode iterations. Figure 5a shows the delay performance in the NSFNet-14 topology. After 3000-episode iterations, TBPPO, TBDRSIR, PPO, and DRSIR divided packets into multiple routes, while Dijkstra only used the shortest path with a fixed delay of 160 ms. TBPPO converged to 223.3 ms after 600-episode iterations, and PPO reached 216.67 ms after 500-episode iterations in terms of average delay. DRSIR displayed gradual convergence, approaching around 260 ms after 1200-episode iterations, albeit with some noticeable fluctuations. In contrast, TBDRSIR initially demonstrated convergence to 300 ms, but experienced a sudden divergence around 2700-episode iterations, ultimately failing to converge. These findings indicate that in the context of low-delay network exploration, PPO outperforms DQN.

Additionally, TBPPO, which needs to balance both exploration and security considerations while also not significantly surpassing PPO, remains competitive. Furthermore, when compared to the theoretically optimal Dijkstra, both TBPPO and PPO exhibit a relatively similar performance.



(**a**) (**b**)

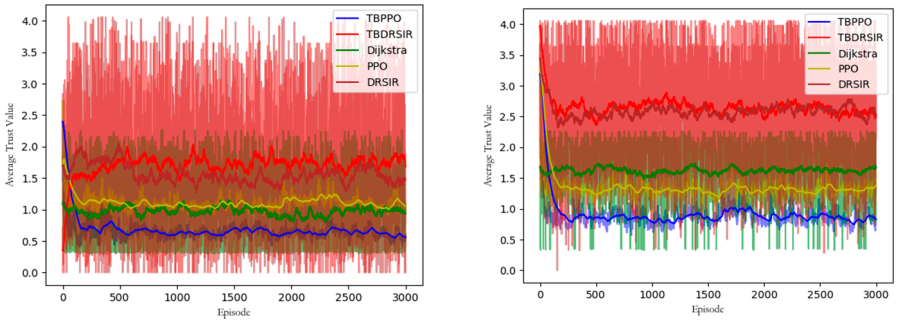
Figure 5. Performance comparison of the average time delay of five algorithms: (**a**) NSFNet-14, and (**b**) Germany-50.

In Figure 5b, the delay performance in the Germany-50 network topology is presented. This network comprises 50 nodes and 176 edges, resulting in a larger number of possible routes compared to NSFNet-14. To address resource constraints, the paths were sorted by delay, with the top 10,000 routes retained. The action space for testing the algorithm consisted of these 10,000 routes. After a 3000-cycle simulation, TBPPO, TBDRSIR, PPO, and DRSIR employed a data packet division strategy using three distinct paths for the data transmission. In contrast, the Dijkstra algorithm, in its quest for the shortest route within the network topology, funneled all traffic along that singular path. Within the simulated network configuration, only one shortest path with a 160 ms delay was available, which ultimately led to Dijkstra maintaining a consistent average delay of 250 ms. According to the delay comparison in the figure, the convergence patterns of various algorithms can be observed. TBPPO starts to approach 423.3 ms after roughly 1400-episode iterations, while PPO achieves convergence at 503.3 ms after 1200-episode iterations. DRSIR begins to approach the 657 ms mark after 300-episode iterations, but its performance continues to fluctuate. In contrast, TBDRSIR does not reach convergence. It is evident that TBPPO consistently performs well on large networks, while TBDRSIR, in our analysis, seems to exhibit an inferior performance, possibly due to the inclusion of the trust mechanism, which appears to negatively impact DQN's performance. When compared to NSFNet-14, PPO and TBPPO still showcase superior convergence and optimization capabilities. Moreover, as the network scales up, PPO's advantage in securing routing becomes more apparent, resulting in lower delay due to fewer instances of attacks.

