# Network Resilience Under Epidemic Attacks: Deep Reinforcement Learning Network Topology Adaptations

Qisheng Zhang, Jin-Hee Cho
*Department of Computer Science, Virginia Tech*
Falls Church, VA, USA
{qishengz19, jicho}@vt.edu

Terrence J. Moore
*US Army Research Laboratory*
Adelphi, MD, USA
terrence.j.moore.civ@army.mil

***Abstract*—In this work, we proposed a Deep reinforcement learning (DRL)-based NETwork Adaptations for network Resilience algorithm, namely DeepNETAR, which aims to generate robust network topologies against epidemic attacks. In DeepNETAR, a DRL agent aims to generate a robust network topology against epidemic attacks by removing vulnerable edges or adding the least vulnerable edges, given multiple objectives of system security and performance. Most existing network topology adaptation algorithms have used the size of the giant component (SGC) to ensure service availability based on network connectivity. However, in real communication networks, where packets may be dropped either from the presence of inside attackers or congestion on long routes, a larger SGC does not necessarily ensure higher service availability. In addition, for the DRL agent to learn fast and handle multiple, conflicting system objectives, we considered vulnerability-based selection of adaptable edge candidates, fractal-based solution search, and diverse reward functions aiming to achieve multi-objective optimization. Via extensive simulation experiments, we analyzed what DeepNETAR-based schemes using different objectives can achieve those two conflicting system objectives and comparing existing and baseline counterparts.**

***Index Terms*—Network resilience, epidemic attacks, service availability, network vulnerability, deep reinforcement learning.**

## I. Introduction

Network scientists have substantially studied network resilience in terms of how well nodes are connected, measured by the size of the giant component (SGC) in a network [2]. Although the SGC has been predominantly used as a key metric to represent network resilience, whether a larger SGC can guarantee higher service availability (e.g., high message delivery) in networks has not been investigated. Even if a network is maintained with a larger SGC, it does not guarantee safe delivery of correct messages due to inside attackers dropping packets, or a higher probability of packets dropped in long paths caused by network congestion.

Our prior works extended the concept of network resilience by embracing both service availability and system security in a network. They mainly focused on minimizing system vulnerability, particularly software vulnerability, by taking diversity-by-security-design approaches in network edge adaptations algorithms [21], [22]. However, those efforts primarily focused on minimizing security vulnerability, not ensuring service availability in the actual delivery of correct messages in the network with insider attackers or under network congestion.

In this work, we consider both the security vulnerability and the service availability based on a deep reinforcement learning (DRL)-based network adaptation algorithm. The proposed scheme is called DeepNETAR representing *a Deep reinforcement learning-based NETwork Adaptations for network Resilience algorithm*. DeepNETAR will identify an optimal pair of network adaptations (i.e., numbers of edge additions and removals) that can meet conflicting system objectives of maximizing service availability and minimizing network vulnerability.

We made the following **key contributions** in this work:

- The proposed DeepNETAR can handle various objective functions for the DRL agent to maximize as its reward. The network topology generated by the DRL agent is to be robust against epidemic attacks to ensure minimum security vulnerability and maximum system service availability. We investigated what types of objective functions the DRL agent uses can more effectively achieve these two conflicting objectives than other counterparts.
- The proposed DeepNETAR consists of two algorithms that can efficiently identify an optimal network adaptation budget strategy (i.e., determining the numbers of edge additions and removals) to minimize security vulnerability and maximize service availability. We developed a vulnerability ranking algorithm of edges and nodes (VREN) to select candidates for edge adaptations (i.e., removing the most vulnerable edges while adding the least vulnerable edges). In addition, we presented a fractal-based solution search algorithm (FSS) to substantially reduce the solution space, which allows the DRL agent to shorten its learning curve.
- Via extensive simulation experiments, we found that a larger size of the giant component is not necessarily aligned with higher service availability (i.e., correct message delivery ratio). In addition, allowing a higher fraction of compromised nodes can increase actual service availability due to the existence of more paths available between two nodes.

## II. Related Work

In network science, network resilience (or robustness) has been heavily studied using percolation theory. Percolation theory originated from statistical physics to mathematically formulate the flow of liquid via a medium [11]. Network science researchers have commonly used the concept of site or bond percolation to select a node or edge, respectively, to remove or add to model epidemics on networks of various types, such as the transmission of epidemic diseases, computer malware or virus spreading, and herding behavior (e.g., purchasing items or adopting products) [8], [19]. In addition, percolation theory is used to study connectivity-based network robustness [2], [19], [1], network reliability [15], and epidemics [3], [17]. Computer scientists adopted percolation theory to propose software diversity methods for solving a software assignment problem [20] under epidemic attacks. Using percolation theory is highly relevant to developing topology-aware software diversity methods because the network topology plays an important role in the propagation of malware or computer viruses [19]. Most works using percolation theory to investigate network resilience to maximize the size of the giant component as a key indicator of network connectivity. However, prior work has rarely investigated whether a network with a larger size of the giant component can provide high service availability in safely delivering correct messages.

