EVADE: Efficient Moving Target Defense for Autonomous Network Topology Shuffling Using Deep Reinforcement Learning

Qisheng Zhang[1](✉), Jin-Hee Cho[1], Terrence J. Moore[2], Dan Dongseong Kim[3], Hyuk Lim[4], and Frederica Nelson[2]

[1] Department of Computer Science, Virginia Tech, Falls Church, VA, USA
qishengz19@vt.edu
[2] US Army Research Laboratory, Adelphi, MD, USA
[3] School of ITEE, The University of Queensland, Saint Lucia, Australia
[4] Korea Institute of Energy Technology, Naju-si, South Korea

Abstract. We propose an Efficient moVing tArget DEfense (EVADE) that periodically changes a network topology to thwart potential attackers for protecting a given network. To achieve autonomous network topology adaptations under high dynamics, we leverage deep reinforcement learning (DRL) in a moving target defense (MTD) strategy to defeat epidemic attacks. EVADE has two objectives, minimizing security vulnerability caused by the software monoculture and maximizing network connectivity for seamless communications. We design EVADE to autonomously shuffle a network topology by identifying a pair of network adaptation budgets to add and remove edges for generating a robust and connected network topology. To improve the learning convergence speed: 1) We propose a vulnerability ranking algorithm of edges and nodes (VREN) to effectively direct the DRL agent to select adaptations; 2) We develop a Fractal-based Solution Search (FSS) to build an efficient sampling environment for the agent to quickly converge to an optimal solution; and 3) We design density optimization (DO)-based greedy MTD to further refine the solution search space. This hybrid approach achieves faster training allowing running the DRL agent online. Via our extensive experiments under both real and synthetic networks, we demonstrate the outperformance of EVADE over its counterparts.

Keywords: Moving target defense · Network shuffling · Network resilience · Epidemic attacks · Deep reinforcement learning

1 Introduction

The resilience of a network corresponds to the ability to maintain network connectivity when nodes fail or are compromised [44]. Unfortunately, much research in the network science domain stresses this connectivity over the study of security vulnerabilities emanating from epidemic attacks. This vulnerability increases when a software monoculture exists in a system (i.e., the system uses the same software packages or operating system). If an attacker has an exploit that can compromise one node, then it can compromise the entire system. This motivates the design of systems with a software polyculture that cultivates diversity to enhance security [46,50]. Common strategies utilizing diversity to minimize vulnerability include the use of graph coloring to solve software assignment [46] and network rewiring adaptations to limit chains of nodes with the same software in the system [48]. This latter approach suggests how to select links or edges between nodes for rewiring but leaves the question of how many to adapt. We seek to address this issue under the concept of moving target defense (MTD) that dynamically changes the attack surface (e.g., network topology) to increase uncertainty for the attacker [6]. Prior research [19,50] has studied a network shuffling-based MTD; however, frequent changes to a network topology can be a detriment to the seamless service provisions in a system [6]. It is an NP-hard problem to identify an optimal network topology that is robust against attacks while maintaining sufficient connectivity under high dynamics.

We aim to develop a novel MTD technique to ensure long-term system security and performance under epidemic attacks. We leverage reinforcement learning to solve this problem by identifying the optimal settings autonomously without relying on prior knowledge, which may not be available under highly dynamic, contested environments. Specifically, we develop an Efficient moVing tArget DEfense, or `EVADE`, that uses efficient deep reinforcement learning (DRL) algorithms to adaptively shuffle a network topology for minimizing security vulnerability and maximizing network connectivity. Our **key contributions** include:

- We propose algorithms for a DRL agent to efficiently find an optimal budget strategy for adapting a network topology that minimizes security vulnerabilities and maximizes network connectivity. `EVADE` includes: (1) `VREN` (Vulnerability Ranking of Edges and Nodes) to weight the vulnerability of edges and nodes in the network, that is used to select candidates for edge adaptations; and (2) `FSS` (Fractal-based Solution Search) to substantially reduce the steps required to find the optimal edge adaptation budget.
- We introduce a hybrid approach to speed up the MTD process. We develop a greedy algorithm, called Density Optimization (`DO`), to narrow the search space for the DRL agent. The agent leverages the initial estimate from `DO` and quickly converges to the optimal solution(s).
- We consider *epidemic attacks* and *state manipulation attacks*. The epidemic attacks are modeled as dynamically arriving attackers that can compromise adjacent nodes. The state manipulation attacks are modeled as inside attackers that can perturb system states informing DRL. The performance of `EVADE` is evaluated as a network topology shuffling-based MTD triggered multiple rounds over time based on real datasets. We prove our approach shows higher resilience against those attacks than existing counterparts.

– Via extensive experiments, we analyze the performance of `EVADE` incorporating one of two DRL methods, Deep Q-Network (DQN) [30] and Proximal Policy Optimization (PPO) [37]. We prove that DRL-based MTD schemes (i.e., DQN/PPO with `EVADE` or hybrid `EVADE` combining DRL and `DO`) outperform over existing counterparts in the two objectives.

This work mainly concerns epidemic attacks and state manipulation attacks as described above (see Attack Model in Sect. 4). Considering other one-shot attacks, such as leaking highly confidential data out to make a system fail with a one-time security failure, is beyond the scope of this work.

2 Related Work

This section provides the brief overview of the related work in terms of secure network topology and network resilience. In particular, we discuss the related work for secure network topology in terms of moving target defense, diversity-based security, and DRL-based network topology adaptations.

2.1 Secure Network Topology

Network Topology Shuffling-Based Moving Target Defense. The key idea of MTD is to prevent potential attacks by changing attack surfaces, such as system configurations, including IP addresses, port numbers, instruction sets, operating systems, or network topology [6]. Network shuffling-based MTD changes a network topology, which can disturb attack processes leveraging existing attack paths [6]. Online network shuffling-based MTD can defend against worm attacks by solving a software assignment problem [19]. Reconnaissance attacks can be defended by changing virtual network topologies as defensive deception [1]. To find adaptive MTD strategies, [12] proposed a game theoretic multi-agent reinforcement learning (RL) algorithm. `DQ-MOTAG` is a MTD designed for a cyber-physical system (CPS) to redirect user-server connections in the presence of Distributed Denial-of-Service (DDoS) attacks [5]. `DQ-MOTAG` can identify an optimal frequency to trigger MTD with RL. They proved the outperformance of the `DQ-MOTAG` in system performance and availability. [24] proposed a DRL-based traffic inspection system and developed a MTD `DIVERGENCE` based on it. `DIVERGENCE` can protect the IP network traffic flows against targeted DoS attacks by shuffling the IP addresses. However, changing software assignments has been proven less effective than changing network topology [48]. In addition, considering only reconnaissance attacks [1] is limited without considering both security vulnerability and network connectivity. DRL can be used to choose a defense policy on whether to be online or offline only for a small network [12] with 10 nodes. [5] focused on determining an optimal interval to shuffle a network topology. However, how to shuffle a network topology is more critical under resource-constrained settings particularly when defense cost and communication interruptions need to be minimized. [24] only considered targeted DoS attacks, thus the network topology is only partially modified by redirecting network connections. Therefore, they did not consider comprehensive network topology adaptation operations including adding and removing network connections.

Diversity-Based Network Topology Adaptations. In network science, researchers try to generate diversity-based secure network topology against various types of attacks. The diversity concept has been introduced in many hardware or software components in networks [41]. It is vital to generate high quality diversity to achieve enhanced system security and survivability [18]. To this end, reliable diversity deployment approaches typically involve solving software assignment problems by graph coloring [22], defending attacks by sensor worms [46], and solving a multi-coloring problem with an approach based on centrality [21] and improving network tolerance by distributed algorithms [34]. Game theory has been used to solve this problem by identifying an optimal defense strategy based on effectiveness and efficiency criteria [2]. Likewise, [2] solved this problem by using game theory to identify an optimal defense strategy based on the effectiveness and effiicency criteria and found a Nash equilibrium strategy between maximizing defense effectiveness and minimizing defense cost in a game theory adapted to graph coloring approach. However, these approaches are often restricted to static networks or incur high reconfiguration costs. A graph coloring technique from graph theory [22] has been leveraged to solve the software assignment problem. [45,46] applied a software diversity technique to solve a software assignment problem and defend against sensor worms. [34] solved a software assignment problem by developing several coloring algorithms. [20,21] proposed an algorithm of differently ordering colors based on priority measured by centrality metrics. Game-theoretic approaches have also solved a graph coloring problem by minimizing the effect of system vulnerabilities under epidemic attacks [42]. [42] further solved a graph coloring problem by using a game theoretic approach to minimize the effect of system vulnerabilities under epidemic attacks. [2] used a graph-coloring algorithm to optimally identify a software diversity strategy to maximize defense effectiveness and minimize defense cost and found a Nash equilibrium solution. However, the graph coloring approach has been primarily studied in static networks while incurring high cost.