5.3.5. Trust Value Comparison

Figure 6 illustrates a comparative analysis of the KL trust values across five algorithms. Specifically, in the context of the NSFNet-14 dataset, as shown in Figure 6a, it is evident that the Dijkstra algorithm fails to take trust values into account. Consequently, it directs all network traffic along a single path, leading to a consistently static KL trust value of approximately 1. This static value serves as a clear indicator of the Dijkstra algorithm's inability to achieve routing convergence. The TBPPO algorithm, on the other hand, displays a noticeable convergence trend and eventually stabilizes at around 0.5, with intermittent fluctuations hovering around 0.2. In contrast, the PPO algorithm, which does not explicitly address security concerns, exhibits a performance similar to Dijkstra. Both the DRSIR and TBDRSIR algorithms exhibit inadequate adaptation to dynamic DDoS attack defense scenarios, failing to achieve trust value convergence. As a result, the TBPPO algorithm

consistently maintains lower KL trust values, suggesting a reduced vulnerability to attacks and an enhanced security performance.



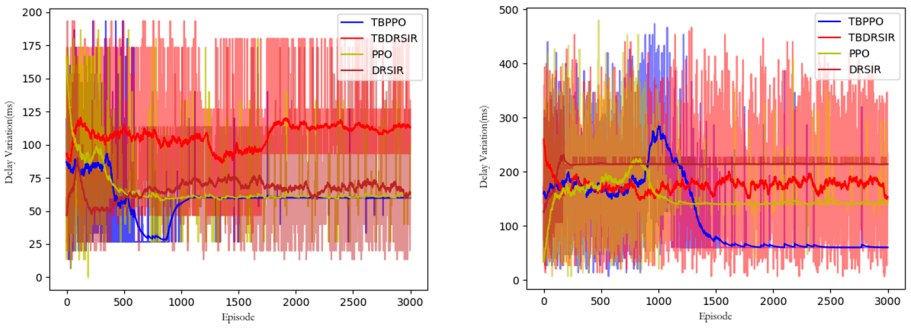
(**a**) (**b**)

Figure 6. Performance comparison of the trust value of five algorithms: (**a**) NSFNet-14, and (**b**) Germany-50.

Figure 6b compares the performance of the KL trust values on Germany-50. Much like the scenario in NSFNet-14, the Dijkstra algorithm does not consider trust values when assessing its performance. It channels all traffic through a single path, resulting in a consistently stable KL trust value of around 1.5 without achieving convergence. Examining the graph, it becomes apparent that the TBPPO algorithm exhibits a noticeable trend towards convergence, although it experiences more fluctuations compared to NSFNet-14. Eventually, it stabilizes at approximately 0.8, with fluctuations hovering around 0.4. In contrast, PPO reaches convergence at 1.3. Regrettably, both the DRSIR and TBDRSIR algorithms fail to achieve trust value convergence. This result shows the capability of the TBPPO model to identify relatively secure routes in a large network environment.

5.3.6. Delay Variation Comparison

In Figure 7, the performance of different routing algorithms in terms of delay variation is compared. Dijkstra, which does not take into account delay variation attributes, is not part of the comparison. A comparison is made between TBPPO, TBDRSIR, PPO, and DRSIR. On the NSFNet-14 data, as depicted in Figure 7a, the TBPPO algorithm starts to converge around 60 ms after roughly 1300-episode iterations, showing a clear trend towards convergence. PPO, on the other hand, converges around 60 ms after 600-episode iterations. TBDRSIR reaches approximately 125 ms after 1700-episode iterations, with significant fluctuations. DRSIR, however, fails to converge, with an average delay of around 70 ms. According to the shaded area in the graph, it continues to fluctuate between 15 ms and 175 ms even after about 3000-episode iterations. These results suggest that TBPPO, while emphasizing security, maintains a competitive delay performance compared to PPO.



(**a**) (**b**)

Figure 7. Performance comparison on delay inequality: (**a**) NSFNet-14, and (**b**) Germany-50.

