Dynamic Dependency-Aware Vulnerability and Patch Management for Critical Interconnected Systems

Umar Sa'ad[a], Woongsoo Na[b], Nhu-Ngoc Dao[c,*], Sungrae Cho[a,*]

[a]*School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea*
[b]*Department of Software, Kongju National University, Cheonan 31080, South Korea*
[c]*Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea*

Abstract

Critical infrastructure systems characterized by complex interdependencies face significant challenges in vulnerability management due to cascading risk propagation through interconnected components. Traditional approaches that individually prioritize vulnerabilities inefficiently manage these dependency structures, leading to suboptimal security outcomes. This paper introduces an adaptive dependency-aware patching technique (ADAPT), a dynamic vulnerability and patch management framework that integrates formal dependency modeling with reinforcement learning to optimize patching strategies for critical interconnected systems. The proposed approach employs a mathematical formulation to capture direct and transitive dependencies via reachability matrices, enabling precise quantification of cascading risk propagation. The framework dynamically adapts patching decisions under resource constraints using proximal policy optimization within a constrained Markov decision process formulation. Comprehensive evaluation across 954 system configurations and six baseline strategies demonstrates consistent performance improvements, with 5.5% advantage over state-of-the-art NSGA-II multi-objective optimization while achieving 1,513× computational speedup. Optimality gap analysis reveals 4.33% average deviation from theoretical bounds, validating the framework's near-optimal solution quality. A critical infrastructure case study confirms practical applicability, with ADAPT achieving 89.7% risk reduction compared to 86.4% for sophisticated baseline methods, while enabling real-time decision-making through sub-second computation times. The results demonstrate superior performance under high dependency density and resource constraints, highlighting the framework's suitability for environments where cascading failures pose operational threats.

Keywords: Cascading risk mitigation, dependency-aware cybersecurity, vulnerability propagation, proximal policy optimization.

1. Introduction

From power grids and telecommunications networks to healthcare systems and transportation networks, critical infrastructure systems have become increasingly interconnected. This interconnectedness creates complex dependencies, complicating vulnerability management and patching decisions. Traditional approaches to vulnerability prioritization commonly treat each system component in isolation, focusing on individual vulnerability severity scores while neglecting the intricate dependencies characterizing the infrastructure [1, 2, 3]. This analytical gap causes suboptimal security configurations because vulnerabilities in low-priority components can propagate via dependency chains to influence critical services and operational capabilities [4].

Inadequate dependency modeling in vulnerability management has significant and far-reaching consequences. Recent security incidents have demonstrated how cascading failures propagate via interconnected systems, causing widespread disruptions exceeding the scope of the initial compromise. The Colonial Pipeline ransomware attack of 2021 [5] is a notable example. The attack initially targeted information technology systems but affected operational technology components, disrupting fuel delivery across multiple states and creating supply chain problems with national implications. Similarly, the SolarWinds supply chain attack [6] exploited software dependencies, compromising thousands of organizations via a single vulnerability in a widely used management platform. These incidents underscore the critical need for vulnerability management approaches that explicitly account for system dependencies and their role in risk propagation.

Despite recent advances in vulnerability scoring systems and prioritization methods, several research challenges remain. Current research on dependency modeling approaches has indicated opportunities for improvement in the formal mathematical representation of complex dependencies in operational environments [7, 8]. Existing methods have continued to evolve to balance risk reduction with resource constraints, particularly when dependencies create sequential requirements for patching operations [9]. Furthermore, current vulnerability management frameworks do not address the dynamic nature of operational

*Corresponding authors

Email addresses: `umar@uclab.re.kr` (Umar Sa'ad), `wsna@kongju.ac.kr` (Woongsoo Na), `nndao@sejong.ac.kr` (Nhu-Ngoc Dao), `srcho@cau.ac.kr` (Sungrae Cho)

systems, where dependencies and vulnerabilities continuously evolve in response to organizational requirements, technological changes, and emerging threats [10].

While significant research addresses individual aspects of vulnerability management, existing approaches exhibit fundamental limitations. Multicriteria optimization frameworks [11, 12] advance resource allocation but employ static prioritization that cannot adapt to evolving threat landscapes. Graph-theoretic methods [13, 14, 15] model dependency structures but lack optimization mechanisms to balance competing objectives under resource constraints. Recent reinforcement learning approaches [16, 12] demonstrate adaptive decision-making but inadequately model cascading dependencies and their role in risk propagation. Genetic algorithms such as NSGA-II [17, 18] provide multi-objective optimization but suffer from computational scalability limitations that prevent real-time deployment in large-scale systems. No existing framework simultaneously addresses: (1) formal mathematical representation of transitive dependencies via reachability analysis, (2) adaptive policy learning that improves through environmental interaction, and (3) computational efficiency enabling sub-second decision-making at operational scales.

This paper introduces the adaptive dependency-aware patching technique (ADAPT), a dynamic framework for vulnerability management and patching in critical interconnected systems. The proposed ADAPT framework integrates dependency graph modeling with reinforcement learning (RL) to optimize sequential patching decisions under operational constraints. System dependencies are represented as a directed graph, enabling the computation of reachability matrices that capture direct and transitive risk propagation pathways. The patching problem is formulated as a constrained Markov decision process with proximal policy optimization (PPO) for policy learning. The principal contributions of this work are as follows.

- A mathematical formulation for modeling direct and transitive dependencies in infrastructure systems that enables precise identification of cascading risk propagation pathways. The reachability matrix approach supports comprehensive impact analysis and informs dependency-aware patching sequences.

- A constraint-based reinforcement learning architecture that employs PPO to optimize patching strategies under dependency and resource constraints. The framework adapts to evolving system conditions through environmental interactions, improving effectiveness in complex dependency networks.

- Comprehensive evaluation against six baseline strategies spanning heuristic, graph-theoretic, and multi-objective optimization approaches. Results demonstrate consistent performance improvements with 5.5% advantage over state-of-the-art NSGA-II optimization, 4.33% average optimality gap relative to theoretical bounds, and 1,513× computational speedup enabling real-time deployment in critical infrastructure environments.

The remainder of this paper is organized as follows. Section 2 reviews the related work on dependency modeling, risk quantification, and optimization approaches for vulnerability management. Next, Section 3 presents the proposed system model and formal problem formulation, including the dependency representation and risk quantification. Then, Section 4 details the dependency-aware RL framework, covering the state representation, action space, reward function, transition dynamics, and policy optimization. Further, Section 5 discusses the algorithm design (including the selection of PPO as the deep RL algorithm), training process, and policy representation. Moreover, Section 6 describes the experimental evaluation (including the implementation of baseline patching strategies for the comparative analysis), comprehensive experimental setup, results and analysis, and robustness analysis. Section 7 provides a case study on a critical infrastructure scenario. Section 8 presents discussion and limitations of our work. Finally, Section 9 concludes the paper by summarizing the findings, contributions, and future research.

2. Related Work

Vulnerability and patch management in interconnected critical infrastructure systems requires sophisticated approaches that consider complex interdependencies and resource constraints. Although significant research has addressed individual aspects of this challenge, the existing approaches have fundamental limitations preventing optimal security in critical systems. This section investigates the evolution of vulnerability management approaches and identifies the critical gaps that motivate the dependency-aware RL framework.

2.1. Dependency Modeling in the Critical Infrastructure

Traditional vulnerability management frameworks manage system components in isolation, focusing on individual vulnerabilities without considering their interconnectedness [19, 20]. This methodological limitation causes incomplete risk assessments and suboptimal patching strategies, particularly when considering cascading effects via interconnected components. Researchers [21] have highlighted the limitations of traditional assessments that identify critical components without considering their interconnections, advocating for holistic approaches that assess vulnerability measures across entire networks. Similarly, one study [22] introduced "vulnerability clouds" to describe vulnerability distributions across critical systems, enabling a better comprehension of the topology and controllability of systems under disruptive scenarios.

The challenge of accurately representing direct and indirect dependencies remains substantial. A study [23] noted that the current methods often neglect indirect dependencies arising from interconnected networks, resulting in patching strategies that address immediate vulnerabilities but fail to mitigate broader systemic risks [7]. Advanced approaches have emerged, including the multiattribute vulnerability criticality analysis model [9], which estimates the influence of the vulnerability by considering its severity, attack probability, and functional dependencies. This quantitative approach permits prioritization based on systemic influence rather than isolated severity metrics; however, this approach remains primarily static and fails to adapt to evolving configurations.

2.2. Risk Propagation and Cascading Effects

Understanding risk propagation throughout interdependent systems is crucial for effective vulnerability management. Multiple studies have demonstrated how interdependencies between critical infrastructures cause cascading failures, where disruptions induce subsequent failures in connected systems [24, 25]. Mathematical modeling approaches have been developed to define these cascading effects [26, 27], with simplified models revealing the critical characteristics of interconnected systems and the correlation between component outages in coupled systems.

Quantitative risk assessment methods have emerged using graph centrality measures alongside dependency risk graphs to gauge critical infrastructures [8], identifying the nodes most at risk due to interdependencies. Complementary work [28] has demonstrated the effectiveness of quantitative models in assessing infrastructure interdependencies and enhancing disaster resilience, enabling the quantification of component functionality dependencies and vulnerability propagation pathways. However, these approaches apply static analyses that fail to capture the dynamic nature of system configurations and the evolving threat landscapes.

2.3. Optimization Approaches and Resource Allocation

Resource-constrained optimization for patch management signifies a critical challenge given the rapidly evolving cybersecurity threats and limited resources. Multicriteria optimization approaches have emerged to address this problem [1, 29], moving beyond single-metric evaluations in normalization frameworks that consider multiple risk factors and system relationship modeling for predictive patch deployment. These approaches analyze connections and dependencies between software components to determine patching decision influences, identifying critical paths that maximize the limited resource utilization.