DRL-Based Network Topology Adaptations. Reinforcement learning (RL) has been studied as one of the promising goal-driven algorithms that offers the capability of making autonomous decisions by learning a policy via trials and errors and maximizing the accumulated reward. RL has achieved huge success in solving low-dimensional problems [25,38]. On the other hand, it has been more challenging for RL to solve high-dimensional problems. To relax the complexity of high-dimensional problems, deep learning has been introduced to RL, creating the so-called *deep reinforcement learning* (DRL) [4]. Deep reinforcement learning (DRL) has also been used for network topology adaptations using graph embedding. [9] considered the edge addition based on a node embedding obtained by an structure2vec (S2V) variant and measured network robustness with a customized metric. Stochastic stability and Markov chains have been used in adaptive and robust RL algorithms to thwart attacks [53]. [53] used stochastic stability and Markov chains to propose two adaptive and robust RL algorithms to thwart attacks with an identified optimal defense strategy. Markov Decision with multiple objectives are used to design an RL algorithm for minimizing security vulnerability [43]. [43] leveraged the Markov Decision with multiple objectives to design an RL algorithm to minimize security vulnerability by reducing attack surface. For networks equipped with multimedia services, a mechanism was developed to redirect routes under Denial-of-Service (DoS) using DQN to learn an optimal mutated route for each source-destination pair [51]. [51] studied networks equipped with multimedia services and developed a mechanism to redirect routes under Denial-of-Service (DoS) using DQN to learn an optimal mutated routes available for each source and destination pair. Recent work studying network adaptations [47,49] also investigated how to minimize security vulnerability while maximizing network connectivity. Recently, DRL has been used to generate an optimal network topology aiming to minimize security vulnerability while maintaining acceptable network connectivity [47,49]. However, they used DQN to identify an optimal network topology by considering one time topology adaptation under one time attack only. Therefore, they did not offer the capability to investigate the performance of a network shuffling-based MTD when different types of DRL algorithms are leveraged along with greedy algorithms to enhance the speed of learning convergence under multiple attackers arriving in a network dynamically.

As evidenced in the prior literature review, the key challenge in using DRL is the slow learning convergence. In particular, this can be a serious hurdle under a network with high hostility, dynamics, and resource constraints. `EVADE` provides efficient solution search methods that can fill this gap. Based on our literature review above, we found the key challenge in using DRL is slow learning convergence. In particular, this can be a serious hurdle under a network with high hostility, dynamics, and resource constraints. Via the proposed `EVADE` that provides efficient solution search methods, our work can fill this gap.

2.2 Network Resilience in Network Science

In network science, the concept of network resilience (or robustness) is partly originated from the theory of percolation. As a topic of statistical physics, percolation theory aims to describe the flow pattern of liquid via a medium with mathematical formulas [16]. Computer scientists have found the interconnection between percolation theory and software diversity. They have already used this concept to defend against epidemic attacks by solving a software assignment problem [45,46]. As the propagation of computer viruses or malware highly relies on network topology, percolation theory plays a significant role in developing software diversity methods considering the network topology [32]. These efforts in network resilience studied by computer scientists are independent from the statistical physicists [7,31,39]. Although network resilience has been substantially studied, the concept of network resilience studied in the network science domain is typically limited to maximizing the size of the giant component in a network without considering security vulnerability, such as the presence of compromised nodes or diverse attacks performed by those undetected, compromised nodes in the network. In our work, we consider both security vulnerability and network connectivity by measuring the fraction of compromised nodes and the size of the giant component in a network topology generated by DRL-based MTD, aiming to identify an optimal budget size in adapting edges between nodes based on their vulnerabilities and provide efficient proactive defense using network topology shuffling-based MTD.

Limitations of the Existing Approaches. We summarize the existing work's main limitations as (1) Shuffling a network topology under resource-constrained settings is not well studied; (2) Graph coloring-based diversity deployment approaches have high cost and limited scope in static networks; and (3) The current DRL-based network topology adaptation approaches have limited scalability and efficiency to provide multi-objective defense strategies.

To fill these gaps, our proposed `EVADE` can address all these challenges by performing moving target defense with an efficient DRL-based shuffling approach in large-scale network topology adaptations under a resource-constrained dynamic network environment.

3 Problem Statement

This work seeks to improve the robustness of a network topology under epidemic attacks by selecting edge adaptations within budget constraints that minimize the security vulnerabilities in the presence of compromised nodes (see Attack Model in Sect. 4) while maximizing the connectivity of the network. We leverage two state-of-the-art DRL algorithms (i.e., DQN and PPO) and design the problem so that the agent learns the budgets for adding b_A and removing b_R edges under the constraint $0 \leq b_A + b_R \leq B$. Removing a node means removing all edges associated with the node. Hence, we only consider the edge operations with a certain number of nodes to perform budget-constrained network adaptations.

Given a current network topology G, the DRL agent finds the number of edge adaptations (i.e., (b_A, b_R)) to generate an adapted network G'. Since identifying an optimal topology that is robust to epidemic attacks by considering all possible edges to remove or add is an NP-hard problem, we propose a network shuffling-based MTD, called `EVADE`, consisting of two heuristic approaches to reduce the solution space and identify the edge adaptation budgets based on a vulnerability estimation: (1) *Vulnerability Ranking of Edges and Nodes* (`VREN`) generates vulnerability scores of existing edges and nodes based on the frequency of use or appearance on simulated attack paths. Using this ranking, for a budget pair (b_A, b_R), the system finds an adapted network G' by selecting the b_R most vulnerable edges to remove from and the b_A least vulnerable edges to add to the current network G. We discuss the details of `VREN` in Sect. 5.1; and (2) *Fractal-based Solution Search* (`FSS`) reduces the search space complexity for identifying an optimal budget pair, (b_A, b_R), when the space increases due to network size or budget size. By significantly reducing this complexity, `FSS` introduces a faster convergence for the DRL agent to find the solution, described in Sect. 5.2.

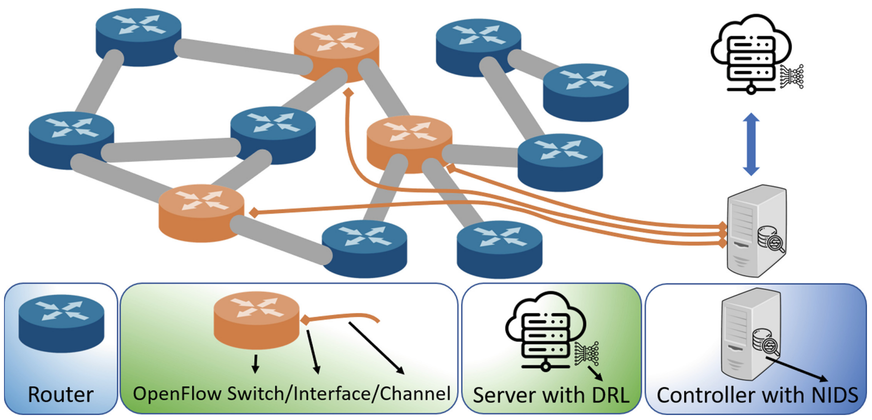


Fig. 1. An overview of the network model.

To state this more formally, the DRL agent finds a solution to the following optimization problem via `EVADE`:

$$\arg\max_{b_A, b_R} f(G') - f(G), \quad s.t. \quad 0 \leq b_A + b_R \leq B, \tag{1}$$

where B is the constraint that bounds the total budget for edge adaptations and $f(G)$ captures the two performance metrics, i.e., $f(G) = \mathcal{S}_{\mathcal{G}}(G) - \mathcal{F}_{\mathcal{C}}(G)$, as the difference between the network connectivity given by the expected size of the giant component composed of non-compromised, active nodes in a network, $\mathcal{S}_{\mathcal{G}}$, and the expected fraction of compromised nodes, $\mathcal{F}_{\mathcal{C}}$. These two metrics are described in Sect. 6. Since the system objectives of maximizing connectivity while minimizing vulnerability can be conflicting, the optimal solutions identified by the DRL agent will not necessarily achieve the best performance in these metrics. To further improve the performance of the MTD in meeting these conflicting goals, we also introduce a hybrid MTD approach of combining `EVADE` with a greedy algorithm that optimizes performance based on network density, as described in Sect. 5.4.

4 System Model

4.1 Network Model

We consider an IP network with a centralized coordinator to run our proposed network topology shuffling-based MTD, `EVADE`. This type of network is very common, such as a software-defined network (SDN) with a single SDN controller [26], an Internet-of-Things (IoT) network that has an edge server or central cloud server, or a special type of an ad-hoc mobile network with a centralized controller [23]. In the present work, we consider a single DRL agent that is capable of making autonomous decisions. The DRL agent will identify a network topology that is robust against epidemic attacks and ensures network connectivity for providing seamless network services. For a large-scale network, we can divide it into multiple network partitions and deploy one DRL agent for each partition. Each agent could update and share its partition with other agents. Therefore, our work can be extensible to a large-scale network with multiple DRL agents for cooperative, autonomous decision-making.

To realize this, we adopt a hybrid SDN structure as the control plane of our IP network. As in Fig. 1, the single SDN controller can gather information from OpenFlow switches and form a global view on the current network state, i.e., the network topology and software packages used by nodes in the network. This information will allow the DRL agent placed on the server to make autonomous decisions in adding or removing edges. The adapted network topology based on such decisions can mitigate security vulnerability and maintain network connectivity. Once the topology adaptation decision is made, the centralized controller notifies the hosts to execute the action through their IP addresses.

We represent a network as an undirected graph, $G = (V, E)$, where V refers to a set of nodes and E is a set of edges between two nodes. The network is dynamic and changes when nodes fail due to defects or attacks or when a network topology is changed by the network shuffling process to meet the two system objectives. We denote a network topology by an adjacency matrix $\mathbf{A}$, which has an entry 1 (i.e., $a_{ij} = 1$) for an edge existing between two nodes i and j and 0 (i.e., $a_{ij} = 0$) otherwise. We adopt a time-based MTD performing a network shuffling operation per time cycle in minutes. The DRL agent keeps making network topology shuffling decisions to prevent potential epidemic attacks while maintaining acceptable network connectivity (e.g., maximum possible connectivity in terms of the size of the giant component described in Sect. 6). We describe how the proposed `EVADE` works in Sect. 5.