In our performance analysis of the Germany-50 data, when comparing delay variations (as shown in Figure 7b), we evaluated TBPPO, TBDRSIR, PPO, and DRSIR. Dijkstra was omitted from this comparison due to its lack of multiple delay-aware paths. As illustrated in Figure 7b, the TBPPO algorithm exhibits a clear convergence trend, starting to converge around the 1500th iteration to an approximate 50 ms delay. In contrast, PPO achieves convergence earlier, at around 1000-episode iterations, with an average delay of approximately 140 ms. TBDRSIR fails to reach convergence, while DRSIR exhibits more noticeable convergence at around 210 ms. Interestingly, even though TBPPO prioritizes security, its delay variation performance is not significantly inferior to PPO. In the context of Germany, TBPPO demonstrates the benefits of secure route discovery by reducing the number of attacks, resulting in lower delay compared to PPO and showcasing improved real-time communication capabilities.

5.3.7. Node Diversity Analysis

In Figure 8, we performed an extensive examination of node diversity across different routing algorithms. Specifically, when observing the NSFNet-14 (see Figure 8a), the Dijkstra algorithm consistently maintains a fixed path for the traffic, resulting in a constant node diversity value of 11. In contrast, the TBPPO algorithm displays an intriguing behavior, as it gradually converges to a node diversity of 8 over approximately 600-episode iterations, demonstrating a clear convergence trend. The PPO algorithm also exhibits a commendable convergence performance but stabilizes at a node diversity value of 15. Conversely, the TBDRSIR and DRSIR algorithms both reach a node diversity of 13, albeit with relatively significant fluctuations. These findings indicate that the TBPPO algorithm places greater emphasis on optimizing the composition of routing nodes compared to the other algorithms. Moreover, the TBPPO algorithm distinguishes itself by offering notable advantages over traditional Dijkstra and PPO approaches concerning security. Additionally, it outperforms the DRSIR algorithm and TBDRSIR in terms of stability and various performance metrics. The algorithm excels in providing multiple dependable routing options in network topologies with consistent structures dynamically changing the link conditions, striking a balance between link delay and security.

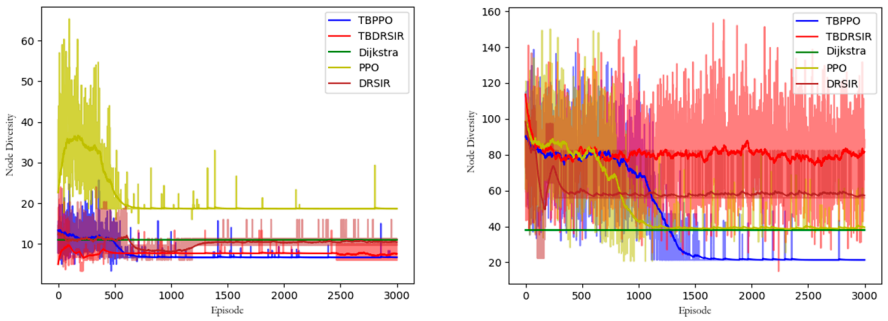


Figure 8. Performance comparison of diversity of nodes of five algorithms: (**a**) NSFNet-14, and (**b**) Germany-50.

In Figure 8b of the Germany-50 data, the Dijkstra algorithm consistently follows a fixed path, maintaining a constant node diversity value of 38. On the other hand, the TBPPO algorithm begins to converge towards a node diversity of 23 after about 1500-episode iterations, displaying a clear convergence pattern. PPO, which does not take into account node types, shows a performance similar to Dijkstra. Both TBDRSIR and DRSIR algorithms still exhibit relatively large fluctuations in node diversity. These findings imply that the TBPPO algorithm places greater emphasis on the composition of routing nodes, thereby improving its ability to prevent failures in similar nodes when sudden events occur from operators.

5.3.8. Multi-Path Routing Performance

In Figure 9, we executed a comprehensive evaluation of the multi-path routing performance of the TBPPO algorithm. This assessment involved an investigation into the effectiveness of TBPPO under various output settings, specifically producing 3, 4, and 5 route paths. We employed a range of essential performance metrics including but not limited to average delay, average trust value, delay variation, and node diversity.