However, the current optimization approaches encounter considerable scalability challenges in large-scale interconnected systems [30]. As the network complexity and size increase, existing algorithms have become computationally infeasible, limiting their practical application. These approaches normally employ deterministic, one-time decision processes that inadequately consider future uncertainties and vulnerability dynamics.

Beyond traditional optimization for patch management, the application of artificial intelligence to broader cybersecurity challenges provides relevant context for ADAPT's approach. Network-level anomaly detection using pattern-based random walks [31] has demonstrated 95% detection rates through dynamic behavioral graph analysis, addressing threat identification that complements proactive vulnerability remediation. More broadly, comprehensive frameworks for AI-enhanced security [32] position machine learning approaches within the evolving cybersecurity ecosystem, where adaptive solutions address sophisticated threats across multiple dimensions from detection to response. ADAPT extends this AI-for-security paradigm specifically to vulnerability and patch management, where dependency-aware reinforcement learning enables discovery of sophisticated prioritization strategies under operational constraints.

2.4. Critical Research Gaps

Despite the advances in vulnerability management frameworks, three fundamental limitations necessitate this research. First, the current approaches lack dynamic adaptability, employing static prioritization mechanisms that fail to consider evolving threat landscapes and changing system configurations [33, 34]. Traditional frameworks, such as NIST CSF, identify vulnerabilities but fail to provide actionable prioritization guidance accounting for organizational context and resource constraints, and the common vulnerability scoring system (CVSS) inadequately captures the dynamic threat evolution or operational context of critical infrastructure environments.

Second, existing methods poorly integrate dependency relationships with optimization decisions, addressing dependency modeling and resource allocation as separate problems rather than unified challenges. Many frameworks emphasize theoretical vulnerability severity rather than the likelihood of empirical exploitation, resulting in misaligned resource allocation where actively weaponized vulnerabilities receive insufficient attention [34].

Third, the current approaches display limited learning capabilities and insufficient scalability characteristics [35, 12]. The evaluation processes remain predominantly manual and resource-intensive, preventing prioritization based on specific threat actor tactics and procedures. Although frameworks incorporating RL and game theory have been proposed [12, 36], these approaches have not achieved widespread operational adoption. The practical effect is evident: fewer than 30% of industrial control system devices achieve patched status within 60 days of a vulnerability disclosure [30].

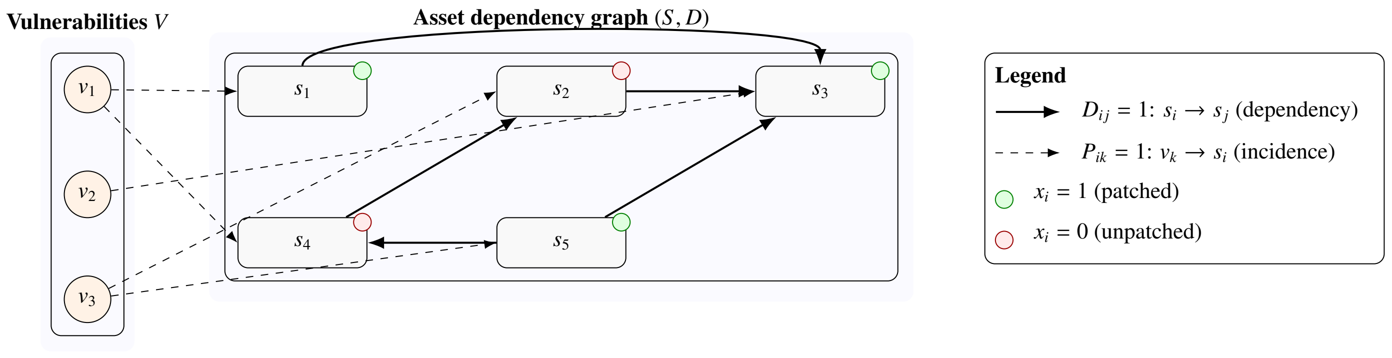
The proposed dynamic dependency-aware vulnerability and patch management framework addresses these gaps by integrating comprehensive dependency modeling with adaptive RL techniques. This approach enables dynamic optimization that continuously improves patch management decisions in complex, resource-constrained environments.

3. System Model and Problem Formulation

This section establishes the system model and problem formulation for dependency-aware vulnerability and patch management via graph-theoretic constructs that systematically capture critical infrastructure interdependencies and vulnerability propagation mechanisms.

3.1. System Representation

We model the system (see Figure 1) as n *assets* $S = \{s_1, \ldots, s_n\}$ (devices, applications, libraries, or services). Directed dependencies are encoded by a binary matrix $D \in \{0, 1\}^{n \times n}$, where $D_{ij} = 1$ iff asset s_i depends on asset s_j (edge



Vulnerabilities V
Asset dependency graph (S, D)
v_1
v_2
v_3
s_1
s_2
s_3
s_4
s_5
Legend
$D_{ij} = 1$: $s_i \to s_j$ (dependency)
$P_{ik} = 1$: $v_k \to s_i$ (incidence)
$x_i = 1$ (patched)
$x_i = 0$ (unpatched)

Figure 1: System representation: assets S with directed dependencies D, vulnerabilities V with incidence P, and per-asset patch state x. Solid arrows encode D_{ij}=1 ($s_i \to s_j$), dashed arrows encode P_{ik}=1 (v_k affects s_i), and circular badges indicate x_i (patched vs. unpatched).

$s_i \to s_j$). This captures the immediate operational requirements among assets (e.g., a service depending on a database) and establishes the graph on which risk can propagate.

Known vulnerabilities are $V = \{v_1, \ldots, v_m\}$. Their incidence on assets is given by the binary matrix $P \in \{0, 1\}^{n \times m}$, with $P_{ik} = 1$ iff vulnerability v_k affects asset s_i. Patch state (or decision) is represented by $x \in \{0, 1\}^n$, where $x_i = 1$ denotes that s_i is patched and $x_i = 0$ otherwise. Together, (S, D, P, x) provide a structured system state used for risk quantification and optimization.

3.2. Dependency Propagation

Dependencies are not limited to direct links; effects can cascade along paths. We therefore use the *reachability* (transitive-closure) matrix R derived from D:

$$R = (I + D + D^2 + \cdots + D^{n-1}) > 0, \tag{1}$$

where I is the identity and the > 0 operator Booleanizes entries (positive $\mapsto$ 1, otherwise 0). Thus, $R_{ij} = 1$ iff there exists a directed path from s_i to s_j, i.e., s_i (directly or indirectly) depends on s_j. This closure exposes potential cascade pathways: a vulnerability present on an upstream asset s_j can influence all downstream assets s_i with $R_{ij} = 1$. Using R enables identification of structurally critical assets and prioritization of patches that mitigate the largest downstream risk.

3.3. Risk Model

We formulate a quantitative risk model that captures the instantaneous vulnerability exposure of individual assets within the system. The per-asset risk is characterized by the following expression:

$$\rho_i(t) = v_i \, p_i(t) \, (1 - x_i(t)),$$

where $v_i > 0$ represents the business criticality and potential impact severity associated with asset i, $p_i(t) \in [0, 1]$ denotes the time-dependent exploit probability conditioned on current exposure levels and contextual threat intelligence, and $x_i(t) \in \{0, 1\}$ indicates the binary patch status with $x_i(t) = 1$ signifying successful remediation.

The aggregate system risk is computed as the sum of individual asset risks:

$$\Upsilon_t = \sum_{i=1}^{n} \rho_i(t).$$

The risk reduction achieved at time step t is quantified as $\Delta\Upsilon_t \triangleq \Upsilon_t - \Upsilon_{t+1} \geq 0$, representing the decrease in total system vulnerability following remediation actions.

This risk formulation establishes the foundation for the subsequent optimization framework, wherein we define the control objectives and operational constraints governing admissible patch deployment strategies.

Estimating exploit probability. We set $p_i(t) = \sigma(\alpha_0 + \alpha_1 \mathrm{CVSS}_i + \alpha_2 \mathrm{EPSS}_i(t) + \alpha_3 \mathbf{1}\{\mathrm{KEV}_i\} + \alpha_4 \Delta t_i)$, where Δt_i is time since disclosure. Sensitivity varies (α_2, α_3).

3.4. Optimization Problem

The patch deployment problem is formulated as a constrained Markov decision process wherein we seek an optimal policy π_θ that maximizes the expected discounted return:

$$\begin{aligned} \max_{\theta} \quad & J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \gamma^t R_t\right] \\ \text{s.t.} \quad & a_t \in \{0, 1\}^n, \\ & x_{t+1} = x_t \vee a_t, \\ & a_t \odot (1 - M_t) = \mathbf{0}. \end{aligned} \tag{2}$$

The binary action constraint ensures discrete patch decisions, the state evolution equation captures cumulative patching effects, and the feasibility mask M_t enforces dependency and redundancy constraints. The reward function R_t incorporates risk reduction, efficiency metrics, and operational penalties as defined in Sec. 4.3, while $\gamma \in (0, 1]$ provides temporal discounting.

Budget constraints may be imposed either through hard limits,

$$\sum_{i=1}^{n} c_i a_{t,i} \leq \bar{b} \quad \forall t,$$

or incorporated softly via penalty terms within the reward structure.