Deep reinforcement learning (DRL) has been employed to develop graph embedding-based topology adaptation algorithms to effectively deal with adversarial attacks injected in graph neural networks (GNN). Structure2vec (S2V) was used to obtain a node embedding that a DRL agent can learn two independent Q-functions in deep Q-learning (DQN) under adversarial attacks, removing a single edge in GNNs [6]. A network topology adaptation strategy was also proposed using edge additions from an empty network based on an S2V variant [7]. Network topology shuffling-based moving target defense (MTD) techniques have also been proposed to handle Distributed Denial-of-Service (DDoS) attacks in CPSs [4]. In addition, periodic route redirection techniques have been proposed where DQN agents make route mutation decisions to prevent DoS and targeted attacks in routes [23]. However, these works above primarily focused on reducing network vulnerability without being concerned much about service availability.

## III. Problem Statement

In this work, we are interested in identifying a pair of network adaptation budgets by edge additions, $b_A$, and edge removals, $b_R$, based on DRL (i.e., DQN), where $0 \leq b_A + b_R \leq B$ with $B$ being the upper bound of the total adaptations budget. When an original network topology $G$ is given, we let the DRL agent produce an adapted network $G'$ by adding or removing edges using $(b_A, b_R)$, respectively. Formally, we seek the DRL agent to solve the following objective:

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

where $f(G)$ returns the values depending on the evaluation function, $f(\cdot)$, that can be defined according to different system objectives as below:

- **O-SG**: $f : G \mapsto \mathcal{S}_G(G) - \mathcal{F}_C(G)$ for the schemes that aim to maximize the size of the giant component and minimize the fraction of compromised nodes. We will append 'SG' at the end of a scheme that considers this evaluation function.
- **O-MD**: $f : G \mapsto \mathcal{P}_{MD}(G) - \mathcal{F}_C(G)$ for the schemes that aim to maximize service availability in terms of the delivery ratio of correct messages and minimize the fraction of compromised nodes. We will append 'MD' at the end of a scheme with this evaluation function.
- **O-SG-MD**: $f : G \mapsto \mathcal{S}_G(G) + \mathcal{P}_{MD}(G) - \mathcal{F}_C(G)$ for the schemes to maximize both the size of the giant component and the service availability and minimize the fraction of compromised nodes. We will append 'SG-MD' at the end of a scheme using this evaluation function.

We described the metrics associated with these three objectives in Section VI. Note that $f(\cdot)$ will be generated based on the two algorithms (i.e., VREN and FSS) that can contribute to efficiently identifying an optimal $(b_A, b_R)$ by the DRL agent, as described in Section V. Note that it is a non-trivial task for the DRL agent to identify an optimal $(b_A, b_R)$ that can meet multiple objectives, which is well known as the nature of multi-objective optimization (MOO) problems [5]. This implies that the optimal solutions identified by the DRL do not necessarily achieve the best performance in all objectives. We elaborate this discussion with more details in Section VII.

## IV. System Model

**Network Model**: We assume a centralized control system where a controller exists to run our proposed DeepNETAR framework. Some examples of such systems include a software-defined network (SDN) using one SDN controller [14], an Internet-of-Things (IoT) using an edge server or central cloud server, or a hierarchical wireless sensor network (WSN) or a mobile ad-hoc network (MANET) with a centralized controller [13]. Although our work considered a single DRL agent to make a centralized decision for reconfiguring a robust network topology, it can be easily extended to consider multiple DRL agents for a large-scale network. We leave this for our future research as outlined in Section VIII.

We assume that the centralized controller is informed of network connectivity and software package information of nodes in the network. This will allow the DRL agent in the centralized controller to make edge shuffling decisions. The considered network can be represented by an undirected graph, $G = (V, E)$, where $V$ is a set of nodes and $E$ is a set of edges existing in $G$. We consider temporal network topologies that can be altered by the dynamics of the network, which are often caused by node failure or compromise from epidemic attacks as well as edge adaptations made by the proposed DeepNETAR. The status of the temporal network topology will be tracked by maintaining an adjacency matrix $\mathbf{A}$ with entry $a_{ij} = 1$ when there is an edge between nodes $i$ and $j$ and $a_{ij} = 0$ when no edges between them.

We leverage DRL to identify a network topology that can achieve multiple system objectives of maximizing service availability and network connectivity while minimizing security vulnerability. We design the DRL agent running on the centralized coordinator to make network adaptation decisions to identify an optimal budget for adding and removing edges. The DRL agent is assumed to periodically run the proposed network topology adaptation algorithm, DeepNETAR, periodically to meet a given set of multiple system objectives. We described more details of the DeepNETAR in Section V.