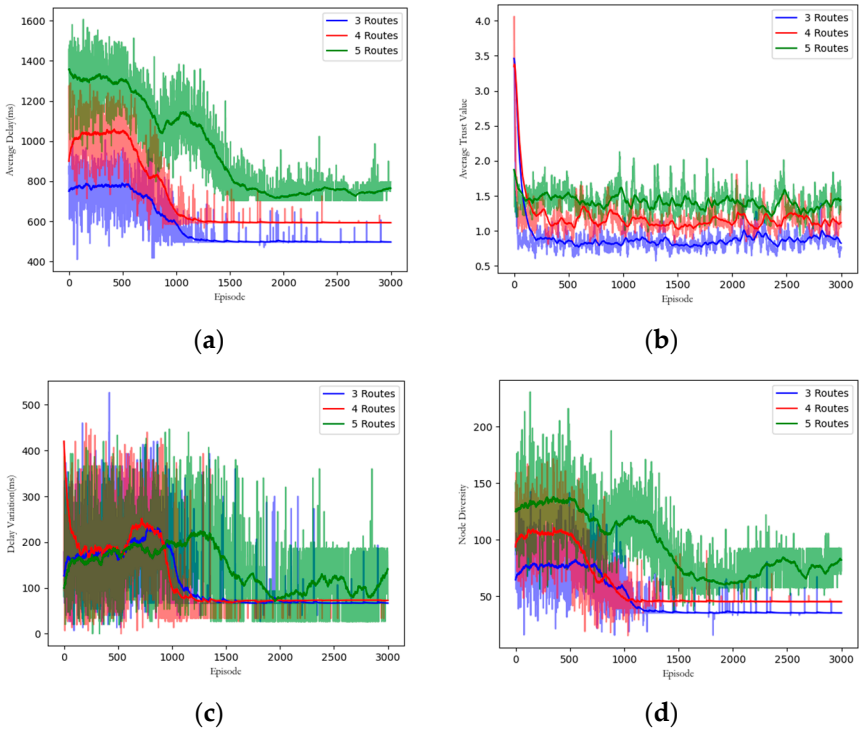


Figure 9. Performance of TBPPO with respect to the quantity of available paths: (**a**)average delay, (**b**) average trust value, (**c**)delay variation, and (**d**) node diversity.

In Figure 9a, in terms of the average delay performance, three routes achieved the best average delay at 500 milliseconds, which was 100 milliseconds faster than the 600 milliseconds of four routes. On the other hand, five routes were slower than four routes by about 200 milliseconds at 800 milliseconds. Both three routes and four routes achieved convergence, with similar convergence speeds. However, under five routes, the algorithm exhibited larger fluctuations but still showed some degree of convergence.

In Figure 9b, regarding the trust value performance, three routes continued to achieve the best trust value at around 0.7, while four routes closely followed with 1.2. In contrast, five routes had the poorest performance, with a trust value of around 1.5. The convergence performance among these three scenarios exhibited slight variations.

In Figure 9c, in terms of the delay variation performance, four routes showed similar results to three routes, with both achieving an average delay variation of around 60 milliseconds with clear convergence trends. However, five routes did not converge to the same extent, resulting in an average delay variation of around 100 milliseconds.

In Figure 9d, with respect to node diversity, four routes exhibited a comparable trend to three routes, with both converging at approximately 50, as opposed to the 40 for three routes. In contrast, five routes ultimately converged at around 80. When comparing their convergence characteristics, three routes and four routes demonstrated a robust performance, indicating strong convergence abilities. On the other hand, five routes exhibited some degree of convergence; however, it displayed a wider range of fluctuations. These findings suggest that TBPPO exhibited good performance with both three routes and four routes displaying strong convergence capabilities. However, its performance declined

when using five routes compared to three and four routes. Therefore, we recommend not selecting multi-path routing tasks with more than four routes when using TBPPO.