As a background defense mechanism, we assume that a network-based intrusion detection system (NIDS) exists on the SDN controller to monitor all inter-host traffic and detect outside attackers and inside attackers, nodes compromised by the attackers. The NIDS will capture and analyze network traffic from switches in a given SDN. We consider the detection accuracy of the NIDS with probability ζ, which has the similar effect of removing nodes as in the Susceptible-Infected-Removed (SIR) epidemic model [32]. As the response to the detected compromised nodes by the NIDS, a secret key used in the network will be changed and this allows their isolation from the network for secure group communications. We characterize the NIDS by false positives and negatives with the probabilities P_{fp} and P_{fn} that set to $1-\zeta$. Since the DRL agent relies on the information from the NIDS for the status of nodes to measure system security (see the metrics defined in Sect. 6), it may not be able to choose the best solution under the ground truth knowledge, which reflects a realistic scenario in practice. Note that our work does not develop a NIDS, which is beyond the scope of our work. We focus on developing an efficient network topology shuffling-based MTD with lightweight features for faster learning convergence to find an optimized pair of network adaptation budgets. We display our network model in Fig. 1.

4.2 Node Model

Each node i has the following attributes: (1) Whether node i is active or not (i.e., $na_i = 0$ or $na_i = 1$); (2) Whether node i is compromised or not (i.e., $nc_i = 0$ or $nc_i = 1$); (3) What software node i has, denoted by $s_i \in \{1, 2, \ldots, N_s\}$, where N_s indicates how many software packages are available for nodes in the network; and (4) What level of vulnerability node i has, denoted by sv_i. To properly model sv_i, we employ the Common Vulnerability Scoring System (CVSS) [8]. In the CVSS, a node's average vulnerabilities for different types of attacks are given in the range of $[0, 10]$. We normalize it to $[0, 1]$ and consider sv_i as $\frac{s_i}{N_s+1}$. Our case study will select these vulnerability values randomly based on uniform distribution as our work does not focus on particular software vulnerabilities, which is beyond the scope of this work.

In summary, based on the above four attributes, node i is characterized by:

$$\mathbf{node}(i) = [na_i, nc_i, s_i, sv_i], \tag{2}$$

where $\mathbf{node} \in \mathbb{R}^{N\times 4}$ is the collection of attributes of all N nodes.

4.3 Attack Model

We assume that attackers could only access the IP network through public addresses. In addition, the attackers do not have direct access to the centralized SDN controller and cannot disable the MTD. We consider the following attacks in this work:

- *Epidemic Attacks*: We randomly select initial epidemic attackers following a Poisson distribution with λ, which is the expected number of new attackers arriving in one attack round. Each initial attacker is assumed to compromise one host from its public IP address. The infection rate, β_{ji}, is used to model the compromising behavior of epidemic attacks from attacker j to node i [17] by $\beta_{ji} = sv_i$ if $\sigma_j(s_i) = 0$; 1 otherwise. The β_{ji} returns sv_i when attacker j has not compromised nodes with the same s_i (i.e., $\sigma_j(s_i) = 0$) and returns 1 otherwise. We consider the epidemic attack behavior based on the Susceptible-Infected-Removed (SIR) model [32] where the removed status refers to the compromised nodes being detected and evicted from the system. The attacker can directly spread viruses or malwares and compromise its adjacent nodes without access rights to their settings/files, such as botnets [13]. The detailed attack procedures are described in Algorithm 1.
- *State Manipulation Attacks* (SMAs): We assume there are inside attackers that can send false information to hinder the DRL agent's decisions to find an optimal budget pair for network topology adaptations. In general, the DRL agent chooses an action and updates its policy based on the reward received from the current system state. Therefore, receiving correct system states is critical for the DRL agent to learn an ideal policy. We consider SMAs to study the resilience of the proposed `EVADE`. We denote the probability the attackers manipulate a system state by P_s. Note that SMAs are independent from epidemic attacks and P_s determines the frequency of performing SMAs.

Algorithm 1. Perform Epidemic Attacks

```
1: Input:
2: $N \leftarrow$ number of nodes in a network
3: $T \leftarrow$ number of attack times
4: $\lambda \leftarrow$ expected number of new attackers arriving in one attack round
5: $\mathbf{A} \leftarrow$ an adjacency matrix
6: node ← all nodes' attributes defined by Eq. (2)
7: $\sigma \leftarrow$ all nodes' vector of exploitable software packages
8: $\gamma \leftarrow$ an intrusion detection probability
9: spread ← all nodes' history attack attempts, initialized as a zero vector
10: procedure performEpidemicAttacks($N$, $T$, $\lambda$, $\mathbf{A}$, node, $\sigma$, $\gamma$, spread)
11:     for $t := 1$ to $T$ do
12:         choose new attackers based on Poisson distribution $Pois(\lambda)$
13:         for $i := 1$ to $N$ do
14:             if $na_i > 0 \wedge nc_i > 0$ then                ▷ Check if $i$ is an active attacker.
15:                 $r_1 \leftarrow$ a random number in $[0, 1]$
16:                 if $r_1 > \gamma \wedge \mathbf{spread}(i) < 2$ then
17:                     $\mathbf{spread}(i) = \mathbf{spread}(i) + 1$
18:                     for $j := 1$ to $N$ do
19:                         if $a_{ij} > 0 \ \wedge na_j > 0 \wedge nc_j == 0$ then        ▷ Check if $j$ is susceptible.
20:                             if $\sigma_i(s_j) = 1$ then              ▷ $i$ knows $s_j$'s vulnerability.
21:                                 $nc_j = 1$                          ▷ $j$ is compromised by $i$.
22:                             else
23:                                 $r_2 \leftarrow$ a random number in $[0, 1]$
24:                                 if $r_2 < sv_j$ then
25:                                     $nc_j = 1$                      ▷ $j$ is compromised by $i$.
26:                                     $\sigma_i(s_j) = 1$              ▷ $i$ learns $s_j$'s vulnerability.
27:                                 end if
28:                             end if
29:                         end if
30:                     end for
31:                 else
32:                     $na_i = 0$       ▷ $i$ is detected or deactivated for infecting behavior.
33:                     $a_{ij} = 0, a_{ji} = 0$           ▷ IDS disconnects all edges adjacent to $i$.
34:                 end if
35:             end if
36:         end for
37:     end for
38: end procedure
```

We assume that `EVADE` is equipped with a detector to detect the SMA. The detection probability, D_r, is considered in the deployed detector. As the system is equipped with the SMA detector, the DRL agent can use the detected system status to respond to the compromised system state. The DRL agent will use a previously stored system state if the received state is detected as false. Otherwise, using the undetected, false system state may result in adverse decision performance by the DRL agent.

5 Description of EVADE

5.1 Ranking Vulnerabilities Using VREN

We propose an algorithm called VREN, representing *Vulnerability Ranking for Edges and Nodes*. VREN is used when network topology adaptations are performed by adding or removing edges where the number of edge additions and removals are given as a budget pair (b_A, b_R), respectively. In a graph $G = (V, E)$ representing a given network, we denote nodes' and edges' vulnerability levels by $\mathbf{V}_V$ and $\mathbf{V}_E$. The vulnerability of edges and nodes are determined from n_a attack simulations on G following the epidemic attacks described in Attack Model in Sect. 4 where an initial set of nodes are compromised randomly. The vulnerability of the edge linking nodes i and j, denoted $\mathbf{V}_E^{ij}$, is estimated by the frequency the edge can be used to compromise other nodes. The vulnerability of node i, denoted $\mathbf{V}_V^i$, is estimated by the frequency the node resides on an attack path. This process will return $\mathbf{V}_V$ and $\mathbf{V}_E$ as vectors of non-negative integers. To complete VREN, the edges and nodes are ranked based on the returned vulnerability levels. The ranks of edges and nodes are represented by $\mathbf{R}_E$ and $\mathbf{R}_V$, respectively, where $\mathbf{R}_E$ is sorted from the highest to the lowest and $\mathbf{R}_V$ is sorted from the lowest to the highest. A sufficiently large n_a (i.e., 500 in our study on graphs described in Sect. 6) demonstrates the convergence of $\mathbf{R}_V$ and $\mathbf{R}_E$.

The proposed VREN can be used to estimate the network vulnerability. Specifically, the vulnerability levels can be used to calculate the mean fraction of compromised nodes, $\mathcal{F}_C$, in a given network as:

$$\begin{aligned}\mathbb{E}(\mathcal{F}_C) &= \sum_{i=1}^{N} P_i = \sum_{I \subseteq V} P_I \sum_{i=1}^{N} P_i^I = \sum_{i=1}^{N} \sum_{I \subseteq V} P_I P_i^I \\ &= \sum_{i=1}^{N} \lim_{n_a \to \infty} (V_V)_i + \|I\| = \|I\| + \lim_{n_a \to \infty} \sum_{i=1}^{N} V_V^i \\ &= \|I\| + \lim_{n_a \to \infty} \sum_{1 \le i < j \le N} V_E^{ij}, \end{aligned} \tag{3}$$

where N refers to the total number of nodes, I is an initial set of attackers, P_i represents the probability node i is compromised, P_I indicates the probability that a set of given attackers, I, is selected to perform attacks, P_i^I refers to the conditional probability that node i is compromised if I is selected as the initial set of attackers, $\mathbf{V}_V^i$ refers to the vulnerability of node i, and $\mathbf{V}_E^{ij}$ is the vulnerability of an edge between nodes i and j.