We consider a network-based intrusion detection system (NIDS) exists in a given system to detect compromised nodes by attackers. To model the process of compromising legitimate nodes and detecting/removing compromised nodes, we leverage the Susceptible-Infected-Removed (SIR) epidemic model [19]. The node compromise rate by the attackers is $\beta_{ji}$ as shown in Eq. (2) while the detection/removal rate of the nodes detected as compromised by the NIDS is $\gamma$. As a response to the nodes detected as compromised, we assume a rekeying process is used to change secret keys and block unauthorized nodes' access to the system. Note that developing an effective and efficient NIDS is beyond the scope of this work. Hence, we simply model the key characteristics of the NIDS with the probabilities of false positives and negatives, denoted by $P_{fp}$ and $P_{fn} = 1 - \gamma$, respectively.

**Node Model**: We consider the the following attributes of each node: (1) a node being alive or failed with $na_i = 1$ or $na_i = 0$, respectively; (2) node $i$ being compromised or not with $nc_i = 1$ or $nc_i = 0$, respectively; (3) node $i$'s software package denoted by $s_i$, which is an integer in $[1, l]$ where $l$ is the number of software packages used in the network; and (4) node $i$'s software vulnerability, denoted by $sv_i$, estimated based on the Common Vulnerabilities and Exposures (CVE) [10] score in the Common Vulnerability Scoring System (CVSS) [18]. The CVSS estimates a node's average vulnerability from multiple vulnerabilities as a real number ranged in $[0, 10]$. We use its normalized value for $sv_i$ as a real number in $[0, 1]$.

**Attack Model**: *Epidemic attacks* are considered where an attacker $j$ can spread malware or computer viruses and compromise adjacent nodes with probability $\beta_{ji}$, which is given by [12]:

$$\beta_{ji} = \begin{cases} 1 & \text{if } \sigma_j(s_i) > 0; \\ sv_i & \text{otherwise.} \end{cases} \quad (2)$$

Here $\beta_{ji} = 1$ means that attacker $j$ already knows the vulnerability of software package $s_i$ (i.e., $\sigma_j(s_i) > 0$) and can immediately compromise another node with $s_i$. Otherwise, attacker $j$ needs to make efforts to compromise a given node with $s_i$ based on its degree of vulnerability, $sv_i$. Recall that $\beta_{ji}$ is used as an infection rate in the SIR model [19].

## V. Proposed DeepNETAR Framework

In this section, we provide the overview of our proposed DeepNETAR in terms of selecting edges to adapt for removing or adding and searching optimal budget pairs (i.e., $(b_A, b_R)$) and the process of DRL-based budget adaption.

**Algorithm 1** **Vulnerability Ranking of Edges and Nodes (VREN)**

$G \leftarrow$ Original network
$N \leftarrow$ Total number of nodes in the network
$b_A \leftarrow$ Addition budget
$b_R \leftarrow$ Removal budget
$n_a \leftarrow$ Total number of attack simulations to identify edge and node vulnerabilities based on their appearance on attack paths
$k \leftarrow$ Hop distance upper bound for edge addition
$G' =$ **VREN**$(G, b_A, b_R)$
$V_V \leftarrow$ Node vulnerability levels, initialized as a vector of zeros
$V_E \leftarrow$ Edge vulnerability levels, initialized as a vector of zeros
**1.A: Edges Removals**
**for** $t \leftarrow 1$ to $n_a$ **do**
    $V_V^*, V_E^* =$ EpidemicAttacks$(G)$ ▷ Single attack simulation
    $V_E = V_E + V_E^*$
**end for**
$R_E \leftarrow$ rank $E$ in descending order with key $V_E$
$G^* \leftarrow$ remove the top $b_R$ edges from $G$ according to $R_E$
**1.B: Edges Additions**
**for** $t \leftarrow 1$ to $n_a$ **do**
    $V_V', V_E' =$ EpidemicAttacks$(G^*)$ ▷ Single attack simulation
    $V_V = V_V + V_V'$
**end for**
$R_V \leftarrow$ rank $V$ in ascending order with key $V_V$
$gc \leftarrow$ the giant component of $G^*$
$R_V^{gc}, R_V^{ngc} \leftarrow$ order $gc$ and the complement $ngc$ based on $R_V$
$G' = G^*$ ▷ All the following adaptations are on $G'$
**for** $i$ in $R_V^{ngc}$ **do**
    **for** $j$ in $R_V^{gc}$ **do**
        $G' \leftarrow$ Connect nodes $i, j$ and break if $dist(i, j) \in [2, k]$ ▷ $dist$ considers the length of a shortest path between nodes $i$ and $j$ and $k$ is the maximum number of hops allowed
    **end for**
**end for**
$b_A^{temp} \leftarrow \|R_V^{ngc}\|$
**for** $h \leftarrow 2$ to $N$ **do**
    $R_V^h \leftarrow$ top $h$ nodes in $R_V$
    **for** $i$ in $R_V^h$ **do**
        $R_V^{i-1} \leftarrow$ top $i-1$ nodes in $R_V$
        **for** $j$ in $R_V^{i-1}$ **do**
            break if $b_A^{temp} \geq b_A$
            **if** $dist(i, j) \leq k$ **then**
                $G' \leftarrow$ Connect nodes $i, j$ if $dist(i, j) \in [2, k]$
                $b_A^{temp} = b_A^{temp} + 1$
            **end if**
        **end for**
    **end for**
**end for**
**return** $G'$