6. Conclusions

This paper presents the TBPPO algorithm, a deep reinforcement learning-based approach that addresses the limitations of existing intelligent routing algorithms in SDN environments. TBPPO leverages a unique trust value mechanism based on KL divergence and optimizes DFS and PPO algorithms to establish secure, low-delay routing solutions. Initially, it refines the DFS algorithm for route selection, effectively reducing the complexity of the reinforcement learning network's action space and avoiding routing loops. Furthermore, it employs an improved KL divergence algorithm to estimate the trust values related to potential node attacks and a node diversity assessment method to estimate the ISP balance, providing an easy-to-calculate security assessment method for this algorithm. Lastly, the algorithm enhances PPO to explore multiple performance-balanced paths, enabling diverse network configurations and communication pairs to select routes that align with the network characteristics. We demonstrated the effectiveness of the TBPPO algorithm compared to cutting-edge methods in medium to large network topologies. The result show that, in large networks, TBPPO with security mechanisms experienced fewer attacks in route selection compared to PPO. Both its delay and variation performance were reduced by about 100 ms, its trust value led by 0.5, and the diversity of nodes was ahead by 20. Moreover, when comparing TBPPO with PPO and TBDRSIR, PPO still converges stably under complex multi-objective dynamic conditions. This result can be attributed to the excellent stability of PPO. In multi-route performance tests, we discovered that when TBPPO is used with more than four routes, its performance deteriorates significantly; therefore, we recommend applying TBPPO to three or four route path tasks to showcase its best capabilities. We hope that TBPPO can offer a better multi-path secure routing optimization method for real-time communication, such as in scenarios like online classes, streaming videos, live web broadcasts, and real-time meetings. In future research, we will focus on the application of the algorithm in heterogeneous network scenarios, making it adaptable to a wider range of practical application scenarios, like smart grid. We plan to introduce the graph convolutional networks (GCNs) into the DRL process for a better performance.

Author Contributions: Conceptualization, Y.Z., L.Q. and Z.W.; methodology, Z.W.; software, Y.X.; validation, X.W. and S.W.; formal analysis, A.P.; investigation, Z.W.; resources, Y.Z.; data curation, S.W.; writing—original draft preparation, S.W.; writing—review and editing, L.Q. and A.P.; visualization, Y.X. and X.W.; supervision, Z.W.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Science and Technology Project of State Grid Zhejiang Electroic Power Co., grant number B311XT230019.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Conflicts of Interest: The authors Yi Zhang, Lanxin Qiu, Yangzhou Xu and Xinjia Wang were employed by the company Information Communication Branch of State Grid Zhejiang Electric Power Co., Hangzhou 310007, China. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The funder had no role in the design of the study; in the collection, analysis, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