Algorithm 2 describes how VREN works in detail. Given $G = (V, E)$ and (b_A, b_R), our proposed network topology adaptation produces a new topology $G' = \text{VREN}(G, b_A, b_R)$ following the procedures below:

Algorithm 2. Vulnerability Ranking of Edges and Nodes (VREN)

```
1: $G \leftarrow$ An original network
2: $N \leftarrow$ Total number of nodes in a network
3: $b_A \leftarrow$ Addition budget
4: $b_R \leftarrow$ Removal budget
5: $n_a \leftarrow$ Total number of attack simulations to identify edge and node vulnerabilities
   based on their appearance on attack paths
6: $k \leftarrow$ Upper hop bound for edge addition
7: $G' = \mathbf{VREN}(G, b_A, b_R)$
8: $V_V \leftarrow$ Node vulnerability levels, initialized as a zero vector
9: $V_E \leftarrow$ Edge vulnerability levels, initialized as a zero vector
10: **1.A: Edges Removals**
11: **for** $t \leftarrow 1$ to $n_a$ **do**
12:     $V_V^*, V_E^* =$ EpidemicAttacks($G$)        ▷ Single attack simulation
13:     $V_E = V_E + V_E^*$
14: **end for**
15: $R_E \leftarrow$ rank $E$ in descending order with key $V_E$
16: $G^* \leftarrow$ remove top $b_R$ edges from $G$ according to $R_E$
17: **for** $t \leftarrow 1$ to $n_a$ **do**
18:     $V_V', V_E' =$ EpidemicAttacks($G^*$)        ▷ Single attack simulation
19:     $V_V = V_V + V_V'$
20: **end for**
21: **1.B: Edges Additions**
22: $R_V \leftarrow$ rank $V$ in ascending order with key $V_V$
23: $gc \leftarrow$ the giant component of $G^*$
24: $R_V^{gc}, R_V^{ngc} \leftarrow$ order $gc$ and the complement $ngc$ based on $R_V$
25: $G' = G^*$        ▷ All the following adaptations are on $G'$
26: **for** $i$ in $R_V^{ngc}$ **do**
27:     **for** $j$ in $R_V^{gc}$ **do**
28:         $G' \leftarrow$ Connect nodes $i, j$ and break if $dist(i, j) \in [2, d_a]$ ▷ $dist$ considers the
   length of a shortest path between nodes $i$ and $j$ and $d_a$ is the maximum number
   of hops allowed
29:     **end for**
30: **end for**
31: $b_A^{temp} \leftarrow \|R_V^{ngc}\|$
32: **for** $h \leftarrow 2$ to $N$ **do**
33:     $R_V^h \leftarrow$ top $h$ nodes in $R_V$
34:     **for** $i$ in $R_V^h$ **do**
35:         $R_V^{i-1} \leftarrow$ top $i-1$ nodes in $R_V$
36:         **for** $j$ in $R_V^{i-1}$ **do**
37:             break if $b_A^{temp} \geq b_A$
38:             **if** $dist(i, j) \leq d_a$ **then**
39:                 $G' \leftarrow$ Connect nodes $i, j$ if $dist(i, j) \in [2, d_a]$
40:                 $b_A^{temp} = b_A^{temp} + 1$
41:             **end if**
42:         **end for**
43:     **end for**
44: **end for**
45: **return** $G'$
```

1. The edge vulnerabilities, $\mathbf{V}_E$, are estimated based on G, and the top b_R edges are removed based on the edge vulnerability ranks in $\mathbf{R}_E$. The resulting graph, G^*, represents an intermediate adapted graph, as described in the line 16 of Algorithm 2.
2. A minimum number of edges are added to restore the connectivity between a pair of nodes with the maximum distance d_a in G^* where the resulting adapted graph is G'. This is described in Algorithm 2 from line 17 to line 30.
3. $\mathbf{R}_V$ of G' is estimated and maximum b_A edges are added to G' based on the rank in $\mathbf{R}_V$. Since a pair of nodes can be connected also based on the distance between them, they can be connected only when the distance between the pair is no greater than distance d_a. This is shown in Algorithm 2 from line 31 to line 45.

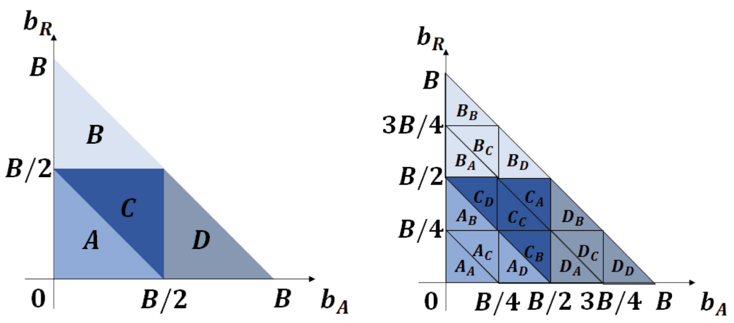
To introduce efficiency while maintaining accuracy of vulnerability estimation, we can use ego networks to estimate $\mathbf{R}_E$ and $\mathbf{R}_V$. We can use hop distance h to describe the ego network. Specifically, an h-hop ego network for node k, denoted by EN_k^h, is defined to be the network including all nodes within distance h from k. Due to the nature of epidemic attacks, nodes are more likely to be compromised when they are closer to the attackers. We consider each node's ego network to simulate attacks and calculate $\mathbf{R}_E$ and $\mathbf{R}_V$. We choose a hop distance h^* and simulate attacks in N ego networks, $EN_k^{h^*}$, for node $1 \leq k \leq N$. Given a certain number of attack simulation runs, using the ego network allows us to compute the number of times each node i $(\neq k)$ becomes compromised and generate the following list:

$$[NC_i^0, NC_i^1, \cdots, NC_i^j, \cdots, NC_i^{h^*}], \tag{4}$$

where NC_i^j is the number of times node i becomes compromised in $EN_k^{h^*}$ for all node k such that node i is within j hop(s) distance away from node k. h^* determines the size of node k's ego network and j decides the distance between nodes i and k. Hence, NC_i^0 refers to the case of $i = k$ and indicates the number of times node i being compromised in its ego network, $EN_i^{h^*}$.

After then, we can rank nodes and edges based on NC_i^j where h^* refers to the hop value used to determine every node's ego network. When h^* is no smaller than the maximum hop-distance between two nodes in the original network, all h^*+1 rankings will become almost the same as all nodes' ego networks would be the same. Specifically, given h^* and a certain number of attack simulations, n_a, larger NC_i^j is obtained with larger j, and vice-versa. In general, given different nodes i_1 and i_2, larger j can result in more distinct differences between $NC_{i_1}^j$ and $NC_{i_2}^j$. Suppose we rank nodes i_1 and i_2 when $|NC_{i_1}^j - NC_{i_2}^j| > C > 0$, where constant C is a given ranking resolution. Larger j can allow ranking with larger C, and vice-versa. That is, using larger j allows more easily identifying C as NC_i^j becomes larger and more differences with other NC_i^j's for other i's. On the other hand, NC_i^j with smaller j can represent higher vulnerability than NC_i^j with larger j. In this work, we use $h^* = 2$ and rank each node i based on NC_i^2 for our experiments in Sect. 7.

5.2 Solution Search with FSS



(a) Original divisions (b) Divided subdivisions

Fig. 2. Self-similar fractal-based division for identifying a reduced solution search space of the budgets for edge additions and removals (i.e., (b_A, b_R)).

We propose an algorithm called *Fractal-based Solution Search (*FSS*)* to aid the DRL agent for efficiently identifying an optimized solution in the solution space. In Eq. (1), we defined our objective function based on the differences of the rewards from the current network G and the adapted network G'. The optimization problem in Eq. (1) can be rewritten by:

$$\arg\max_{b_A, b_R} f(G') - f(G) \quad \text{where} \quad G' = \text{VREN}(G, b_A, b_R) \tag{5}$$
$$s.t. \quad 0 \leq b_A + b_R \leq B,$$

where $f : G \mapsto \mathcal{S}_G(G) - \mathcal{F}_C(G)$, and the budgets for edge removal and addition are denoted by b_A and b_R, respectively. B refers to the maximum number of budgets allowed (i.e., the budget upper bound).

The optimization problem in Eq. (5) is defined based on the triangle area $\{(b_A, b_R) | 0 \leq b_A + b_R \leq B\}$. Specifically, Fig. 2 describes a triangle divided into smaller triangles. Each smaller triangle, denoted by A, B, C and D, can be further subdivided similarly. This self-similar fractal can efficiently lead the DRL agent to identify an optimized solution since each action would result in a smaller triangle as the new search space. It also enables the evaluation of the objective function in Eq. (5) by using the centroid of each division and search the nearest integer pair (b_A, b_R) from the centroid. In Fig. 2 (a), under each subdivision, division C (i.e., subscript) has an invariant centroid. Given the triangle subdivided, if the longest side length of the triangle is less than 1, the nearest integer points of the centers of its subdivisions will be its vertices. This means it is sufficient to iterate the subdivision until the longest side length of the triangle is less than 1 due to no more new budget pairs to be generated.

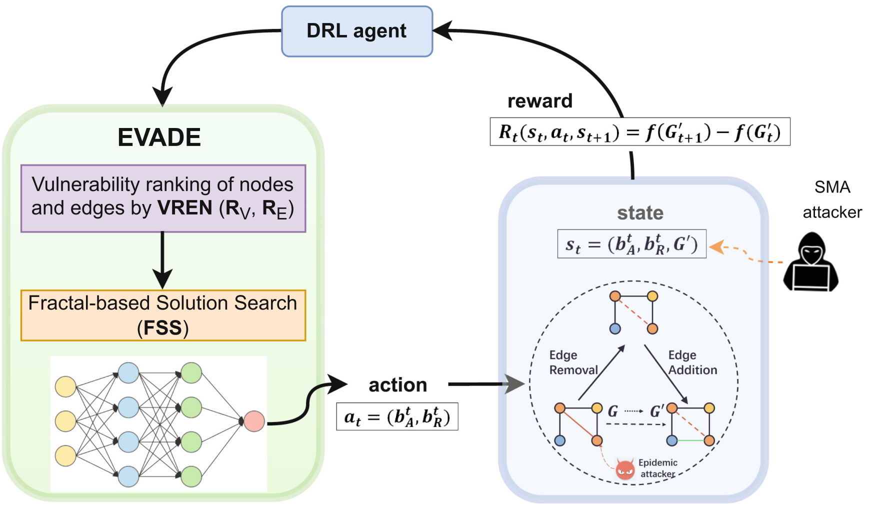

DRL agent
EVADE
Vulnerability ranking of nodes and edges by VREN (R_V, R_E)
Fractal-based Solution Search (FSS)
action
$a_t = (b_A^t, b_R^t)$
reward
$R_t(s_t, a_t, s_{t+1}) = f(G'_{t+1}) - f(G'_t)$
state
$s_t = (b_A^t, b_R^t, G')$
SMA attacker
Edge Removal
Edge Addition
G → G'
Epidemic attacker

Fig. 3. DRL-Based optimal budget identification in EVADE.