**Vulnerability Ranking of Edges and Nodes (VREN)**: We denote budgets for adding or removing edges by $b_A$ and $b_R$, respectively, and propose a Vulnerability Ranking for Edges and Nodes algorithm, called *VREN*. For any budget pair $(b_A, b_R)$, VREN can perform network topology adaptations efficiently and effectively. Algorithm 1 details our proposed VREN procedure. Given a network with an underlying graph $G = (V, E)$, we denote the edge vulnerability levels and node (or vertex) vulnerability levels as $V_E$ and $V_V$, respectively. A node's vulnerability level is estimated based on the frequency

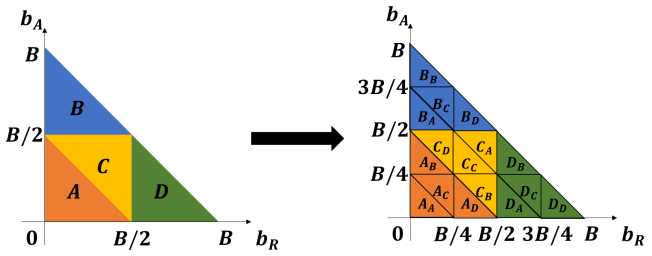



Fig. 1. Key idea of the Fractal-based solution search (FSS) to generate self-similar fractals for reducing solution search space in identifying optimal $(b_A, b_R)$.

of the node appearing on attack paths, representing the extent of the node being compromised by attackers. Similarly, an edge's vulnerability level is computed based on the frequency of the edge being used by attackers to compromise other nodes. Then, we run $n_a$ attack simulations on $G$ and calculate $V_E$ and $V_V$ (lines 11 – 14, 17 – 20). To this end, based on the frequency of nodes and edges appearing on attack paths, we implement `EpidemicAttacks`$(G)$ to obtain their vulnerability values. Lastly, based on these vulnerability levels $V_E$ and $V_V$, we rank edges and nodes (lines 15, 22) and use $R_E$ and $R_V$ to denote these ranks, respectively. Here, we sort $R_E$ in descending order and $R_V$ in ascending order. As $n_a$ increases, the $R_E$ and $R_V$ converge. Hence, it is critical to choose an appropriate $n_a$. We can estimate the expected fraction of compromised nodes ($\mathcal{F}_C$) in the network as follows:

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

where $N$ refers to the total number of nodes, $I$ is the set of epidemic attackers at the initial stage of attack, $P_i$ is the probability that node $i$ can be compromised, $P_I$ is the probability that the set $I$ is selected as the set of initial attackers, $P_i^I$ is the probability that node $i$ is compromised, given that $I$ is the set of initial attackers, $V_V^i$ is node $i$'s vulnerability level, and $V_E^{ij}$ is the vulnerability of an edge $e_{ij}$.

The VREN runs as follows: First, we calculate $R_E$ of a given $G = (V, E)$ and remove the top $b_R$ edges according to $R_E$ and a given budget pair $(b_A, b_R)$ (lines 11 – 16). After this phase, we refer to the intermediate adapted graph as $G^*$. Next, we attempt to restore the connectivity of $G^*$ by adding a minimum number of edges and call the adapted graph $G'$ (lines 18 – 30). Then, we calculate $R_V$ of $G'$ and add at most $b_A$ edges based on $R_V$ (lines 31 – 44). Instead of considering arbitrarily connected pairs of nodes in the removal process, we only consider disconnected nodes within $k$ hops distance in the addition process.

**Fractal-based Solution Search (FSS)**: We propose this *FSS* (Fractal-based Solution Search) to optimize the solution search process by restructuring the search space. Recall that Eq. (1) defines our objective as maximizing the differences between evaluation function values from the original network $G$ and the adapted network $G'$.