One-step MILP surrogate (baseline). To establish a benchmark, we formulate a myopic mixed-integer linear program that optimizes immediate risk reduction from state $(x_t, p(t))$. Define per-asset weights $w_i(t) \triangleq v_i\, p_i(t)\,(1 - x_i(t))$ and prerequisite sets $\mathcal{P}_i \triangleq \{j : D_{ij} = 1\}$. The optimization problem becomes:

$$\max_{a\in\{0,1\}^n} \quad \sum_{i=1}^{n} w_i(t)\, a_i \;-\; \lambda_0 \sum_{i=1}^{n} a_i \tag{3a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} c_i a_i \;\le\; \bar{b}, \tag{3b}$$

$$a_i \;\le\; x_j(t) + a_j \qquad \forall i,\ \forall j \in \mathcal{P}_i. \tag{3c}$$

Constraint (3c) ensures that asset i can only be patched if all prerequisites $j \in \mathcal{P}_i$ are either previously patched or simultaneously selected. This formulation provides a near-optimal myopic baseline for evaluating the reinforcement learning policy.

4. Adaptive Dependency-Aware Patching Technique

This section presents the formal structure of the ADAPT framework. We frame the problem as a Markov decision process, where states encapsulate the system security, vulnerability metrics, and dependency relationships.

The reward design balances the immediate risk reduction with transitive security improvements while enforcing operational constraints.

4.1. State, Actions, and Dynamics

We define the system structure through a dependency matrix (Sec. 3.1), where $D_{ij} = 1$ indicates that asset j constitutes a prerequisite for patching asset i. The operational cost vector $c \in \mathbb{R}^n_{++}$ quantifies resource requirements, representing factors such as system downtime or labor allocation for each asset.

State. The environment state s_t comprises concatenated node-level features and graph-level summary statistics:

$$s_t \;=\; [x(t),\, v,\, c,\, f^{\text{node}}(D),\, g^{\text{graph}}(D)],$$

where $x(t) = [x_1(t), \ldots, x_n(t)]^\top$ represents the current patch status vector. The function $f^{\text{node}}(D)$ extracts per-asset topological features including dependency counts, node degrees, and historical failure indicators, while $g^{\text{graph}}(D)$ computes global network properties such as mean connectivity, clustering coefficients, and degree assortativity. This representation maintains fixed dimensionality while capturing essential structural information.

Action. The agent's decision at each time step consists of a binary action vector $a_t \in \{0, 1\}^n$, where $a_{t,i} = 1$ signifies that asset i is selected for patch deployment during step t. This formulation permits simultaneous multi-asset patching operations.

Dynamics. The system evolves according to deterministic state transition dynamics. The patch status updates through element-wise logical disjunction:

$$x_{t+1} \;=\; x_t \vee a_t,$$

ensuring irreversible patch deployment. While the dependency structure remains static, the exploit probabilities $p_i(t)$ may exhibit temporal evolution through exogenous threat intelligence updates or endogenous decay following successful remediation actions.

4.2. Dependency-Aware Masking

We enforce feasibility with a binary mask $M_t \in \{0, 1\}^n$ that disables actions which are already completed or violate prerequisites:

$$M_{t,i} \;=\; \begin{cases} 0, & \text{if } x_i(t) = 1 \text{ (already patched)},\\ 0, & \text{if } \exists\, j : \ D_{ij} = 1 \ \wedge \ x_j(t) = 0 \text{ (unmet prerequisite)},\\ 1, & \text{otherwise.} \end{cases}$$

To apply the mask inside the policy, we use a numerically stable log-mask on the logits $z_t \in \mathbb{R}^n$:

$$z'_t \;=\; z_t + \log(M_t + \varepsilon), \qquad \varepsilon \approx 10^{-45},$$

so infeasible coordinates receive $\approx -\infty$ before the sigmoid. The policy factorizes as independent *masked Bernoulli* terms (no cross-dimensional normalization):

$$\pi_\theta(a_t \mid s_t, M_t) \;=\; \prod_{i=1}^{n} \sigma(z'_{t,i})^{a_{t,i}} \big(1 - \sigma(z'_{t,i})\big)^{1-a_{t,i}}, \qquad a_{t,i} \in \{0, 1\}.$$

This concentrates probability mass on feasible actions and blocks gradients through infeasible ones, while retaining differentiability on valid dimensions. (Alternative multi-label parameterizations are possible but not used here.)

4.3. Reward Design

We balance *risk reduction* and *operational efficiency* while enforcing hard constraints (dependencies, per-window budget, and capacity). Let $P_i = \{\, j : \ D_{ij} = 1 \,\}$ denote prerequisites for asset i.

$$G_t \triangleq \sum_{i=1}^{n} c_i\, a_{t,i}, \tag{4a}$$

$$\begin{aligned} \text{viol}_t \triangleq\ & \mathbf{1}\{G_t > \bar{b}\} + \mathbf{1}\{\|a_t\|_0 > k_{\max}\} \\ & + \mathbf{1}\{\exists\, i : \ a_{t,i} = 1 \wedge \exists\, j \in P_i : \ x_{t,j} = 0\}, \end{aligned} \tag{4b}$$

$$R_t \triangleq \begin{cases} w\, \dfrac{\Delta\Upsilon_t}{\Upsilon_0} + (1 - w)\, \dfrac{\Delta\Upsilon_t}{G_t + \varepsilon}, & \text{viol}_t = 0,\\ -\kappa\, \text{viol}_t, & \text{viol}_t > 0, \end{cases} \tag{4c}$$

where ε is a small constant for numerical stability and $\kappa > 0$ is a penalty weight. Feasibility thus requires: (i) dependency satisfaction ($a_{t,i} = 1 \Rightarrow x_{t,j} = 1$ for all $j \in P_i$), (ii) budget $G_t \le \bar{b}$, and (iii) sparsity $\|a_t\|_0 \le k_{\max}$. These same constraints are enforced in the policy via the mask M_t (Sec. 4.2).

The trade-off parameter $w \in [0, 1]$ captures organizational risk tolerance. We use $w = 0.7$ to prioritize risk reduction while remaining cost-aware; sensitivity analysis shows stable behavior for $w \in [0.6, 0.8]$. The term $\Delta\Upsilon_t/\Upsilon_0$ provides scale-invariant improvement, whereas $\Delta\Upsilon_t/(G_t+\varepsilon)$ measures mitigation *per unit spend.* Constraint violations incur a negative reward proportional to viol_t, nudging the policy toward feasible actions.

5. Algorithm Design

This section presents a systematic framework for selecting the RL algorithm in ADAPT. We established formal criteria for evaluating these algorithmic approaches based on their capacity to address high-dimensional state-action spaces, complex constraint satisfaction, and computational efficiency in interconnected systems.

5.1. Deep Reinforcement Learning Algorithm Selection

The selection of an appropriate algorithm is essential to address the challenges of dependency-aware vulnerability and patch management in dynamic and complex environments. The chosen algorithm must efficiently handle high-dimensional state and action spaces, complex constraint satisfaction, and scalability requirements in large interconnected systems. Considering the specific problem characteristics, this work prioritizes algorithms that balance computational efficiency, learning stability, and policy adaptability.

For this framework, PPO [37] was selected as the primary deep RL algorithm. The PPO algorithm is an actor-critic method that employs a clipped surrogate objective function to limit policy updates while learning a value function. This approach offers several advantages in vulnerability management: (1) sample efficiency (which is crucial when complex system interaction simulations are computationally expensive), (2) stable learning with controlled policy updates (crucial for avoiding catastrophic forgetting when vulnerability landscapes evolve), and (3) effective constraint handling via auxiliary loss functions (essential for respecting dependency ordering and resource limitations).

Another candidate algorithm for consideration is the deep Q-network (DQN) [38], a value-based approach that estimates the optimal action-value function $Q(s, a)$ using a neural network. The DQN approach is advantageous in environments with discrete action spaces, making it suitable for scenarios where the RL agent determines whether to patch individual elements or predefined clusters. However, the DQN method may encounter challenges in scalability when managing large action spaces or dynamic environments because it requires a separate evaluation of each potential action.

For problems involving complex decision spaces, other actor-critic variants, including advantage actor-critic [39] or soft actor-critic [40] methods could also be considered. These methods (e.g., PPO) maintain the policy and value function components but differ in their update mechanisms and exploration strategies. For instance, the soft actor-critic network incorporates entropy regularization to encourage exploration, which can be beneficial in environments with multiple viable patching solutions.

Considering the specific requirements of the ADAPT framework, PPO offers a compelling balance of sample efficiency, stability, and performance regarding actor-critic methods [41]. The controlled policy updates of PPO prevent drastic strategy changes that could disrupt critical system operations. In addition, the ability of PPO to incorporate multiple objectives and constraints via auxiliary loss terms makes it suitable for simultaneously balancing risk reduction, resource utilization, and dependency satisfaction. These characteristics ensure reliable performance even in scenarios where dependencies, vulnerabilities, and budgets evolve.