References

1. Scott-Hayward, S.; Natarajan, S.; Sezer, S. A Survey of Security in Software Defined Networks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 623–654. [CrossRef]
2. Alsmadi, T.; Alqudah, N. A Survey on malware detection techniques. In Proceedings of the 2021 International Conference on Information Technology (InCIT), Amman, Jordan, 14–15 July 2021; pp. 371–376. [CrossRef]
3. Yoo, Y.; Yang, G.; Lee, J.; Shin, C.; Kim, H. TeaVisor: Network Hypervisor for Bandwidth Isolation in SDN-NV. *IEEE Trans Cloud Comput.* **2023**, *11*, 2739–2755. [CrossRef]
4. Pizzutti, M.; Schaeffer-Filho, A.E. An Efficient Multipath Mechanism Based on the Flowlet Abstraction and P4. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–6. [CrossRef]
5. Jin, H.; Yang, G. TALON: Tenant Throughput Allocation Through Traffic Load-Balancing in Virtualized Software-Defined Networks. In Proceedings of the International Conference on Information Networking (ICOIN), Kuala Lumpur, Malaysia, 9–11 January 2019; pp. 233–238. [CrossRef]
6. Shen, L.; Wu, M.; Zhao, M. Secure Virtual Network Embedding Algorithms for a Software-Defined Network Considering Differences in Resource Value. *Electronics* **2022**, *11*, 1662. [CrossRef]
7. Klöti, R.; Kotronis, V.; Smith, P. OpenFlow: A security analysis. In Proceedings of the 2013 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 7–10 October 2013; pp. 1–6. [CrossRef]
8. Perrig, A.; Szewczyk, R.; Tygar, J. SPINS: Security Protocols for Sensor Networks. *Wirel. Net.* **2002**, *8*, 521–534. [CrossRef]
9. Karlof, C.; Wagner, D. Secure routing in wireless sensor networks: Attacks and countermeasures. In Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA), Anchorage, AK, USA, 11 May 2003; pp. 113–127. [CrossRef]
10. Zhou, L.; Haas, Z.J. Securing ad hoc networks. *IEEE Netw.* **1999**, *13*, 24–30. [CrossRef]
11. Buchegger, S.; Boudec, J.Y.L. Performance analysis of the CONFIDANT protocol. In Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing (MobiHoc), New York, NY, USA, 9–11 June 2003; pp. 226–236. [CrossRef]
12. Ali-Eldin, A.M.T. A cloud-based trust computing model for the social Internet of Things. In Proceedings of the 2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), Cairo, Egypt, 9 May 2021; pp. 161–165. [CrossRef]
13. Suryani, V.; Widyawan, S. A survey on trust in Internet of Things. In Proceedings of the 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, 5–6 October 2016; pp. 1–6. [CrossRef]
14. Gautam, A.K.; Kumar, R. A comprehensive study on key management. *Authen. Trust Manag. Tech. Wirel. Sens. Net.* **2021**, *3*, 50. [CrossRef]
15. Khan, W.Z.; Arshad, Q.; Hakak, S. Trust Management in Social Internet of Things: Architectures, Recent Advancements, and Future Challenges. *IEEE Internet Things J.* **2021**, *8*, 7768–7788. [CrossRef]
16. Pourghebleh, B.; Wakil, K.; Navimipour, N.J. A Comprehensive Study on the Trust Management Techniques in the Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 9326–9337. [CrossRef]
17. Jiang, J.R.; Huang, H.W.; Liao, J.H.; Chen, S.Y. Extending Dijkstra's shortest path algorithm for software defined networking. In Proceedings of the 16th Asia-Pacific Network Operations and Management Symposium (APNOMS), Hsinchu, Taiwan, 17–19 September 2014; pp. 1–4. [CrossRef]
18. Wu, Y.J.; Hwang, P.C.; Hwang, W.S. Artificial Intelligence Enabled Routing in Software Defined Networking. *Appl. Sci.* **2020**, *10*, 6564. [CrossRef]
19. Mnih, V.; Kavukcuoglu, K.; Silver, D. Playing Atari with deep reinforcement learning. In Proceedings of the Neural Information Processing Systems Conference and Workshops (NIPS), Lake Tahoe, CA, USA, 5–10 December 2013; pp. 529–536. [CrossRef]
20. Schulman, J.; Moritz, P.; Levine, S. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv* **2015**, arXiv:1506.02438. [CrossRef]
21. Xiang, J.; Li, Q.; Dong, X. Continuous Control with Deep Reinforcement Learning for Mobile Robot Navigation. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019; pp. 1501–1506. [CrossRef]
22. Schulman, J.; Levine, S.; Moritz, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; pp. 1889–1897. [CrossRef]
23. Du, J.; Zhang, C.; He, S. Learning-Based Congestion Control Assisted by Recurrent Neural Networks for Real-Time Communication. In Proceedings of the 2023 IEEE Symposium on Computers and Communications (ISCC), Gammarth, Tunisia, 9–12 July 2023; pp. 323–328. [CrossRef]
24. Chen, J.; Xiao, Z.; Xing, H. STDPG: A Spatio-Temporal Deterministic Policy Gradient Agent for Dynamic Routing in SDN. In Proceedings of the 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
25. Casas-Velasco, D.M.; Rendon, O.M.C.; da Fonseca, N.L.S. DRSIR: A Deep Reinforcement Learning Approach for Routing in Software-Defined Networking. *IEEE Trans. Net. Serv. Manag.* **2021**, *19*, 4807–4820. [CrossRef]
26. Alkhalaf, S.; Alturise, F. A novel method for routing optimization in software-defined networks. *Comput. Mat. Cont.* **2022**, *73*, 6393–6405. [CrossRef]
27. Kullback, S.; Leibler, R.A. The Information in Distributions. *Ann. Math. Stat.* **1951**, *22*, 79–86. [CrossRef]