Since we aim to solve Eq. (1) within a triangle area $\{(b_A, b_R) | 0 \le b_A + b_R \le B\}$, we leverage the fractal structure of this area to obtain a solution search space wherein the DRL agent can efficiently identify an optimal solution. Fig. 1 shows the self-similar fractals generated by subdividing triangles. From each subdivision, we can have four self-similar parts, namely $A, B, C, D$. Based on this procedure, we can utilize the centroid to indicate each division and the nearest integer pair $(b_A, b_R)$ from the centroid to evaluate the division by $f(\cdot)$ in Eq. (1). The centroid of division $C$ is invariant under each subdivision. Since we consider $b_A, b_R$ as integers, we can sufficiently iterate the subdivision until two legs of each corresponding triangle have a length less than 1.

**DRL-based Budget Adaptation**: We propose a DRL environment where the DRL agent could add or remove edges with a given total edge adaptation budget. We describe the key components of the Markov Decision Process (MDP) considered in this work where DeepNETAR uses FSS with the following MDP components:

- **States**: We employ FSS and adopt the subdivision for each step. Hence, the DRL agent only needs $\lceil \log_2 B \rceil$ steps in total to find the optimal budget pair, where $B$ is the total upper bound adaptation budget. In each step $t$, state $s_t$ can be presented as a tuple, $s_t = (b_A^t, b_R^t, G_t')$, where $b_A^t$ and $b_R^t$ refer to the budget for adding and removing edges, respectively, based on the division evaluation at time $t$ where $G_t'$ refers to a network adapted at time $t$.
- **Actions**: We determine an action set $\mathbf{a}_t$ available at step $t$ based on the division ID for a next subdivision, which is represented by $\mathbf{a}_t = \{A, B, C, D\}$ where $1 \le t \le \lceil \log_2 B \rceil$.
- **Reward**: We denote a reward function at step $t$ by $\mathcal{R}(s_t, a_t, s_{t+1})$, based on the difference between the evaluation functions of two consecutive states by $\mathcal{R}(s_t, a_t, s_{t+1}) = f(G_{t+1}') - f(G_t')$, where $f(\cdot)$ can be defined based on different objective functions in each scheme, as discussed in Section III.

## VI. Experimental Setup

Each of the proposed algorithms will adapt an initially deployed network a single time. For the DRL schemes, the agent will adapt the network by optimizing the objective function in Eq. (1). The network is then seeded with an initial number of attackers $P_a$ that initiate epidemic attacks, which are then evaluated using metrics described below to measure the service availability, network connectivity, and system security. To estimate the vulnerabilities of the nodes and edges, we set the number of attack simulations $n_a = 500$ in the VREN procedure (see Algorithm 1). For the initially deployed network, we generate a random graph using the Erdös–Rényi (ER) model [9], with $N = 200$ nodes and

connection probability $p = 0.05$. Our experimental results are based on the average over $n_r = 200$ simulation runs. The evaluation of each DRL-based scheme (described at the end of this section) is conducted after $n_e = 1,000$ training episodes.

We use the following **metrics** for performance analysis: (1) **Delivery of correct messages ($\mathcal{P}_{MD}$)** measures the ratio of the number of correct messages delivered over the total number of the messages transmitted by a given set of source-destination pairs. This is a key indicator of measuring actual service availability of a system; (2) **Size of the giant component ($\mathcal{S}_G$)** measures the connectivity of the network consisting of active, non-compromised nodes. This simply captures how well nodes are connected as a single component; and (3) **Fraction of compromised nodes ($\mathcal{F}_C$)** counts the number of (both detected and undetected) compromised nodes due to epidemic attacks and normalizes this by the number of nodes $N$ in the network.

To measure the service availability of a given network based on $\mathcal{P}_{MD}$, we selected 20 pairs of active source-destination nodes at random. We allow each source to start the transmission by sending two packets to two randomly selected adjacent nodes to increase the delivery ratio of correct messages. The packet will be forwarded until reaching a destination node through the shortest path. If the destination does not receive a packet within the number of hops defined as the length of the shortest path, the packet is considered lost. If an attacker (i.e., compromised and undetected by the NIDS) receives a packet, it will drop a packet randomly with probability $P_d = 0.5$. If the attacker decides to forward the packet, it will modify the packet randomly with probability $P_m = 0.5$. An attacker can also send packets to a particular legitimate node. We set the probability of a node successfully forwarding a packet to $e^{-\lambda \cdot N_p}$ where $\lambda = 0.1$ is a constant to model a reasonable failure rate of packet delivery in a given network and $N_p$ is the number of packets received in a given node. An attacker also performs epidemic attacks with the infection probability based on Eq. (2). We summarized the key design parameters, their meanings, and default values used in Table I.