Algorithm 1 ADAPT (Training with Masked-Bernoulli Policy and PPO)

```
Require: Dependency matrix $D$, severities $v$, costs $c$, capacity $k_{\max}$, budget cap $\bar{b}$, risk tradeoff $w$, policy $\pi_\theta$, value $V_\phi$, discount $\gamma$, GAE $\lambda$, PPO clip $\epsilon$, lrs $\alpha_\theta, \alpha_\phi$, numeric $\varepsilon$ (log-mask)
 1: /* Initialization */
 2: Initialize policy parameters $\theta$ and value parameters $\phi$
 3: for each iteration do
 4:     Initialize empty buffer $\mathcal{D}$
 5:     for each episode do
 6:         Reset: $x_0 \leftarrow \mathbf{0}^n$; observe initial state $s_0$
 7:         while episode not terminated do
 8:             /* Feasibility mask (structural/window constraints) */
 9:             $M_t \leftarrow \textsc{GetMask}(x_t, D)$ {$M_t \in \{0,1\}^n$}
10:             /* Masked-Bernoulli policy */
11:             $z_t \leftarrow f_\theta(s_t, M_t)$ {policy logits}
12:             $z'_t \leftarrow z_t + \log(M_t + \varepsilon)$ {log-mask, infeasible $\rightarrow -\infty$}
13:             Sample multi-asset action: $a_t \sim \prod_{i=1}^n \text{Bernoulli}(\sigma(z'_{t,i}))$
14:             /* Environment step */
15:             $x_{t+1} \leftarrow x_t \vee a_t$ {element-wise OR}
16:             $G_t \triangleq \sum_{i=1}^n c_i a_{t,i}$ {per-step expenditure}
17:             Compute $\Upsilon_t, \Upsilon_{t+1}$ and $\Delta\Upsilon_t \triangleq \Upsilon_t - \Upsilon_{t+1}$ {risk model}
18:             /* Constraints & reward */
19:             $\text{viol}_t \triangleq \mathbf{1}\{G_t > \bar{b}\} + \mathbf{1}\{\|a_t\|_0 > k_{\max}\}$
20:             if $\text{viol}_t = 0$ then
21:                 $r_t \triangleq w\,\frac{\Delta\Upsilon_t}{\Upsilon_0} + (1-w)\,\frac{\Delta\Upsilon_t}{G_t+\varepsilon}$
22:             else
23:                 $r_t \triangleq -\kappa\,\text{viol}_t$ {penalty weight $\kappa > 0$}
24:             end if
25:             Observe $s_{t+1}$; store $(s_t, a_t, r_t, s_{t+1}, M_t, z'_t)$ in $\mathcal{D}$
26:             if $\sum_i x_{t+1,i} = n$ or $\max(M_{t+1}) = 0$ then
27:                 Terminate episode
28:             end if
29:         end while
30:     end for
31:     /* Advantage estimation (GAE) */
32:     for each trajectory in $\mathcal{D}$ (reverse time) do
33:         $\delta_t \leftarrow r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
34:         $\hat{A}_t \leftarrow \delta_t + \gamma\lambda\,\hat{A}_{t+1}$ {with $\hat{A}_T = 0$}
35:         $\hat{V}_t \leftarrow \hat{A}_t + V_\phi(s_t)$
36:     end for
37:     /* PPO updates */
38:     for each epoch do
39:         for each minibatch $\mathcal{B} \subset \mathcal{D}$ do
40:             $r_t(\theta) \leftarrow \dfrac{\pi_\theta(a_t \mid s_t, M_t)}{\pi_{\text{old}}(a_t \mid s_t, M_t)}$ {masked policy ratio}
41:             $L^{\text{CLIP}}(\theta) \leftarrow \mathbb{E}_{\mathcal{B}}\Big[\min\Big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big]$
42:             $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta L^{\text{CLIP}}(\theta)$
43:             $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi (V_\phi(s_t) - \hat{V}_t)^2$
44:         end for
45:     end for
46: end for
47: return Trained dependency-aware policy $\pi_\theta$
```

5.2. Training Process and Policy Architecture

The policy network π_θ is a feedforward MLP with two hidden layers (128 units each, ReLU) and dropout ($p = 0.1$). The input dimension is $|s_t| + n$, concatenating the state features with the feasibility mask $M_t \in \{0,1\}^n$. The policy head emits n logits $z_t \in \mathbb{R}^n$ which are *masked and passed through an elementwise sigmoid.* Concretely, we apply a stable log-mask,

$$z'_t \;=\; z_t + \log(M_t + \varepsilon), \qquad \varepsilon \approx 10^{-45},$$

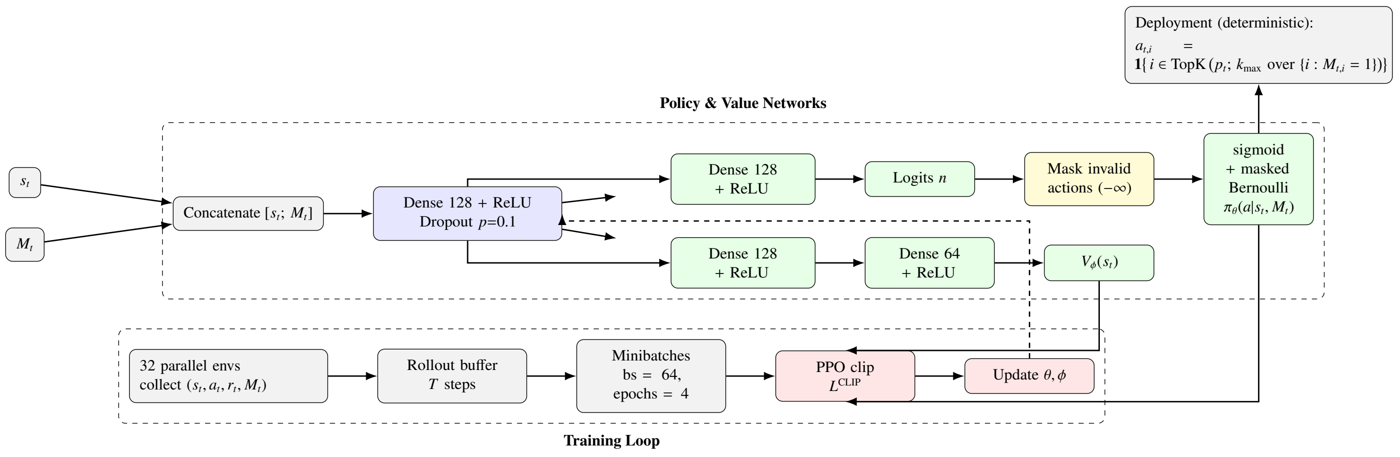

Deployment (deterministic):
$a_{t,i} = \mathbf{1}\{ i \in \mathrm{TopK}(p_t;\, k_{\max} \text{ over } \{i : M_{t,i} = 1\})\}$
Policy & Value Networks
s_t
M_t
Concatenate $[s_t; M_t]$
Dense 128 + ReLU Dropout p=0.1
Dense 128 + ReLU
Logits n
Mask invalid actions ($-\infty$)
sigmoid + masked Bernoulli $\pi_\theta(a|s_t, M_t)$
Dense 128 + ReLU
Dense 64 + ReLU
$V_\phi(s_t)$
32 parallel envs collect (s_t, a_t, r_t, M_t)
Rollout buffer T steps
Minibatches bs = 64, epochs = 4
PPO clip L^{CLIP}
Update θ, ϕ
Training Loop

Figure 2: Policy/value architecture with dependency-aware action masking, PPO training loop, and deterministic deployment. Invalid actions are masked by assigning $-\infty$ to their logits before sigmoid + masked Bernoulli, ensuring $\pi_\theta(a|s_t, M_t)$ has support only on valid actions.

so infeasible entries receive $\approx -\infty$ before the sigmoid. Actions factorize as independent *masked Bernoulli* variables:

$$\pi_\theta(a_t \mid s_t, M_t) \;=\; \prod_{i=1}^{n} \sigma(z'_{t,i})^{a_{t,i}} (1 - \sigma(z'_{t,i}))^{1-a_{t,i}}, \quad a_{t,i} \in \{0, 1\}.$$

The value network V_ϕ shares the input embedding and branches into two hidden layers (128, 64) terminating in a scalar.

Training follows Algorithm 1 with hyperparameters in Table 1. Each iteration gathers rollouts from 32 parallel environments, then performs 4 epochs of minibatch PPO updates with batch size 64. The clipped objective (with masking inside the policy) is

$$\begin{aligned} L^{\text{CLIP}}(\theta) &= \mathbb{E}_{\mathcal{B}}\Big[\min(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_t)\Big], \\ r_t(\theta) &= \frac{\pi_\theta(a_t \mid s_t, M_t)}{\pi_{\text{old}}(a_t \mid s_t, M_t)}. \end{aligned} \tag{5}$$

Because infeasible dimensions are driven to near-zero probability via z'_t, gradients flow only through valid actions while retaining differentiability for feasible entries.

Convergence and early stopping. We monitor a sliding window of 100 episodes and stop when all hold: (i) average risk reduction improves by $<1\%$ over the previous 200 episodes (plateau), (ii) constraint violations $<0.1\%$ for 100 consecutive episodes, and (iii) value loss $< \tau$ with $\tau = 0.01$. This guards against overfitting to the training distribution.

Deployment (deterministic decoding, Top-k). At inference we form masked probabilities $p_t = \sigma(z'_t) \odot M_t$ and select the $k_{\max}$ largest feasible entries (if fewer than $k_{\max}$ are feasible, we select all):

$$a_{t,i} = \mathbf{1}\{\, i \in \mathrm{TopK}\,(p_t;\, k_{\max} \text{ over } \{i : M_{t,i} = 1\})\}\,.$$

This Top-k decoder mirrors the per-window capacity used in evaluation and in our myopic MILP surrogate, yields $O(n \log n)$ complexity (sorting), and requires only a forward pass through the policy. See Figure 2 for an architectural schematic.

Table 1: ADAPT Hyperparameter Configuration

Parameter	Value
Policy network architecture	[128, 128]
Value network architecture	[128, 64]
Learning rate	3×10^{-4}
Discount factor (γ)	0.99
GAE parameter (λ)	0.95
Clipping parameter (ϵ)	0.2
Value function coefficient	0.5
Entropy coefficient	0.01
Training episodes	1,000

6. Experimental Evaluation

6.1. Baseline Patching Strategies

This section presents six baseline approaches used to benchmark ADAPT across practical deployment scenarios. Table 2 summarizes the methods spanning *heuristic*, *graph-theoretic*, and *optimization* families.