5.3 DRL-based Optimal Budget Identification

This section discusses how an DRL agent identifies a budget pair (b_A^*, b_R^*) given B, an upper bound budget.

Here we first provide the brief steps about how a DRL agent learns a policy function and identifies an optimal action by following the key idea of reinforcement learning (RL). RL uses a Markov Decision Process (MDP), that has the following key components: (1) a set of states $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_t, \ldots \mathbf{s}_T\}$ with a distribution of starting states p$(\mathbf{s}_1)$ where $\mathbf{s}_t$ represents a set of states s_t at time t; (2) a set of actions, $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_t, \ldots \mathbf{a}_T\}$ where $\mathbf{a}_t$ is a set of actions a_t available at time t; (3) a transition probability, $\mathcal{T}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$, from state $\mathbf{s}_t$ to state $\mathbf{s}_{t+1}$ via $\mathbf{a}_t$; (4) an immediate reward function, $\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$; (5) a discount factor $\gamma \in [0, 1]$ where lower γ counts immediate rewards more ; and (6) a policy π mapping from states to a probability distribution of actions, $\pi : \mathcal{S} \rightarrow (\mathcal{A} = a \mid \mathcal{S})$. The key components of the policy trajectory include the sequence of states, actions, and rewards in an episode. After the whole trajectory of the policy, the accumulated reward is denoted by $R = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$ where T is the length of the episode [40]. This means that a DRL agent has an objective of finding an optimal policy π^* which can produce the maximum expected reward from all states where $\pi^* = \arg\max_{\pi} \mathbb{E}[R \mid \pi]$ [3].

We use the following key components of the MDP when applying DRL to solve the objective function in Eq. (5). We describe how our proposed FSS is used in the process of MDP in utilizing DRL as follows:

- **States**: When the DRL agent uses FSS and the subdivision is made in each step, $\lceil \log_2 B \rceil$ steps will be taken in total, where B refers to the budget constraint in Eq. (5). A state s_t at step t is defined by $s_t = (b_A^t, b_R^t, G'_t)$, where b_A^t and b_R^t are the numbers of edge additions and removals found by evaluating each division at time t, respectively, and G'_t refers to an adapted network at time t.

- **Actions**: When FSS is used, the DRL agent takes action $\mathbf{a}_t$ at step t based on the division ID to determine a next subdivision. The action $\mathbf{a}_t$ is represented by $\mathbf{a}_t = \{A, B, C, D\}$, where $1 \leq t \leq \lceil \log_2 B \rceil$.
- **Rewards**: The reward function at step t, denoted by $\mathcal{R}(s_t, a_t, s_{t+1})$, is formulated based on the improvement made at $(t+1)$ compared to one at t in the evaluation function $f(\cdot)$, which is formulated by $\mathcal{R}(s_t, a_t, s_{t+1}) = f(G'_{t+1}) - f(G'_t)$, where $f : G \mapsto \mathcal{S}_G(G) - \mathcal{F}_C(G)$ refers to the objective function in Eq. (1).

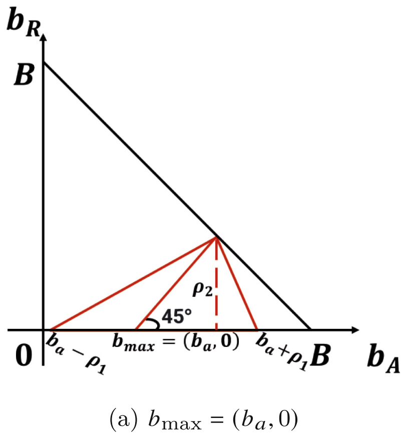
Figure 3 summarizes the overall process of DRL-based MTD considered in our EVADE. In each iteration of RL, the DRL agent needs to go through the whole EVADE process. Further, the DRL agent recalculates the vulnerability rankings by VREN and uses FSS to determine a next action.

5.4 Greedy MTD Using Density Optimization (DO)

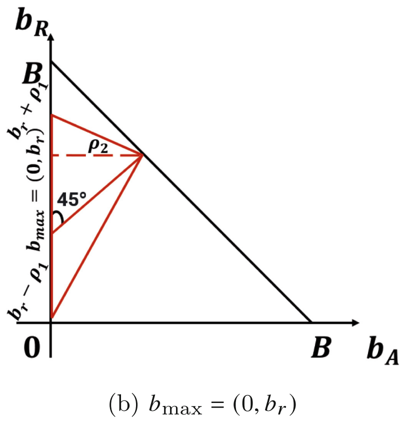
Although we introduce a lightweight solution search algorithm, FSS, we found that DRL still requires substantially long training times. Since DRL is mainly useful when the network environment is significantly varied, we introduce a hybrid MTD approach that can minimally trigger DRL-based MTD while a more lightweight greedy algorithm is developed to trigger MTD more frequently. We discuss how this time-based MTD is triggered in Sect. 5.5. DO aims to optimize the network density with respect to the given function $f(\cdot)$ by only adding or removing edges from the current network. First, we enumerate all possible adapted network densities, given a budget constraint B of edge additions and removals. Then we calculate the minimum budget needed to achieve any possible network density under the constraint B. Lastly, we sample the enumerated budget pairs and check them. After then, we choose the best budget b_{max} with respect to a given evaluation function $f(\cdot)$ defined in Eq. (1).

5.5 Hybrid MTD with EVADE and DO

We consider a time-based MTD [6] which triggers an MTD operation at fixed intervals. As shown in Fig. 5, we first use MTD to take proactive defense by adapting a network topology. Then epidemic attacks are applied on the adapted network. In the attack phase, new attackers will arrive in and the NIDS will detect and isolate them with probability ζ (see Sect. 4). These events, including MTD, attack, and detection by the NIDS, occur within each round in the simulation. We will repeat this procedure multiple times until a given session ends. We consider a hybrid MTD (see Fig. 6) that uses both our proposed greedy MTD algorithm in Sect. 5.4 and our proposed DRL-based MTD in Sect. 5.3. We initiate the greedy MTD and DRL-based MTD periodically with time intervals



(a) $b_{\max} = (b_a, 0)$



(b) $b_{\max} = (0, b_r)$

Fig. 4. The procedure of generating an expanded triangle based on the proposed greedy MTD algorithm: This algorithm can reduce and refine the solution search space to identify a desirable (b_A^*, b_R^*) and is detailed in Sect. 5.5.

ADC_g and ADC_{DRL}, respectively. In the hybrid MTD, we can refine the solution search area by `FSS` with `DO` in Sect. 5.4. To realize this, we start with the point b_{max} identified by `DO` and obtain the corresponding expanded triangle as described in Fig. 4. Here ρ_1 and ρ_2 are the customized lengths related to the area of the expanded triangle. Notice that the angle is 45°C so that the DRL agents can search for budgets corresponding to similar network densities in the refined triangle area. To develop an efficient and effective online MTD, we propose the following ways to reduce the offline evaluation times for the DRL-based MTD: (1) We build a reward tree to store the previous evaluation results; (2) We adjust the DRL agent's reward function to $\mathcal{R}(s_t, a_t, s_{t+1}) = f(G'_{t+1})$ if $t = \lceil \log_2 B \rceil$; 0 otherwise; and (3) We initialize multiple `FSS` environment instances proposed in Sect. 5.2 and evaluate them simultaneously with the same DRL policy function until the policy updates. The MTD agent makes decisions and takes actions based on the information provided by the NIDS about which nodes are compromised in the network. Since the NIDS is characterized by false positives and false negatives, the MTD agent may make decisions based on the imperfect information from the NIDS as the ground truth attack states are not available in real environments.

5.6 Practical Applicability of `EVADE` and `DO`

We discuss several practical, real-world examples of our proposed network topology adaptation strategies. In an SDN, since its vital merit is flexible manageability that can separate the data plane from the control plane, an SDN controller has been commonly used to reconfigure a logical network topology and its flow table. Thus, packets can be forwarded based on routing instructions given by the SDN controller at node levels [19]. Generating virtual network topologies in optical networks, called “virtual topology design,” is well-known to optimize service provision [14,52]. In wireless sensor networks, network topology reconfiguration has been frequently considered for accurate estimates of sensed data by sensors where a gateway provides each node its next node to which a packet is forwarded [11,27]. Moreover, by opening, sectionalizing, and closing tie switches of the network, power distribution systems perform efficient and effective network reconfiguration to minimize their power loss [10,36]. There is a potential for service degradation when the moving target defense (MTD) is actively being applied. Thus, there is a critical tradeoff because more frequent MTD operations to enhance system security can introduce service and performance degradation due to the interruptions introduced by the system reconfiguration. We currently envision a least active implementation to limit any adverse effect on network service during the execution of the network shuffling adaptations as an MTD mechanism.

An example of day-to-day operations can include network configurations, status updates, and maintenance operations upon attacks or outages when the proposed `EVADE` and `DO` are applied in a given network. These operations can be performed based on the following procedures: (1) *Network Configurations*: A network needs to be configured online with the key design parameters that the `EVADE` and `DO` require. Furthermore, to proactively defend against existing attacks, the attack model should be known in advance. For instance, we need to configure the values of key parameters (see Table 1) affected by the system constraints of the current state of the network and the attack model. (2) *Network Status Updates*: As network and environmental conditions may vary due to network topology changes (e.g., node mobility or failure) or attacks, the network vulnerability needs to be reevaluated by `VREN` periodically. Note that `EVADE` and `DO` can be executed offline once the network status is updated. The optimally identified network configuration can be deployed online afterward. (3) *Maintenance*: The network topology adaptation performance can be recorded online every time the MTD operation is triggered. We assume that all the topology and corresponding performance information is backed up and can provide redundancy to maintain reliability and resilience. Under some situations caused by a power outage, operational failures, or successful internal and external attacks, the offline backup information of the network configuration can be used.