Given different evaluation functions as discussed in Section III (i.e., O-SG, O-MD, O-SG-MD), we use the following **six schemes for comparative performance analysis**: (1) **DQN-DeepNETAR-SG** utilizes DQN with neural networks parameterized by $\boldsymbol{\theta}$ to represent an action-value function (i.e., a Q function) assuming that a DRL agent can make full observations about the environment [16]. This scheme uses DeepNETAR in Section V with the evaluation function O-SG; (2) **DQN-DeepNETAR-MD** follows DeepNETAR with DQN and O-MD; (3) **DQN-DeepNETAR-SG-MD** follows DeepNETAR with DQN and O-SG-MD; (4) **Greedy-SG** uses the FSS algorithm to make decisions at each step by looking ahead one step and greedily choosing the subdivision that returns the maximum reward with O-SG; (5) **Optimal-SG** identifies adaptation budgets maximizing O-SG; and (6) **Random** is the baseline model where an edge removal budget $b_R$ and an edge addition budget $b_A$ are selected at random.

DQN is evaluated with or without using FSS or VREN. Given fixed adaptation budgets $(b_A, b_R)$, random adaptation (RA) is used as a baseline adaptation algorithm. We used linear search (LS) as a baseline alternative for FSS and RA as an baseline alternative for VREN. We call DQN with FSS and VREN as 'DQN with DREVAN,' DQN with LS and VREN as 'DQN with VREN,' DQN with FSS and RA as 'DQN with FSS,' and DQN with LS and RA as 'DQN.'

TABLE I
KEY DESIGN PARAMETERS, MEANINGS, AND DEFAULT VALUES

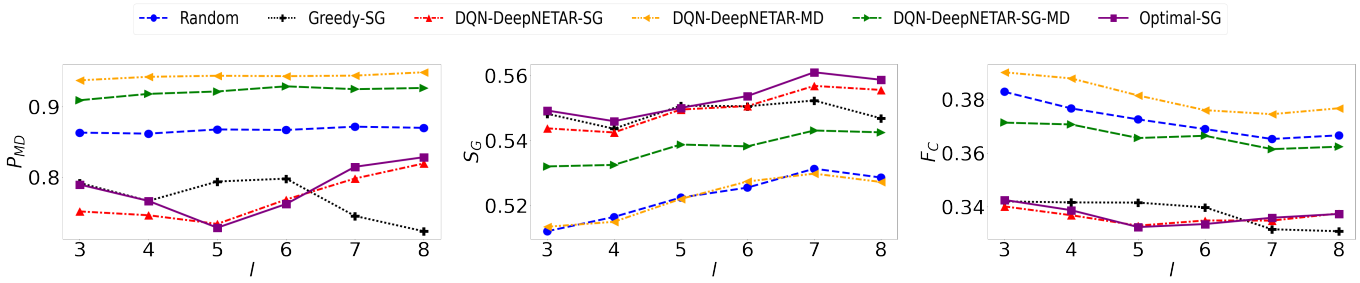
| Param. | Meaning | Value |
|---|---|---|
| $n_a$ | Number of attack simulations | 500 |
| $n_r$ | Number of simulation runs | 200 |
| $n_e$ | Training episodes of DRL-based schemes | 1000 |
| $N$ | Total number of nodes in a network | 200 |
| $k$ | Upper hop bound for edge addition | 3 |
| $\gamma$ | Intrusion detection probability | 0.9 |
| $P_{fn}, P_{fp}$ | False negative or positive probability | 0.1, 0.05 |
| $P_d$ | Package drop probability | 0.5 |
| $P_m$ | Package modification probability | 0.5 |
| $\lambda$ | Constant used in package forward failure rate | 0.1 |
| $\mathbf{x}$ | 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 |
| $P_a$ | Fraction of initial attackers in a network | 0.3 |
| $B$ | Upper bound of the total adaptation budget | 500 |

## VII. NUMERICAL RESULTS & ANALYSIS

This section demonstrates the experimental results obtained from the extensive comparative performance analysis of the following schemes: DQN-DeepNETAR-SG, DQN-DeepNETAR-MD, and DQN-DeepNETAR-SG-MD, and three counterparts (i.e., Random, Greedy-SG, and Optimal-SG). We varied the number of available software packages $l$ and the upper bound of the total adaptation budget $B$ to investigate their impact on the three metrics (i.e., $\mathcal{P}_{MD}$, $\mathcal{S}_G$, $\mathcal{F}_C$).