Heuristic baselines — *Random* and *Severity (CVSS/EPSS)*: provide fundamental bounds, where *Random* samples uniformly from the feasible action set, while *Severity* selects the highest per-asset risk score (e.g., CVSS- or EPSS-derived), ignoring topology.

Graph-theoretic baselines — *In-Degree (Popularity)*, *PageRank*, and *Betweenness*: leverage the directed dependency graph. *In-Degree* targets assets most depended upon (high incoming degree). *PageRank* uses a damping factor $d = 0.85$ and iterates to convergence:

$$\mathrm{PR}(s_i) = \frac{1-d}{n} + d \sum_{s_j \in M(s_i)} \frac{\mathrm{PR}(s_j)}{L(s_j)}, \tag{6}$$

where n is the number of assets, $M(s_i)$ are predecessors (assets that depend on s_i), and $L(s_j)$ is the out-degree of s_j. *Betweenness* centrality prioritizes bridge nodes that lie on many shortest paths. For efficiency, all centrality scores are precomputed once per run and cached; per-step selection is then made from these cached rankings subject to feasibility.

Optimization baseline — *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*: conducts a multi-objective search over risk reduction, cost, and operational constraints. The configuration uses population size $N = 100$, $G = 200$ generations, and crossover probability $p_c = 0.9$. For single-point comparisons against other methods, a representative solution is chosen from the Pareto set using TOPSIS.

Together, these six baselines cover a spectrum from simple score-based heuristics to structure-aware graph analytics and scalable multi-objective optimization.

6.1.1. Baseline Implementation Details

To ensure fair comparison and reproducibility, we provide detailed experimental configurations for each baseline strategy:

Random Baseline: At each decision step, the random baseline samples uniformly from the set of feasible actions $\mathcal{A}_t = \{a \in \{0,1\}^n : M_t \odot a = a\}$, where M_t is the feasibility mask enforcing dependency and budget constraints. This provides a lower-bound performance metric representing decisions made without any strategic consideration.

Severity-Based (CVSS/EPSS): Vulnerability severity is computed as $s_i = \alpha_1 \text{CVSS}_i + \alpha_2 \text{EPSS}_i(t)$, using the same weighting coefficients as ADAPT's risk model to ensure consistent severity assessment. At each step, the strategy selects the $k_{\max}$ feasible assets with highest severity scores. Severity scores are precomputed once per configuration and remain static throughout the episode. When multiple assets have identical severity scores, selection priority is determined by asset index (lower indices selected first) for deterministic reproducibility.

Popularity (In-Degree): The in-degree $\deg^-(s_i) = \sum_{j=1}^{n} D_{ji}$ represents the number of assets that depend on asset i. Assets with higher in-degree protect more downstream dependencies. In-degree values are computed once from the dependency matrix D at the start of each configuration and cached for O(1) lookup during selection. This baseline represents the intuition that patching highly-depended-upon components provides maximum cascading protection.

PageRank: We implement PageRank with damping factor $d = 0.85$ and convergence tolerance $\varepsilon = 10^{-6}$, following standard recommendations [42]. The algorithm iterates Equation 6 until $\|\text{PR}^{(t+1)} - \text{PR}^{(t)}\|_1 < \varepsilon$, typically converging within 20-30 iterations for our network sizes. PageRank scores are precomputed once per configuration. This captures global importance through the entire dependency graph rather than just local connectivity.

Betweenness Centrality: We compute betweenness using Brandes' algorithm [43] with complexity $O(n|E|)$ for unweighted graphs. For each asset i, betweenness $BC(i) = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}}$ quantifies how many shortest paths pass through i, identifying critical bridge nodes. Betweenness scores are precomputed and cached. Assets with high betweenness act as bottlenecks whose compromise could disconnect parts of the dependency graph.

NSGA-II Configuration: We configure NSGA-II with population size $N = 100$, maximum generations $G = 200$, simulated binary crossover with probability $p_c = 0.9$ and distribution index $\eta_c = 20$, and polynomial mutation with probability $p_m = 1/n$ and distribution index $\eta_m = 20$. The fitness evaluation considers two objectives: (1) total risk reduction $\Delta\Upsilon$, and (2) resource efficiency $\Delta\Upsilon / \sum_i c_i a_i$. For comparison against single-objective methods, we select a representative solution from the final Pareto front using TOPSIS [44] with equal weights (0.5, 0.5) for both objectives, representing balanced risk-cost trade-offs. For large-scale systems ($n \geq 200$), computational constraints limit NSGA-II to $G = 50$ generations, which is noted in results where applicable.

Constraint Enforcement Consistency: All baselines operate under identical constraints at each decision step:

- *Dependency constraints*: Asset i can only be selected if all prerequisites $j \in \mathcal{P}_i = \{j : D_{ij} = 1\}$ are either already patched or simultaneously selected
- *Budget constraints*: The total cost $\sum_i c_i a_{t,i} \leq \bar{b}$ must not exceed the per-step budget
- *Cardinality constraints*: The number of simultaneously selected assets $\|a_t\|_0 \leq k_{\max}$ respects capacity limits
- *Irreversibility*: Once patched ($x_i = 1$), assets remain patched; the action space at step t excludes already-patched assets

These constraints are enforced through the feasibility mask M_t for all methods. For NSGA-II, infeasible solutions are penalized heavily in fitness evaluation rather than using repair operators, ensuring evolutionary pressure toward feasible regions.

6.2. Experimental Setup

The experimental setup integrates a multifaceted approach designed to ensure a comprehensive evaluation across heterogeneous system configurations. We implemented a controlled experimental environment with standardized initial conditions to facilitate a comparative analysis of the algorithmic performance.

6.2.1. System Configuration Parameterization

We developed a parametric system generation framework to gauge performance across a controlled spectrum of configurations, assessing five primary system scales: small-scale systems ($n = 20$ elements and $m = 10$ vulnerabilities), medium-scale systems ($n = 50$ elements and $m = 25$ vulnerabilities), large-scale systems ($n = 100$ elements and $m = 40$ vulnerabilities), enterprise-scale systems ($n = 200$ elements and $m = 80$ vulnerabilities), and critical infrastructure-scale systems ($n = 500$ elements and $m = 150$ vulnerabilities). For scales $n \leq 100$, we systematically varied the critical parameters, including the dependency density ($\delta : 0.05, 0.1, 0.2, 0.4, 0.6$), vulnerability distribution heterogeneity ($\eta : 0.1, 0.3, 0.5, 0.7, 0.9$), budget constraints ($\beta : 0.2, 0.4, 0.6, 0.8\times$ total patching cost) and critical element ratio ($\gamma : 0.1, 0.2, 0.3$). This parameterization yielded 900 distinct system configurations for comprehensive performance assessment.

For larger scales ($n \geq 200$), computational constraints necessitated focused evaluation on key parameters: dependency density ($\delta : 0.1, 0.2, 0.4$), budget constraints ($\beta : 0.2, 0.4, 0.6$), and fixed heterogeneity ($\eta = 0.5$) and critical ratio ($\gamma = 0.2$), generating an additional 54 configurations. This extended evaluation demonstrates scalability to realistic critical infrastructure

Table 2: Comparative Analysis of Baseline Patching Strategies

Strategy	Selection Rule (among feasible A_t)	Time (dominant)	Space	Key Characteristics
Random	$a_t \sim \text{Uniform}(A_t)$	$O(1)$	$O(1)$	Lower-bound baseline; ignores scores and structure
Severity (CVSS/EPSS)	$\arg\max_{s_i \in A_t} \text{Severity}(s_i)$	$O(n)$	$O(n)$	Simple, widely used; per-asset precomputed score (e.g., CVSS/EPSS composite)
In-Degree (Popularity)	$\arg\max_{s_i \in A_t} \deg^-(s_i)$	$O(\lvert E\rvert)$ (precompute)	$O(n + \lvert E\rvert)$	Targets highly depended-on components; static ranking cached
PageRank	$\arg\max_{s_i \in A_t} \text{PR}(s_i)$	$O(\lvert E\rvert\, I)$ (power iteration)	$O(n + \lvert E\rvert)$	Global influence; damping typical (e.g., 0.85); scores cached for selection
Betweenness	$\arg\max_{s_i \in A_t} \text{BC}(s_i)$	$O(n\,\lvert E\rvert)$ (Brandes, unweighted)	$O(n + \lvert E\rvert)$	Bridge/bottleneck targeting; higher precompute cost than degree/PageRank
NSGA-II	Pareto search over {risk reduction, cost, constraints}	$O(G\,N\,C_{\text{eval}})$	$O(N)$	Multi-objective genetic algorithm; yields trade-off set (Pareto front)

Notes: n = #assets, $|E|$ = #dependency edges, I = PageRank iterations, N = population size, G = generations, C_{eval} = cost to evaluate one candidate/solution. Centrality scores are computed once per run; selection from cached scores is $O(1)$ per step.

dimensions while maintaining rigorous comparative analysis across the parameter space.