6 Experiment Setup

Network Datasets. We use 4 different undirected networks: (1) a sparse network from an observation of the Internet at the autonomous systems level with 1,476 nodes and 2,907 edges [28]; (2) a medium dense network with 1,000 nodes and 6,123 edges derived from a backbone topology of the Internet service provider Level 3 [35]; (3) a dense network with 963 nodes and 11,310 edges derived from a traceroute-based Internet mapping provided by CAIDA Macroscopic Internet Topology Data Kit [29]; and (4) an Erdös-Rényi (ER) random network with 200 nodes and 1,021 edges [32]. We use the original network topology for the

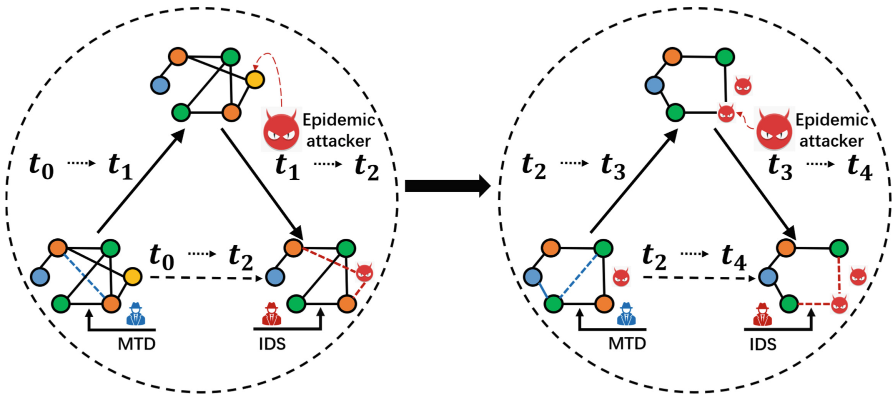

Epidemic attacker
$t_0 \cdots\rightarrow t_1$
$t_1 \cdots\rightarrow t_2$
$t_0 \cdots\rightarrow t_2$
MTD
IDS
Epidemic attacker
$t_2 \cdots\rightarrow t_3$
$t_3 \cdots\rightarrow t_4$
$t_2 \cdots\rightarrow t_4$
MTD
IDS

Fig. 5. The dynamics of attacks and defenses with respect to time in terms of attack arrivals, their detection by the NIDS, and network shuffling-based MTD, as described in Sect. 5.5.

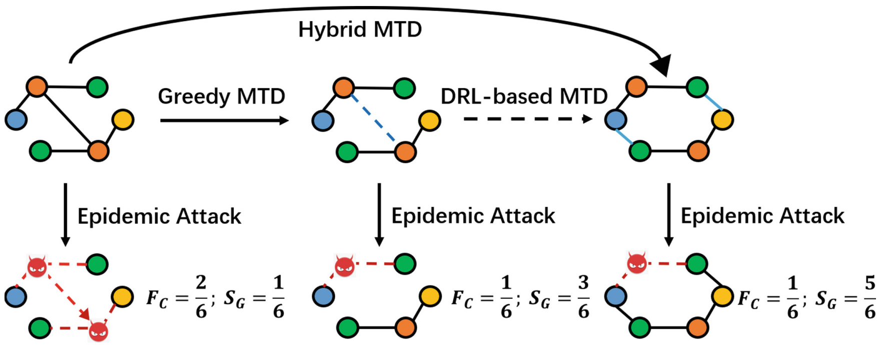

Hybrid MTD
Greedy MTD
DRL-based MTD
Epidemic Attack
Epidemic Attack
Epidemic Attack
$F_C = \frac{2}{6}; S_G = \frac{1}{6}$
$F_C = \frac{1}{6}; S_G = \frac{3}{6}$
$F_C = \frac{1}{6}; S_G = \frac{5}{6}$

Fig. 6. The overview of hybrid MTD, EVADE, with the greedy and DRL-based MTD.

sparse network. We derive networks of comparable size for the medium-dense and dense networks with the sparse network. To this end, we generate them using the following procedure: (1) Rank all nodes in the original network by the degree in descending order, and (2) Identify a new network as the largest connected component of the induced subgraph consisting of nodes with rankings from 1 to 1000.

Parameterization. In a given network, $G(V, E)$, the proposed hybrid MTD is triggered periodically to shuffle the network topology aiming to minimize network vulnerability while maintaining network connectivity. Specifically, greedy MTD will be triggered per ADC_g min. while the DRL-based MTD will be triggered per ADC_{DRL} min. We simulate epidemic attacks for 60 rounds, where the attack rounds happen per t_{atk} min. In each attack round, new attackers will arrive based on the Poisson distribution with λ, which is the expected number of new attackers arriving in one attack round. Then all existing attackers in the network will perform attacks once and all nodes' attributes (i.e., na_i and nc_i) will be updated accordingly. After attackers perform epidemic attacks, the network G is evaluated in terms of the size of a giant component and the fraction of the compromised nodes. We also assume that inside attackers can perform *state manipulation attacks* (SMAs), with probability P_s. We assume that the DRL agent can detect such attacks with detection rate D_r. If the SMA is detected, the DRL agent will use the previous stored system state, or it will use the falsified state otherwise. To estimate the expected vulnerabilities of nodes and edges (i.e., $\mathbf{V}_V$ and $\mathbf{V}_E$) in `VREN`, we use 500 attack simulations (i.e., $n_a = 500$). We use four different undirected networks, as described earlier. Initially all nodes are set as active and uncompromised. We use 100 simulation runs and show the mean results for each data point in our experiments. We evaluate all four DRL-based schemes after $n_e = 500$ training episodes. Table 1 summarizes the key design parameters, their meanings, and default values used. Note that the default values are fine-tuned to balance the performance and efficiency of proposed schemes.

Metrics. We use the following **metrics**, which are two objectives considered in the DRL agent's reward function:

- **Size of the giant component ($\mathcal{S}_G$)**: This metric captures the degree of network connectivity composed of non-compromised, active nodes in a network, and measured by $\mathcal{S}_G = \frac{N_G}{N}$, where N is the total number of nodes in the network and N_G is the number of nodes in the giant (largest) connected component. Higher $\mathcal{S}_G$ means higher network resilience under epidemic attacks.
- **Fraction of compromised nodes ($\mathcal{F}_C$)**: This measures the fraction of the number of compromised nodes due to epidemic attacks, including both detected and non-detected compromised nodes by the NIDS over the total number of nodes N in a network. This includes both currently infected (not detected by the NIDS) and removed (previously infected and detected by the NIDS) nodes. $\mathcal{F}_C$ is computed by $\mathcal{F}_C = \frac{N_C}{N}$, where N_C is the total number of compromised nodes after epidemic attacks on a given network.

Comparing Schemes. We consider the following algorithms to identify the budget pairs for network topology adaptations. For DRL algorithms used in `EVADE`, we use Proximal Policy Optimization (PPO) and Deep-Q Network (DQN) because DQN represents the most common DRL algorithm and PPO is known for its fast convergence and efficient hyperparameter optimization. Table 2 lists the fine-tuned parameters and their values used in DQN and PPO. We compare the performance of the following schemes in our experiments:

- **PPO with** `EVADE` uses PPO that adopts the actor-critic architecture to find the optimal policy for the DRL agent along with `VREN` and `FSS`.
- **S-G-PPO with** `EVADE` uses a greedy MTD to obtain the initial budget and then uses PPO.
- **DQN with** `EVADE` employs deep neural networks parameterized by $\boldsymbol{\theta}$ to represent an action-value function (i.e., Q function) along with `VREN` and `FSS`. In this scheme, the DRL agent is assumed to fully observe the environment.

- **S-G-DQN with `EVADE`** uses greedy MTD to obtain the initial budget and then uses DQN to further refine the results.
- **GA** is based on the genetic algorithm proposed in [15]. The agent selects an optimal network topology among n_s random generated candidate topologies based on Eq. (1). This scheme triggers MTD per 5 min.
- **Random** is a baseline scheme where the DRL agent first selects b_R at random and then selects b_A at random. This scheme triggers MTD per 5 min.

Table 1. Key Design Parameters, Their Meanings and Default Values

Param.	Meaning	Value
n_a	Number of attack simulations to determine vulnerabilities	500
n_r	Number of simulation runs	100
n_e	Training episodes of DRL-based schemes	500
n_s	Number of the random sample topologies generated by GA	10
N	Total number of the nodes in a network	200
d_a	Upper hop bound for edge addition	3
ζ	Intrusion detection probability	0.9
P_{fn}, P_{fp}	False negative or positive probability	0.1, 0.05
$\mathbf{x}$	Degree of software vulnerability	0.5
p	Connection probability between pairs of nodes in an ER network	0.05
l	Number of software packages available	5
λ	Expected number of new attackers arriving in each attack round	2
B	Upper bound of the total adaptation budget	800
ρ_1, ρ_2	Length parameters of the expanded triangle	64
P_s	Probability of state manipulation attacks	0.3
D_r	Detection rate of state manipulation attacks	0.8
ADC_g	Adaptation cycle for the greedy MTD (min.)	5
ADC_{DRL}	Adaptation cycle for the DRL-based MTD (min.)	20
T	Number of attack rounds	60
t_{atk}	The time interval a set of attackers arrive in a given network	1

7 Simulation Experimental Results

Algorithmic Complexity Analysis of `EVADE`. We discuss the algorithmic complexity of S-G-DQN/S-G-PPO with `EVADE`, DQN/PPO with `EVADE`, GA, and Random. Table 3 shows the asymptotic complexities in Big-O notation of the six schemes considered in this work. We notice that S-G-DQN/S-G-PPO with `EVADE` incurs the similar low cost as DQN/PPO with `EVADE` because the proposed `FSS` can significantly decrease the number of trajectories per training episode. Note that n_e refers to the training episode, B is the maximum total adaptation budget allowed (i.e., upper bound budget), n_a is the number of attack simulations, and n_s is the number of random sample topologies generated. Since the GA and

Random algorithms are rule-based and do not have the training phase, their complexities are represented by n_a and n_s. Furthermore, our proposed DRL schemes could be faster than GA given a high convergence speed, i.e., $n_e << n_s$. Table 3 shows that the GA and Random are the most efficient algorithms among all while showing poor performance in $\mathcal{S}_G$ and $\mathcal{F}_C$.