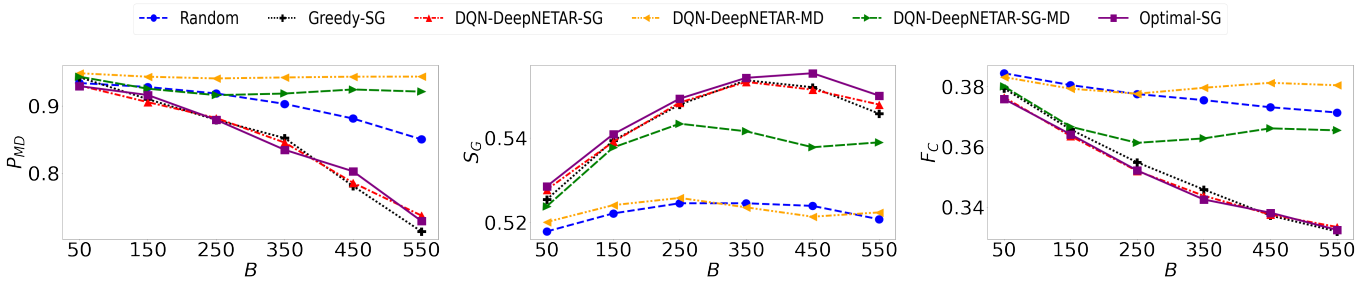
Fig. 2 shows how the different number of software packages available ($l$) affect the performance of each scheme in the three metrics in an ER network. We observed that larger $l$ introduces the increase of the delivery of correct messages ($\mathcal{P}_{MD}$) and size of the giant component ($\mathcal{S}_G$) while decreasing the fraction of compromised nodes ($\mathcal{F}_C$). Notice that high $l$ corresponds to a high level of software diversity, which leads to the low percentage of compromised nodes and, consequently, better network connectivity and service availability. DQN-DeepNETAR-MD, with the highest $\mathcal{F}_C$ and the highest $\mathcal{P}_{MD}$, sacrifices security to achieve the best service availability. Conversely, DQN-DeepNETAR-SG has the lowest $\mathcal{F}_C$ and $\mathcal{P}_{MD}$ by focusing more on security over service availability. DQN-DeepNETAR-SG-MD balances these two schemes by achieving a relatively high security level while achieving fairly good service availability. As shown in Fig. 2, in conclusion, $\mathcal{S}_G$ is more closely related to $\mathcal{F}_C$ than $\mathcal{P}_{MD}$. Specifically, high $\mathcal{S}_G$ corresponds to low $\mathcal{F}_C$, but does not necessarily correspond to high $\mathcal{P}_{MD}$, representing service availability.

Fig. 3 shows how different upper bound levels of the total adaptation budget ($B$) influence the performance of each scheme in terms of the three metrics. As for DQN-




(a) Delivery of correct messages ($\mathcal{P}_{MD}$) (b) Size of the giant component ($\mathcal{S}_G$) (c) Fraction of compromised nodes ($\mathcal{F}_C$)

Fig. 2. Effect of varying the number of software packages available ($l$) under an ER network.




(a) Delivery of correct messages ($\mathcal{P}_{MD}$) (b) Size of the giant component ($\mathcal{S}_G$) (c) Fraction of compromised nodes ($\mathcal{F}_C$)

Fig. 3. Effect of varying the upper bound of the total adaptation budget ($B$) under an ER network.

DeepNETAR-MD and DQN-DeepNETAR-SG-MD, we observed overall that higher $B$ decreases $\mathcal{P}_{MD}$ and $\mathcal{F}_C$ since $\mathcal{F}_C$ is much more sensitive than $\mathcal{P}_{MD}$ with a relatively small $B$, which motivates the agent in each scheme to focus more on the security gain. However, once the optimal budget is identified, higher $B$ would slightly degrade the performance since higher $B$ corresponds to a larger search space, which makes it harder to identify the optimal adaptation budgets. In DQN-DeepNETAR-SG, its performance in $\mathcal{F}_C$ continues increasing under higher $B$. Note that larger $B$ does not necessarily increase $\mathcal{S}_G$ since all schemes perform network adaptations with their corresponding evaluation functions, and $\mathcal{S}_G$ is closely related to $\mathcal{F}_C$. Also notice that different schemes have different optimal $B$ with respect to $\mathcal{P}_{MD}$, $\mathcal{S}_G$ and $\mathcal{F}_C$. For example, the optimal $B$ of DQN-DeepNETAR-MD and DQN-DeepNETAR-SG-MD is near 250 while the optimal $B$ of DQN-DeepNETAR-SG, Greedy-SG, Optimal-SG, and Random is identified at about 350.

## VIII. Conclusion & Future Work

In this work, we proposed a Deep reinforcement learning (DRL)-based NETwork Adaptations for network Resilience algorithm, namely DeepNETAR, to generate robust network topologies against epidemic attacks but with sufficient connectivity available for seamless communications between nodes. In the proposed DeepNETAR, a DRL agent removes vulnerable edges or adds the least vulnerable edges to ensure system security and service availability. Unlike the existing approaches focusing on either objective, our work achieved the two conflicting goals. In particular, considering real-world communication networks, where packets may be dropped either from the presence of inside attackers or congestion on long routes, we developed a generic DRL-based model that can achieve different types of metrics to ensure service availability. It is well-known that the size of the giant component has not realized sufficient connectivities between nodes for seamless service provisions in communication networks. Hence, our work pioneered in providing a generic framework that can evaluate multiple metrics to ensure service availability while achieving a required level of system security. To attain these objectives, we considered vulnerability-based selection of adaptable edge candidates, fractal-based solution search, and diverse reward functions to achieve multi-objective optimization.