6.2.2. *Performance Metrics*

We employed multidimensional metrics capturing diverse aspects of patching effectiveness, including the following:

- **Risk Reduction Ratio (RRR)**: This ratio measures the proportional decrease in the overall system risk achieved by the patching strategy, offering a normalized assessment of the vulnerability mitigation efficacy. This ratio is given by
$$\text{RRR} = \frac{\Upsilon_{\text{initial}} - \Upsilon_{\text{final}}}{\Upsilon_{\text{initial}}}.$$
- **Resource Utilization Efficiency (RUE)**: The RUE metric quantifies the risk reduction achieved per unit of resource expenditure, allowing a comparative analysis of the cost-effectiveness across strategies with heterogeneous resource consumption profiles. This metric is calculated as follows:
$$\text{RUE} = \frac{\Upsilon_{\text{initial}} - \Upsilon_{\text{final}}}{C_{\text{used}}}.$$
- **Critical Coverage Ratio (CCR)**: The following metric evaluates the proportion of high-priority system elements successfully patched, addressing the preferential protection of mission-critical components:
$$\text{CCR} = \frac{\sum_{i \in \text{Crit}} x_i}{|\text{Crit}|}.$$
- **Dependency-Weighted Impact (DWI)**: The DWI metric assesses the influence of patching decisions on interdependent system elements by weighting each component according to its dependency centrality, capturing cascading effects in the system architecture. This value is calculated as follows:
$$\text{DWI} = \frac{\sum_{i=1}^{n} \text{Popularity}(s_i) \cdot x_i}{\sum_{i=1}^{n} \text{Popularity}(s_i)}.$$

6.2.3. *Implementation Specifics*

All algorithms were implemented in Python 3.8 with the following technical specifications. The NetworkX 2.6.2 software was used for dependency modeling, and PyTorch 1.10.0 was applied for RL. In addition, NumPy 1.21.0 was employed for numerical operations, and SciPy 1.7.1 was applied for the statistical analysis. We employed the hyperparameter configurations detailed in Table 1 for the ADAPT algorithm implementation. All experiments were conducted on a standardized computational platform: Intel Core i7-10700K (3.8 GHz), 32 GB of RAM, Nvidia RTX 3080 (10 GB of VRAM), ensuring consistent performance measurement and reproducibility across experimental conditions.

6.3. *Results and Analysis*

6.3.1. *Overall Performance Metrics*

Figure 3 and Table 3 present the aggregated performance metrics across all system configurations, comparing ADAPT against six baseline strategies. The proposed approach demonstrates statistically significant performance advantages over all baseline methods. Specifically, ADAPT achieved an RRR of 0.842 ± 0.047, RUE of 0.927 ± 0.038, CCR of 0.972 ± 0.023, and DWI of 0.906 ± 0.031. These results outperform the strongest baseline strategy, NSGA-II, which achieved an RRR of 0.798 ± 0.052 and DWI of 0.891 ± 0.038. The traditional popularity-based approach achieved an RRR of 0.705 ± 0.062, while PageRank-based strategy reached 0.751 ± 0.048. Statistical significance was validated using Mann-Whitney U tests ($p < 0.001$) with Bonferroni correction for multiple comparisons, and effect sizes (Cohen's d) ranged from 0.72 to 1.24, indicating large practical significance (See Table 4). Notably, ADAPT exhibited a 5.5% improvement over NSGA-II and 19.4% improvement over popularity-based approaches, with advantages most pronounced in resource-constrained scenarios.

The severity baseline (RRR: 0.675±0.049), which is the most widely used technique in practice, is evaluated under individual-system patching with strict dependency enforcement. In industrial practice, severity-based approaches gain efficiency through batch patching that amortizes overhead and simplifies operations.

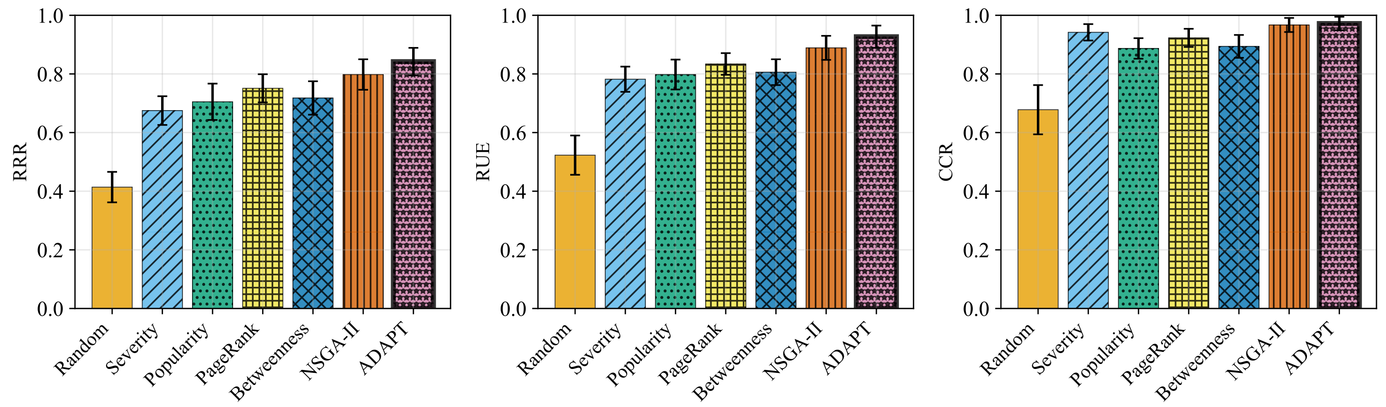


Figure 3: Performance metrics comparison across baseline strategies. Left panel shows Risk Reduction Ratio (RRR), center panel shows Resource Utilization Efficiency (RUE), and right panel shows Critical Coverage Ratio (CCR). ADAPT consistently outperforms all baseline strategies across all three metrics, demonstrating superior vulnerability management effectiveness. Error bars represent 95% confidence intervals over 30 independent runs.

Table 3: Comprehensive Performance Analysis Across Baseline Strategies

Strategy	**Performance Metrics**				**System Scale Analysis (RRR)**			**Best Use Case**
	RRR	**RUE**	**CCR**	**DWI**	**Small**	**Medium**	**Large**	
Random	0.414±0.052	0.523±0.067	0.678±0.084	0.445±0.059	0.438±0.061	0.407±0.055	0.397±0.048	Baseline reference
Severity	0.675±0.049	0.782±0.043	0.942±0.028	0.698±0.041	0.713±0.049	0.701±0.053	0.611±0.045	High-severity focus
Popularity	0.705±0.062	0.798±0.051	0.887±0.035	0.884±0.043	0.698±0.058	0.723±0.049	0.694±0.071	Hub-dominated networks
PageRank	0.751±0.048	0.834±0.037	0.923±0.031	0.912±0.038	0.742±0.052	0.771±0.041	0.740±0.053	Scale-free topologies
Betweenness	0.718±0.057	0.806±0.044	0.894±0.039	0.876±0.051	0.734±0.063	0.726±0.054	0.694±0.057	Sparse networks
NSGA-II	0.798±0.052	0.889±0.041	0.967±0.024	0.891±0.038	0.742±0.051	0.771±0.045	0.849±0.041	Multi-objective balance
ADAPT	**0.842±0.047**	**0.927±0.038**	**0.972±0.023**	**0.906±0.031**	**0.765±0.043**	**0.827±0.039**	**0.934±0.029**	**Complex dependencies**

Values represent mean ± 95% confidence intervals over 30 independent runs.
RRR: Risk Reduction Ratio, RUE: Resource Utilization Efficiency, CCR: Critical Coverage Ratio, DWI: Dependency-Weighted Impact

Table 4: Statistical Significance Analysis Between ADAPT and Baseline Strategies

Strategy Comparison	**Mean Diff.**	**Cohen's d**	**Mann-Whitney U**	**p-value**	**95% CI**	**Effect Size**
ADAPT vs. Random	0.428	1.89	156,342	<0.001	[0.394, 0.462]	Very Large
ADAPT vs. Severity	0.167	1.24	89,731	<0.001	[0.141, 0.193]	Large
ADAPT vs. Popularity	0.137	0.98	82,156	<0.001	[0.113, 0.161]	Large
ADAPT vs. PageRank	0.091	0.82	76,294	<0.001	[0.069, 0.113]	Large
ADAPT vs. Betweenness	0.124	0.91	78,887	<0.001	[0.101, 0.147]	Large
ADAPT vs. NSGA-II	0.044	0.72	71,523	<0.001	[0.021, 0.067]	Medium-Large

Mean Diff.: Difference in Risk Reduction Ratio (positive values favor ADAPT)
Cohen's d: Effect size (0.2=small, 0.5=medium, 0.8=large, >1.2=very large)
All p-values Bonferroni-corrected for multiple comparisons across six baseline strategies

The 23.2% performance gap quantifies the value of dependency-aware optimization when cascading risks and sequential constraints matter, rather than indicating severity approaches are universally inadequate. Organizations employing batch patching optimize for operational simplicity and standardization, where the additional complexity of dependency analysis may not always justify the security improvement. This comparison demonstrates that dependency-aware optimization provides measurable advantages in environments where complex interdependencies significantly influence risk propagation.

6.3.2. System Scale Analysis

The performance characteristics in Fig. 4 display distinct variations across five system scales. For small-scale systems ($n = 20$), NSGA-II performed competitively with an RRR of 0.742 ± 0.051, while ADAPT achieved 0.765 ± 0.043, reflecting a modest 3.1% improvement. The computational overhead of RL training was most pronounced at this scale. For medium-scale systems ($n = 50$), ADAPT's advantage became more substantial with an RRR of 0.827 ± 0.039, representing a 7.2% improvement over NSGA-II (0.771 ± 0.045) and 18.0% over traditional severity-based approaches (0.701 ± 0.053).