28. Iqbal, A.; Zubair, M.; Khan, M.A.; Ullah, I.; Ur-Rehman, G.; Shvetsov, A.V.; Noor, F. An Efficient and Secure Certificateless Aggregate Signature Scheme for Vehicular Ad hoc Networks. *Future Internet* **2023**, *15*, 266. [CrossRef]
29. Abidi, R.; Azzouna, N.B. Self-adaptive trust management model for social IoT services. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 31 October–2 November 2021; pp. 1–7. [CrossRef]
30. Magdich, R.; Jemal, H.; Nakti, C. An efficient trust related attack detection model based on machine learning for social Internet of Things. In Proceedings of the 2021 International Wireless Communications and Mobile Computing (IWCMC), Harbin City, China, 28 June–2 July 2021; pp. 1465–1470. [CrossRef]
31. Amiri-Zarandi, M.; Dara, R.A.; Fraser, E. LBTM: A lightweight blockchain-based trust management system for social Internet of Things. *J. Supercom.* **2022**, *78*, 8302–8320. [CrossRef]
32. Wang, X.; Zhong, X.; Li, L. TOT: Trust aware opportunistic transmission in cognitive radio social Internet of Things. *Comput. Commun.* **2020**, *162*, 1–11. [CrossRef]
33. Farahbakhsh, B.; Fanian, A.; Manshaei, M.H. TGSM: Towards trustworthy group-based service management for social IoT. *Internet Things* **2021**, *13*, 100312. [CrossRef]
34. Ashwin, M.; Kamalraj, S.; Azath, M. Weighted Clustering Trust Model for Mobile Ad Hoc Networks. *Wirel. Pers. Commun.* **2017**, *94*, 2203–2212. [CrossRef]
35. Rajeswari, A.R.; Kulothungan, K.; Ganapathy, S. A trusted fuzzy based stable and secure routing algorithm for effective communication in mobile adhoc networks. *Peer-to-Peer Net. Appl.* **2019**, *12*, 1076–1096. [CrossRef]
36. Zhang, T.; Yan, L.; Yang, Y. Trust evaluation method for clustered wireless sensor networks based on cloud model. *Wirel. Net.* **2016**, *24*, 777–797. [CrossRef]
37. Mingwu, Z.; Bo, Y.; Yu, Q. Using Trust Metric to Detect Malicious Behaviors in WSNs. In Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD), Qingdao, China, 30 July–1 August 2007; pp. 104–108. [CrossRef]
38. Sagar, S.; Mahmood, A.; Sheng, M. Towards a Machine Learning-driven Trust Evaluation Model for Social Internet of Things: A Time-aware Approach. In Proceedings of the 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous), New York, NY, USA, 7–9 December 2020; pp. 283–290. [CrossRef]
39. Sagar, S.; Mahmood, A.; Sheng, Q.Z.; Zhang, W.E. Trust Computational Heuristic for Social Internet of Things: A Machine Learning-based Approach. In Proceedings of the IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6. [CrossRef]
40. Marche, C.; Nitti, M. Trust-Related Attacks and Their Detection: A Trust Management Model for the Social IoT. *IEEE Trans. Net. Serv. Manag.* **2020**, *18*, 3297–3308. [CrossRef]
41. Jafarian, B.; Yazdani, N.; Haghighi, M.S. Discrimination-aware trust management for social internet of things. *Comp. Net. Inter. J. Comp. Tel. Net.* **2020**, *178*, 11. [CrossRef]
42. Abdelghani, W.; Amous, I.; Zayani, C.A.; Sèdes, F. Dynamic and scalable multi-level trust management model for Social Internet of Things. *J. Supercomput.* **2022**, *78*, 8137–8193. [CrossRef]