The **key findings** obtained from this study are:

- By setting an appropriate evaluation function, DQN-DeepNETAR-SG-MD can better ensure security, connectivity, and service availability simultaneously.
- As a network connectivity metric, the size of the giant component is more related to security rather than actual service availability under epidemic attacks.
- By setting different evaluation functions, our proposed DeepNETAR is a generic framework that can handle multiple, competing objectives.

For the **future work**, we plan to extend the proposed work by considering the following approaches:

- We will extend our current single agent DRL-based approach to a multi-agent DRL-based approach applicable in a large-scale network.

- We will extend the proposed approach to a network shuffling-based moving target defense (MTD) which requires a periodic triggering of network topologies. We will consider lightweight solutions by triggering the MTD as least as possible while ensuring a required level of system security.
- We will investigate how the proposed network adaptation approaches can help prevent or mitigate the impact introduced by other types of attacks and their benefit to the legacy defense mechanisms, such as intrusion detection or response systems.

## ACKNOWLEDGEMENT

This research is partly supported by the Army Research Office under Grant Contract Number W91NF-20-2-0140. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation.

## REFERENCES

[1] R. Albert, H. Jeong, and A. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, 2000.
[2] A.-L. Barabási, *Network Science*. Cambridge University Press, 2016.
[3] J. L. Cardy and P. Grassberger, "Epidemic models and percolation," *J. Physics A: Mathematical and General*, vol. 18, no. 6, p. L267, 1985.
[4] X. Chai, Y. Wang, C. Yan, Y. Zhao, W. Chen, and X. Wang, "DQ-MOTAG: Deep reinforcement learning-based moving target defense against DDoS attacks," in *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2020, pp. 375–379.
[5] J.-H. Cho, Y. Wang, I.-R. Chen, K. S. Chan, and A. Swami, "A survey on modeling and optimizing multi-objective systems," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1867–1901, 2017.
[6] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," *Proceedings of Machine Learning Research (PMLR)*, vol. 80, pp. 1115–1124, 2018.
[7] V.-A. Darvariu, S. Hailes, and M. Musolesi, "Improving the robustness of graphs through reinforcement learning and graph neural networks," *arXiv preprint arXiv:2001.11279*, 2020.
[8] Z. Dezsö and A.-L. Barabási, "Halting viruses in scale-free networks," *Physical Review E*, vol. 65, May 2002.
[9] P. Erdös and A. Rényi, "On the evolution of random graphs," in *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 1960, pp. 17–61.
[10] "Common Vulnerabilities and Exposures (CVE)," Forum of Incident Response and Security Teams. [Online]. Available: https://cve.mitre.org/
[11] G. Grimmett, "Percolation and disordered systems," in *Lectures on Probability and Statistics*. Springer, 1997, pp. 153–300.
[12] K. J. Hole, "Diversity reduces the impact of malware," *IEEE Security Privacy*, vol. 13, no. 3, pp. 48–54, May 2015.
[13] T. Kaur and J. Baek, "A strategic deployment and cluster-header selection for wireless sensor networks," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 4, pp. 1890–1897, 2009.
[14] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[15] D. Li, Q. Zhang, E. Zio, S. Havlin, and R. Kang, "Network reliability analysis based on percolation theory," *Reliability Engineering & System Safety*, vol. 142, pp. 556–562, 2015.
[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[17] C. Moore and M. E. Newman, "Epidemics and percolation in small-world networks," *Physical Review E*, vol. 61, no. 5, p. 5678, 2000.
[18] "Common Vulnerability Scoring System (CVSS)," National Vulnerability Database. [Online]. Available: https://www.first.org/cvss/
[19] M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.
[20] Y. Yang, S. Zhu, and G. Cao, "Improving sensor network immunity under worm attacks: A software diversity approach," *Ad Hoc Networks*, vol. 47, no. Supplement C, pp. 26–40, 2016.
[21] Q. Zhang, J.-H. Cho, T. J. Moore, and R. Chen, "Vulnerability-aware resilient networks: Software diversity-based network adaptation," *IEEE Transactions on Network and Service Management*, 2020.
[22] Q. Zhang, A. Z. Mohammed, Z. Wan, J.-H. Cho, and T. J. Moore, "Diversity-by-design for dependable and secure cyber-physical systems: A survey," 2020.
[23] T. Zhang, C. Xu, B. Zhang, X. Kuang, Y. Wang, S. Yang, and G.-M. Muntean, "DQ-RM: Deep reinforcement learning-based route mutation scheme for multimedia services," in *2020 IEEE Int'l Wireless Comms. and Mobile Computing (IWCMC)*, 2020, pp. 291–296.