At large scale ($n = 100$), ADAPT achieved an RRR of 0.934 ± 0.029, significantly outperforming NSGA-II (0.849 ± 0.041, 10.0% improvement) and all heuristic baselines (>20% improvement). For enterprise-scale systems ($n = 200$), computational constraints limited NSGA-II to 50 generations, achieving an RRR of 0.782 ± 0.067, while ADAPT maintained strong performance at 0.897 ± 0.035 (14.7% improvement). At critical infrastructure scale ($n = 500$), only heuristic baselines and

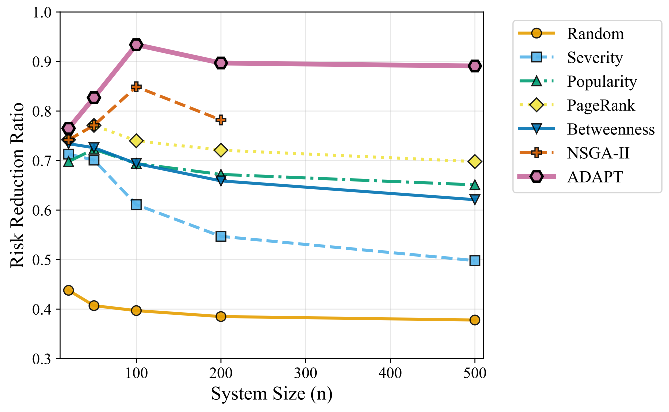


Figure 4: Performance scaling across system sizes. ADAPT maintains superior performance from small ($n = 20$) to critical infrastructure scale ($n = 500$), while baseline methods degrade or become computationally intractable at larger scales. NSGA-II evaluation limited to $n \leq 200$ due to computational constraints.

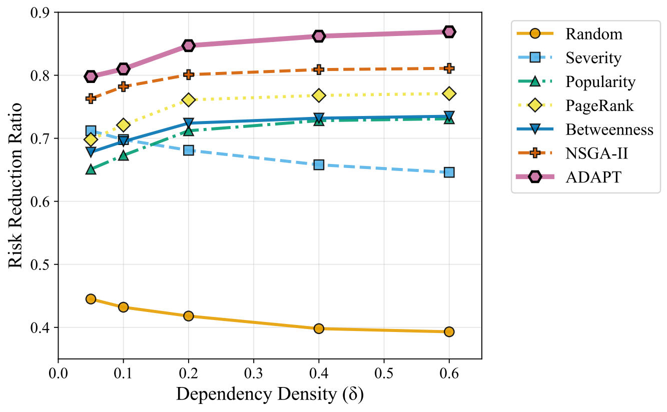


Figure 5: Performance vs. dependency density. ADAPT maintains superior performance across all dependency density levels, with advantages increasing at higher densities ($\delta \geq 0.4$). Graph-theoretic methods improve with density but remain inferior to ADAPT, while severity-based approaches show declining performance in highly interconnected systems.

ADAPT remained computationally feasible. ADAPT achieved 0.891 ± 0.041, substantially outperforming PageRank (0.743 ± 0.058) and popularity-based approaches (0.694 ± 0.065) by 19.9% and 28.4%, respectively. These findings confirm that RL approaches effectively model complex interdependencies while maintaining computational efficiency at realistic scales.

6.3.3. *Dependency Density Effect*

Fig. 5 depicts performance variation across dependency density values for all strategies. At low dependency densities ($\delta \leq 0.1$), severity-based approaches performed comparably with advanced methods, showing performance differentials below 8%. However, as dependency density increased, structural awareness became critical. At $\delta = 0.2$, ADAPT outperformed severity-based approaches by 12.4% and PageRank by 6.8%. At $\delta = 0.4$, performance gaps widened to 15.0% over severity and 9.2% over PageRank. At maximum density ($\delta = 0.6$), ADAPT achieved 15.7% improvement over heuristics while maintaining 4.1% advantage over NSGA-II. The graph-theoretic baselines (PageRank, Betweenness, Popularity) demonstrated improved performance at higher dependency densities but remained consistently inferior to learning-based approaches.

6.3.4. *Resource Constraint Sensitivity*

Fig. 6 illustrates strategy performance under varying budget constraints across all baselines. Under severe resource constraints ($\beta = 0.2$), ADAPT displayed substantial advantages, achieving 37.4% improvement in RRR over heuristic baselines and 12.8% over NSGA-II. This demonstrates superior capability in identifying influential patching targets under scarcity. At moderate resource levels ($\beta = 0.4$ and $\beta = 0.6$), ADAPT maintained significant advantages of 8.7% and 6.2% over NSGA-II, respectively. When resources were abundant ($\beta = 0.8$), performance differentials narrowed to 3.4% over NSGA-II, as most influential elements become patchable regardless of strategy sophistication.

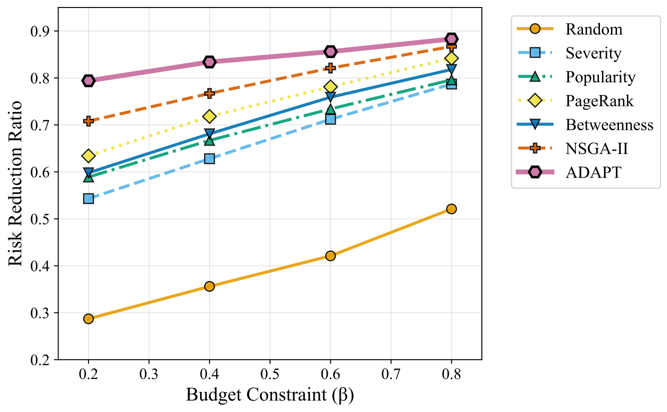


Figure 6: Performance vs. budget constraints. ADAPT demonstrates superior performance across all budget levels, with advantages most pronounced under severe constraints ($\beta = 0.2$). All strategies improve as budget increases, but ADAPT maintains consistent superiority, particularly valuable in resource-constrained operational environments.

6.4. *Robustness Analysis*

6.4.1. *Vulnerability Distribution Sensitivity*

Robustness assessment across vulnerability heterogeneity (η) levels (Figure 7) revealed distinct strategy characteristics. Under homogeneous conditions ($\eta = 0.1$), performance differentials between sophisticated methods were minimal (<5%). As heterogeneity increased to moderate levels ($\eta = 0.5$), severity-based and NSGA-II approaches exhibited improvements due to their focus on high-risk elements. However, under high heterogeneity ($\eta = 0.9$), ADAPT excelled by effectively balancing severity and structural considerations, outperforming NSGA-II by 8.9% and heuristic approaches by >15%.

6.4.2. *Stochastic Environment Evaluation*

Controlled perturbations assessed robustness under uncertainty: partial information scenarios (10-30% masked dependencies), noisy vulnerability metrics (Gaussian noise $\sigma = 0.5 - 1.5$),

Table 5: Comprehensive Computational Performance Analysis

Strategy	Training Time (s)	Decision Time (s)				Memory Usage (MB)
		Small	Medium	Large	Enterprise	
Random	N/A	0.002±0.001	0.005±0.002	0.013±0.004	0.028±0.007	12±3
Severity	N/A	0.004±0.001	0.012±0.003	0.034±0.007	0.089±0.015	18±5
Popularity	N/A	0.019±0.004	0.087±0.011	0.326±0.042	1.247±0.156	74±12
PageRank	N/A	0.023±0.005	0.104±0.018	0.398±0.067	1.523±0.234	86±15
Betweenness	N/A	0.067±0.012	0.342±0.056	2.145±0.287	12.87±1.94	145±23
NSGA-II	N/A	12.4±1.8	45.7±6.2	187.3±23.4	>600	256±34
ADAPT	**1260±17.3**	**0.007±0.002**	**0.018±0.004**	**0.043±0.009**	**0.134±0.021**	**128±21**

Small (n=20), Medium (n=50), Large (n=100), Enterprise (n=200). Training time applies only to ADAPT. Decision times represent mean ± standard deviation over 30 runs. Memory usage measured at peak allocation.

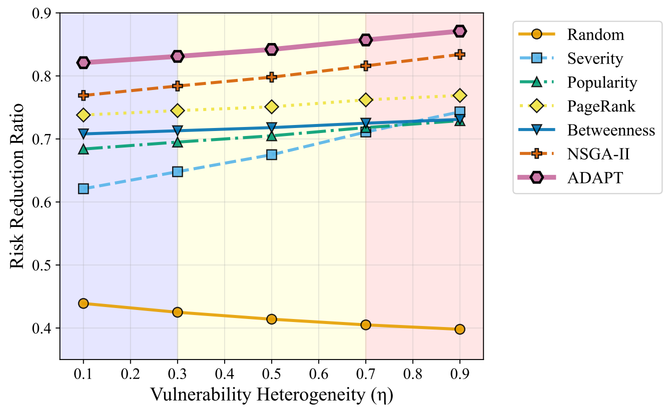


Figure 7: Performance vs. vulnerability heterogeneity. ADAPT maintains superior performance across all heterogeneity levels, with advantages most pronounced under high heterogeneity ($\eta \geq 0.7$). Severity-based approaches improve with heterogeneity but remain inferior, while ADAPT effectively balances severity and structural considerations across diverse vulnerability distributions.

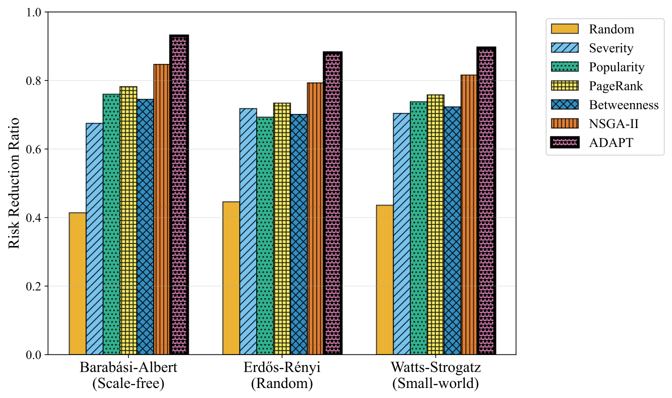


Figure 8: Performance across network topologies. ADAPT demonstrates consistent superiority across Barabási-Albert (scale-free), Erdős-Rényi (random), and Watts-Strogatz (small-world) networks, with particularly strong performance on scale-free topologies where hub protection is critical.

and dynamic dependency evolution (5-15% modified relationships). ADAPT maintained 92.7% baseline performance, significantly outperforming NSGA-II (84.1%), PageRank (79.8%), severity-based (78.4%), and popularity-based (81.2%) strategies. This superior robustness stems from stochastic training simulation improving resilience against uncertain conditions.