43. Abadi, A.F.E.; Asghari, S.A.; Marvasti, M.B.; Abaei, G. RLBEEP: Reinforcement-Learning-Based Energy Efficient Control and Routing Protocol for Wireless Sensor Networks. *IEEE Access* **2022**, *10*, 44123–44135. [CrossRef]
44. Chen, C.; Xue, F.; Lu, Z. RLMR: Reinforcement Learning Based Multipath Routing for SDN. *Wirel. Commun. Mob. Comp.* **2022**, *2022*, 5124960. [CrossRef]
45. Yao, H.; Mai, T.; Xu, X.; Zhang, P.; Li, M. NetworkAI: An intelligent network architecture for self-learning control strategies in software defined networks. *IEEE Internet Things J.* **2018**, *5*, 4319–4327. [CrossRef]
46. Chen, Y.R.; Rezapour, A.; Tzeng, W.G. RL-Routing: An SDN Routing Algorithm Based on Deep Reinforcement Learning. *IEEE Trans. Net. Sci. Eng.* **2020**, *7*, 3185–3199. [CrossRef]
47. Li, Z.; Wang, X.; Pan, L. Network Topology Optimization via Deep Reinforcement Learning. *IEEE Trans. Commun.* **2022**, *71*, 2847–2859. [CrossRef]
48. Guo, X.; Lin, H.; Li, Z.; Peng, M. Deep-Reinforcement-Learning-Based QoS-Aware Secure Routing for SDN-IoT. *IEEE Internet Things J.* **2020**, *7*, 6242–6251. [CrossRef]
49. Wang, J.; Liu, J.; Guo, H.; Mao, B. Deep Reinforcement Learning for Securing Software-Defined Industrial Networks with Distributed Control Plane. *IEEE Trans. Ind. Infor.* **2022**, *18*, 4275–4285. [CrossRef]
50. Xu, Q.; Zhang, Y.; Wu, K.; Wang, J. Evaluating and Boosting Reinforcement Learning for Intra-Domain Routing. In Proceedings of the IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Monterey, CA, USA, 4–7 November 2019; pp. 265–273. [CrossRef]
51. Douligeris, C.; Mitrokotsa, A. DDoS attacks and defense mechanisms: A classification. In Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (ISSPRT), Darmstadt, Germany, 17 December 2003; pp. 190–193. [CrossRef]
52. Zargar, S.T.; Joshi, J.; Tipper, D. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Commun. Sur. Tutor.* **2013**, *15*, 2046–2069. [CrossRef]
53. Vishwakarma, R.; Jain, A.K. A survey of DDoS attacking techniques and defence mechanisms in the IoT network. *Telecom. Syst.* **2020**, *73*, 3–25. [CrossRef]

54. Mycek, M.; Secci, S.; Pióro, M.; Rougier, J.-L. Cooperative multi-provider routing optimization and income distribution. In Proceedings of the 7th International Workshop on Design of Reliable Communication Networks (DRCN), Washington, DC, USA, 25–28 October 2009; pp. 281–288. [CrossRef]
55. Jimmy, L.B.; Jamie, R.K.; Geoffrey, E.H. Layer Normalization. *arXiv* **2016**. [CrossRef]
56. Aslansefat, K.; Khanh, N.Q.; Rastogi, O. CICIDS2017: Intrusion Detection Evaluation Dataset. 2019. Available online: https://www.kaggle.com/datasets/cicdataset/cicids2017 (accessed on 16 October 2023).
57. Suárez-Varela, J. NSFNet Topology. 2019. Available online: http://knowledgedefinednetworking.org/data/datasets_v0/nsfnet.tar.gz (accessed on 16 October 2023).
58. Rusek, K. Germany50 Topology. 2020. Available online: http://knowledgedefinednetworking.org/data/datasets_v1/germany50bw.tar.gz (accessed on 16 October 2023).