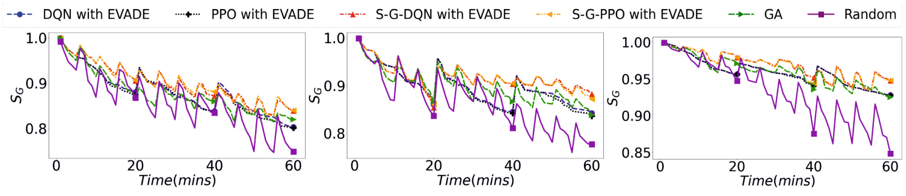
Comparative Analyses. For the three real network topologies, we conducted comprehensive comparative analyses of two types of hybrid schemes (i.e., S-G-DQN, S-G-PPO), two DRL schemes (i.e., DQN, PPO), and two baselines (i.e., GA and Random). In Figs. 7 and 8, we observe that when MTD operates for an hour, $\mathcal{S}_G$ decreases while $\mathcal{F}_C$ increases due to the incoming attack frequency λ.

Table 2. DRL Parameters and Values

Model	Parameter	Value
PPO	Discounting factor	0.9
	Actor learning rate	0.008
	Critic learning rate	0.08
	Clipping range	0.5
	Number of epoch when optimizing the surrogate loss	10
DQN	Discount factor	0.9
	Learning rate	0.08
	Exploration decay rate	0.85
	Minimum exploration rate	0.001

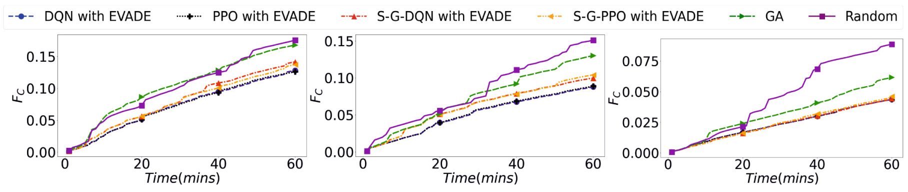
Table 3. Asymptotic Complexity Analysis of the Compared Schemes

Scheme	Complexity
S-G-DQN/S-G-PPO with `EVADE`	$O(n_e \times \lceil \log_2 B \rceil \times n_a)$
DQN/PPO with `EVADE`	$O(n_e \times \lceil \log_2 B \rceil \times n_a)$
GA	$O(n_s \times n_a)$
Random	$O(n_a)$



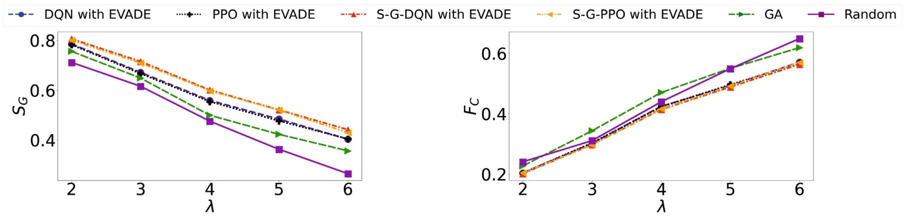
(a) Dense network (b) Medium network (c) Sparse network

Fig. 7. Comparative performance analysis of the six MTD schemes with respect to the simulation time in terms of the size of a giant component ($\mathcal{S}_G$).



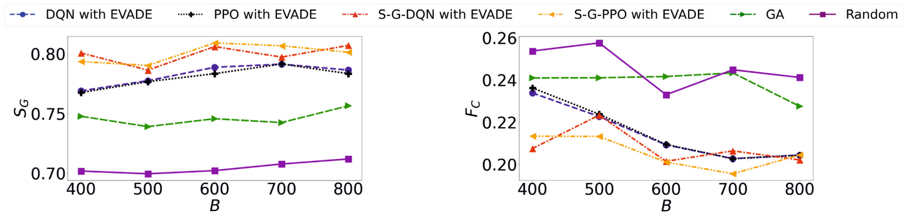
(a) Dense network (b) Medium network (c) Sparse network

Fig. 8. Comparative performance analysis of the six MTD schemes with respect to time in terms of the fraction of compromised nodes ($\mathcal{F}_C$).



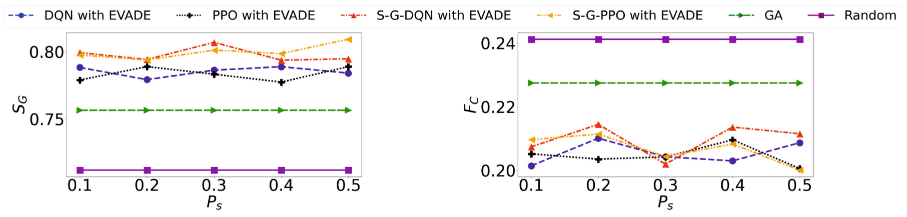
(a) Size of the giant component ($\mathcal{S}_G$) (b) Fraction of the compromised nodes ($\mathcal{F}_C$)

Fig. 9. Effect of varying the attack frequency (λ).



(a) Size of the giant component ($\mathcal{S}_G$) (b) Fraction of the compromised nodes ($\mathcal{F}_C$)

Fig. 10. Effect of varying the upper bound of the total adaptation budget (B).



(a) Size of the giant component ($\mathcal{S}_G$) (b) Fraction of the compromised nodes ($\mathcal{F}_C$)

Fig. 11. Effect of varying probability of state manipulation attacks (P_s).

There are some bumps in $\mathcal{S}_G$ curves where each bump is aligned with the time the MTD is triggered for the network topology adaptations. With the same adaptation frequency, GA performs worse than hybrid schemes, S-G-DQN/S-G-PPO, and DQN/PPO (without greedy). This shows the effectiveness of our proposed hybrid MTD schemes and DRL-based MTD schemes. In general, `EVADE`-based schemes outperform under the sparse network. Based on Eq. (3), network vulnerability is related to the sum of edge vulnerabilities. Thus, a fewer number of edges means less edge vulnerability for each edge and thus less network vulnerability overall. Furthermore, the edge number also affects the effectiveness of `VREN`. Less edge number leads to better convergence of rankings given by `VREN`, thus showing better adaptation performance. This is shown in the results where all schemes perform worst and our proposed `EVADE`-based schemes outperform least in a dense network. The overall performance order in the two metrics (i.e., $\mathcal{S}_G$ and $\mathcal{F}_C$) is: S-G-PPO with `EVADE` $\approx$ S-G-DQN with `EVADE` $\geq$ PPO with `EVADE` $\approx$ DQN with `EVADE` $\geq$ GA $\geq$ Random.

Sensitivity Analyses. For the Erdös-Rényi (ER) random network [33], we conducted in-depth sensitivity analyses of two types of hybrid schemes (i.e., S-G-DQN, S-G-PPO), two DRL schemes (i.e., DQN, PPO), and two baselines (i.e., GA and Random) under varying a wide range of the expected number of new attackers arriving in the network λ, the upper bound of the total adaptation budget B, and the probability of state manipulation attacks P_s.

Figure 9 shows the effect of varying the expected number of new attackers arriving in the network, λ, on the performance of the six schemes in terms of the two metrics in the network. We observe that increasing attack frequency (λ) decreases $\mathcal{S}_G$ while increasing $\mathcal{F}_C$. The hybrid approaches outperform other schemes as they use DRL and greedy MTD with a higher adaptation frequency. Figure 10 explains how the total adaptation budget (B) can affect the performance of the six schemes in terms of the two metrics in the given network. Higher B enables the increased $\mathcal{S}_G$ while decreasing $\mathcal{F}_C$ for all MTD schemes. This result implies that higher budgets enable the agent to find more desirable network topologies and lead to higher performance. Figure 11 shows how the different levels of SMAs (P_s) impact the performance of the six MTD schemes using the two metrics. We found that our `EVADE` outperforms other baselines and counterparts and clearly shows high resilience against SMAs. Recall that SMAs can perform attacks only with DRL-based approaches.

8 Limitations

The current work has the following limitations:

- The performance overhead introduced by running DRL in an SDN controller is not considered in the objective function defined in Eq. 1. This may affect the scalability of proposed schemes.
- The current attack model is not inclusive enough. Adaptive attacks against DRL agents or SDN/OpenFlow switches are not considered.

– The current work is empirical in nature. There is a lack of evaluation and discussion on attack coverage.

9 Conclusions

– We proposed an efficient moving target defense (MTD) mechanism, called `EVADE` with a fractal-based environment (`FSS`) to substantially lower the training time of DRL algorithms. Using `FSS` significantly reduces the convergence time to an (close-to) optimal solution for the network adaptation budgets.
– We developed a vulnerability-aware ranking algorithm, called `VREN`, as one of the key design features in `EVADE`, to make strategic edge adaptations and achieve highly efficient and effective network topology reconfigurations.
– Our `EVADE` uses a density optimization (`DO`)-based greedy algorithm to further reduce the search space for DRL algorithms, along with `VREN`. Using `DO` enabled our hybrid MTD approach to converge even faster with a smaller solution space and result in the outperformance of `EVADE` over the existing counterparts.
– Our `EVADE` was evaluated based on two different DRL algorithms, including Deep Q-learning Networks (DQN) and Proximal Policy Optimization (PPO), along with their corresponding hybrid approaches (i.e., S-G-DQN with `EVADE` and S-G-PPO with `EVADE`), and two existing baselines (i.e., GA and Random). The hybrid `EVADE` showed acceptable asymptotic complexity compared to their effectiveness. In addition, they showed higher performance than the counterpart and baseline schemes in minimizing system vulnerability while maintaining comparable or better network connectivity.

References

1. Achleitner, S., Porta, T.L., McDaniel, P., Sugrim, S., Krishnamurthy, S.V., Chadha, R.: Deceiving network reconnaissance using SDN-based virtual topologies. IEEE Trans. Netw. Serv. Manage. **14**, 1098–1112 (2017)
2. Anwar, A.H., Leslie, N.O., Kamhoua, C., Kiekintveld, C.: A game theoretic framework for software diversity for network security. In: GameSec 2020. LNCS, vol. 12513, pp. 297–311. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64793-3_16
3. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: a brief survey. IEEE Signal Process. Mag. **34**(6), 26–38 (2017)
4. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: A brief survey of deep reinforcement learning. arXiv preprint: arXiv:1708.05866 (2017)
5. Chai, X., Wang, Y., Yan, C., Zhao, Y., Chen, W., Wang, X.: DQ-MOTAG: deep reinforcement learning-based moving target defense against DDoS attacks. In: 2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC), pp. 375–379. IEEE (2020)
6. Cho, J.H., et al.: Toward proactive, adaptive defense: a survey on moving target defense. IEEE Commun. Surv. Tutorials **22**(1), 709–745 (2020)
7. Colbourn, C.: Network resilience. SIAM J. Algebraic Discrete Methods **8**(3), 404–409 (1987)

8. CVSS, Common Vulnerability Scoring System (CVSS), National Vulnerability Database (2022). https://www.first.org/cvss/
9. Darvariu, V.-A., Hailes, S., Musolesi, M.: Improving the robustness of graphs through reinforcement learning and graph neural networks. arXiv preprint: arXiv:2001.11279 (2020)
10. Das, D.: A fuzzy multiobjective approach for network reconfiguration of distribution systems. IEEE Trans. Power Delivery **21**(1), 202–209 (2005)
11. Desai, A., Milner, S.: Autonomous reconfiguration in free-space optical sensor networks. IEEE J. Sel. Areas Commun. **23**(8), 1556–1563 (2005)
12. Eghtesad, T., Vorobeychik, Y., Laszka, A.: Adversarial deep reinforcement learning based adaptive moving target defense. In: GameSec 2020. LNCS, vol. 12513, pp. 58–79. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64793-3_4
13. Mavoungou, S., et al.: Survey on threats and attacks on mobile networks. IEEE Access **4**, 4543–4572 (2016)
14. Fernández, N., et al.: Virtual topology reconfiguration in optical networks by means of cognition: evaluation and experimental validation. IEEE/OSA J. Opt. Commun. Networking **7**(1), A162–A173 (2015)
15. Ge, M., Cho, J.-H., Kim, D., Dixit, G., Chen, I.-R.: Proactive defense for internet-of-things: moving target defense with cyberdeception. ACM Trans. Internet Technol. (TOIT) **22**(1), 1–31 (2021)
16. Grimmett, G.: Percolation and disordered systems. In: Bernard, P. (ed.) Lectures on Probability Theory and Statistics. LNM, vol. 1665, pp. 153–300. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0092620
17. Hole, K.J.: Diversity reduces the impact of malware. IEEE Secur. Privacy **13**(3), 48–54 (2015)
18. Hong, J.B., Kim, D.S.: Assessing the effectiveness of moving target defenses using security models. IEEE Trans. Dependable Secure Comput. **13**(2), 163–177 (2016)
19. Hong, J.B., Yoon, S., Lim, H., Kim, D.S.: Optimal network reconfiguration for software defined networks using shuffle-based online MTD. In: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), pp. 234–243 (2017)
20. Huang, C., Zhu, S., Erbacher, R.: Toward software diversity in heterogeneous networked systems. In: Atluri, V., Pernul, G. (eds.) DBSec 2014. LNCS, vol. 8566, pp. 114–129. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43936-4_8
21. Huang, C., Zhu, S., Guan, Q., He, Y.: A software assignment algorithm for minimizing worm damage in networked systems. J. Inf. Secur. Appl. **35**, 55–67 (2017)
22. Jensen, T.R., Toft, B.: Graph Coloring Problems, vol. 39. John Wiley & Sons, Hoboken (2011)
23. Kaur, T., Baek, J.: A strategic deployment and cluster-header selection for wireless sensor networks. IEEE Trans. Consum. Electron. **55**(4), 1890–1897 (2009)
24. Kim, S., et al.: DIVERGENCE: deep reinforcement learning-based adaptive traffic inspection and moving target defense countermeasure framework. IEEE Trans. Netw. Serv. Manag. **19**, 4834–4846 (2022)
25. Kohl, N., Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: Proceedings. ICRA2004, vol. 3, pp. 2619–2624. IEEE (2004)
26. Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
27. Leong, A.S., Quevedo, D.E., Ahlén, A., Johansson, K.H.: On network topology reconfiguration for remote state estimation. IEEE Trans. Autom. Control **61**(12), 3842–3856 (2016)

28. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graphs over time: densification laws, shrinking diameters and possible explanations. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, pp. 177–187 (2005)
29. Leskovec, J., Mcauley, J.: Learning to discover social circles in ego networks. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
30. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
31. Najjar, W., Gaudiot, J.L.: Network resilience: a measure of network fault tolerance. IEEE Trans. Comput. **39**(2), 174–181 (1990)
32. Newman, M.: Networks: An Introduction. Oxford University Press, Oxford (2010)
33. Newman, M., Watts, D.: Scaling and percolation in the small-world network model. Phys. Rev. E **60**(6), 7332–7342 (1999)
34. O'Donnell, A.J., Sethu, H.: On achieving software diversity for improved network security using distributed coloring algorithms. In: Proceedings of the 11th ACM Conference on Computer and Communications Security, pp. 121–131. ACM (2004)
35. University of Washington, Rocketfuel maps and data, April 2003. http://www.cs.washington.edu/research/networking/rocketfuel/
36. Rao, R.S., Ravindra, K., Satish, K., Narasimham, S.: Power loss minimization in distribution system using network reconfiguration in the presence of distributed generation. IEEE Trans. Power Syst. **28**(1), 317–325 (2012)
37. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR, vol. abs/1707.06347 (2017). http://arxiv.org/abs/1707.06347
38. Singh, S., Litman, D., Kearns, M., Walker, M.: Optimizing dialogue management with reinforcement learning: experiments with the NJFun system. J. Artif. Intell. Res. **16**, 105–133 (2002)
39. Sterbenz, J.P., et al.: Resilience and survivability in communication networks: strategies, principles, and survey of disciplines. Comput. Netw. **54**(8), 1245–1265 (2010)
40. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning, 1st edn. MIT Press, Cambridge (1998)
41. Temizkan, O., Park, S., Saydam, C.: Software diversity for improved network security: optimal distribution of software-based shared vulnerabilities. Inf. Syst. Res. **28**(4), 828–849 (2017)
42. Touhiduzzaman, M., Hahn, A., Srivastava, A.K.: A diversity-based substation cyber defense strategy utilizing coloring games. IEEE Trans. Smart Grid **10**, 5405–5415 (2018)
43. Tozer, B., Mazzuchi, T., Sarkani, S.: Optimizing attack surface and configuration diversity using multi-objective reinforcement learning. In: IEEE 14th International Conference on Machine Learning and Applications (ICMLA), pp. 144–149. IEEE (2015)
44. Wan, Z., Mahajan, Y., Kang, B.W., Moore, T.J., Cho, J.-H.: A survey on centrality metrics and their implications in network resilience (2020)
45. Yang, Y., Zhu, S., Cao, G.: Improving sensor network immunity under worm attacks: a software diversity approach. In: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing, ser. MobiHoc 2008, pp. 149–158 (2008)
46. Yang, Y.: Improving sensor network immunity under worm attacks: a software diversity approach. Ad Hoc Networks, vol. 47, no. Supplement C, pp. 26–40 (2016)

47. Zhang, Q., Cho, J.H., Moore, T.J.: Network resilience under epidemic attacks: deep reinforcement learning network topology adaptations. In: IEEE Global Communications Conference (GLOBECOM), pp. 1–7 (2021)
48. Zhang, Q., Cho, J.H., Moore, T.J., Chen, R.: Vulnerability-aware resilient networks: Software diversity-based network adaptation. IEEE Trans. Netw. Serv. Manag. (2020)
49. Zhang, Q., Cho, J.H., Moore, T.J., Nelson, F.F.: DREVAN: deep reinforcement learning-based vulnerability-aware network adaptations for resilient networks. In: IEEE Conference on Communications and Network Security (CNS), pp. 137–145 (2021)
50. Zhang, Q., Mohammed, A.Z., Wan, Z., Cho, J.H., Moore, T.J.: Diversity-by-design for dependable and secure cyber-physical systems: a survey (2020)
51. Zhang, T., et al.: DQ-RM: deep reinforcement learning-based route mutation scheme for multimedia services. In: 2020 IEEE International Wireless Communications and Mobile Computing (IWCMC), pp. 291–296 (2020)
52. Zhang, Y., Murata, M., Takagi, H., Ji, Y.: Traffic-based reconfiguration for logical topologies in large-scale WDM optical networks. J. Lightw. Technol. **23**(10), 2854–2867 (2005)
53. Zhu, M., Hu, Z., Liu, P.: Reinforcement learning algorithms for adaptive cyber defense against heartbleed. In: Proceedings of the First ACM Workshop on Moving Target Defense, pp. 51–58 (2014)