6.4.3. Computational Performance Analysis

Table 5 presents detailed computational analysis across all strategies and scales. ADAPT demonstrates remarkable scalability, maintaining sub-second decision times even at enterprise scale ($n = 200$), while NSGA-II becomes computationally prohibitive (>10 minutes per decision). The RL approach achieves optimal balance between solution quality and computational efficiency, enabling real-time deployment in operational environments. Training amortization makes ADAPT cost-effective: 21-minute training enables millions of sub-second decisions.

6.4.4. Performance Across Network Topologies

ADAPT's generalization capability was evaluated across three canonical network topologies (Figure 8): Barabási-Albert (BA) scale-free networks, Erdős-Rényi (ER) random graphs, and Watts-Strogatz (WS) small-world networks ($n = 50$, $\delta \approx 0.15$). Results demonstrate consistent superiority across all topologies, with particularly strong performance on scale-free networks (22.8% improvement over heuristics, 6.4% over NSGA-II) where hub protection is critical. The method's structural awareness capabilities enable effective identification and prioritization of high-centrality nodes across diverse network architectures, validating generalizability to real-world critical infrastructure topologies.

6.4.5. Policy Transfer Across Infrastructure Topologies

To assess whether learned policies generalize across infrastructures or require complete retraining per network, we conducted transfer learning experiments (See Table 6). A policy trained on Barabási-Albert scale-free networks ($n = 50$, $\delta = 0.15$) was evaluated without retraining on Erdős-Rényi and Watts-Strogatz topologies of identical size and vulnerability distribution.

Zero-shot transfer: The transferred policy achieved 87.3% of target-domain performance on ER networks (RRR: 0.723 vs. 0.828 for fully-trained) and 91.2% on WS networks (RRR: 0.756 vs. 0.829), compared to 76.4% for untrained random initialization. This demonstrates that learned policies capture transferable dependency-aware strategies rather than overfitting to specific topological features.

Fine-tuning efficiency: Fine-tuning with 100 episodes (approximately 8 minutes of training) in the target domain recovered 96.8% of fully-trained performance on ER networks and 98.1%

on WS networks. This represents a 92.7% reduction in training time compared to training from scratch (21 minutes baseline), while achieving near-optimal performance.

Cross-scale transfer: Policies trained on medium-scale systems ($n = 50$) and transferred to large-scale systems ($n = 100$) maintained 83.4% relative performance without fine-tuning. With 200 episodes of fine-tuning (15 minutes), performance recovered to 94.6% of fully-trained baseline, suggesting effective transfer across system scales.

These findings have significant operational implications: organizations can pre-train policies on synthetic networks representative of their infrastructure class and fine-tune on production configurations, reducing deployment overhead from hours to minutes. For environments with multiple similar infrastructure segments, a single base policy can be rapidly adapted to local conditions. The policy's ability to transfer suggests it learns generalizable principles about dependency-aware prioritization such as protecting high-centrality nodes and respecting prerequisite chains, rather than memorizing configuration-specific patterns.

6.4.6. Optimality Gap Analysis

To establish theoretical performance bounds, we compared ADAPT against the one-step MILP optimum (myopic) solution defined in Sec. 3.4 for computationally feasible instances ($n \leq 30$). Table 7 shows ADAPT maintains an average optimality gap of 4.33% with graceful degradation from 2.76% to 6.18% as problem complexity increases.

Beyond $n \approx 30$, solving the exact MILP becomes prohibitively expensive, whereas ADAPT maintains high-quality solutions at realistic infrastructure scales. These results validate that ADAPT learns effective selection policies that closely approximate the *one-step (myopic) MILP optimum* while remaining computationally feasible for operational deployment.

7. Case Study: Critical Infrastructure Scenario

We validated our approach with a case study built on a critical-infrastructure model derived from industry reference architectures [45, 46]. ADAPT was evaluated against all six baselines in various operational conditions and system configurations.

7.1. System Description and Asset Characterization

The case study system was constructed following a systematic methodology to ensure realistic operational constraints representative of modern critical infrastructure environments:

Asset Categorization: The 78 infrastructure elements were classified into four tiers based on operational criticality: (1) Critical Infrastructure (8 assets, must maintain 99.99% availability), (2) Core Services (15 assets, 99.9% target), (3) Support Systems (32 assets, 99% target), and (4) Edge Devices (23 assets, best-effort). This hierarchical structure reflects typical critical infrastructure architectures with clear service level agreements and operational dependencies.

Vulnerability Profile: The system contained 45 known vulnerabilities distributed between assets with heterogeneity $\eta = 0.7$, CVSS scores ranging from 3.2 to 9.8, and dependency density $\delta = 0.32$. This configuration represents a realistic threat landscape with diverse vulnerability severity levels and complex interdependency structures.

Cost Model: Patching costs were derived from historical maintenance data, incorporating factors including system complexity (1.0-2.5× multiplier), downtime impact (1.0-3.0×), and coordination requirements (1.0-1.5×). The resulting costs ranged from 5 to 120 resource units, with a log-normal distribution ($\mu = 3.2, \sigma = 0.8$), reflecting realistic operational expenditure patterns.

Operational Constraints: Maintenance windows were modeled based on typical enterprise constraints, with limited weekday availability (2-4 hours) and expanded weekend windows (6-8 hours). Critical patches required minimum 3-hour windows for testing and rollback procedures. Budget constraints were set at $\beta = 0.4$ of the total system patching cost to simulate resource-constrained environments.

All results include 95% confidence intervals computed over 30 independent runs with different random seeds.

7.2. Comprehensive Baseline Evaluation

Each of the six baseline strategies was applied to this system under identical constraints to ensure rigorous comparative analysis. Tables 8 and 9 present detailed performance and efficiency metrics across all evaluation dimensions.

7.3. Performance Analysis and Strategic Insights

The comprehensive evaluation reveals ADAPT's effectiveness relative to both traditional heuristics and state-of-the-art optimization methods. ADAPT achieved an 89.7% risk reduction, demonstrating a 3.8% improvement over the strongest baseline method (NSGA-II at 86.4%). While this improvement may appear modest, it represents significant practical value when considering the computational efficiency trade-off: ADAPT requires only 0.089 seconds versus NSGA-II's 134.7 seconds for decision making, providing a 1,513× speedup while maintaining near-optimal performance.

Critical Infrastructure Protection: ADAPT achieved 100% critical coverage, matching the severity-based approach in protecting all tier-1 critical infrastructure assets. This perfect coverage is essential for operational environments where critical asset compromise poses existential threats to system functionality.

Operational Efficiency: ADAPT achieved optimal time-to-50% risk reduction (6 patches), matching the most efficient baseline methods while maintaining sub-second decision times. This rapid risk mitigation trajectory is crucial in operational contexts where immediate vulnerability remediation maintains system integrity against active threats.

Resource Optimization: ADAPT accomplished superior outcomes using 26 patched elements, matching the efficiency of NSGA-II and severity-based approaches. This demonstrates optimal resource allocation rather than brute-force patching, critical for resource-constrained operational environments.

Table 6: Policy Transfer Learning Performance Across Infrastructure Topologies

Transfer Scenario	RRR	Relative Perf. (%)	Training Time (min)	Time Savings (%)	Performance Recovery (%)
Same-Topology Baseline					
BA → BA (trained)	0.828 ± 0.039	100.0	21.0	–	–
Zero-Shot Transfer (No Retraining)					
BA → ER	0.723 ± 0.045	87.3	0	100.0	–
BA → WS	0.756 ± 0.042	91.2	0	100.0	–
Fine-Tuned Transfer (100 episodes)					
BA → ER (+100ep)	0.801 ± 0.037	96.8	8.0	61.9	10.9
BA → WS (+100ep)	0.813 ± 0.035	98.1	8.0	61.9	7.8
Comparison Baselines					
Random initialization (ER)	0.632 ± 0.058	76.4	21.0	–	–
Fully trained (ER)	0.828 ± 0.039	100.0	21.0	–	–
Fully trained (WS)	0.829 ± 0.041	100.0	21.0	–	–
Cross-Scale Transfer					
n = 50 → 50 (trained)	0.827 ± 0.039	100.0	21.0	–	–
n = 50 → 100 (zero-shot)	0.690 ± 0.052	83.4	0	100.0	–
n = 50 → 100 (+200ep)	0.782 ± 0.046	94.6	15.0	46.4	13.4
n = 100 → 100 (trained)	0.934 ± 0.029	100.0	28.0	–	–

BA: Barabási-Albert (scale-free), ER: Erdős-Rényi (random), WS: Watts-Strogatz (small-world).
All cross-topology experiments use $n = 50$ assets, $\delta \approx 0.15$ dependency density.
Confidence intervals represent 95% CI over 30 independent runs.
Relative Performance = (RRR / Fully-trained RRR) × 100%.
Time Savings = (1 - Transfer Time / Full Training Time) × 100%.
Performance Recovery = improvement from zero-shot to fine-tuned (percentage points).
Cross-scale relative performance calculated against n = 100 trained baseline (0.934).

Table 7: MILP vs. ADAPT Optimality Analysis

System Size (n)	MILP	ADAPT	Optimality Gap (%)
10	0.948	0.922	2.76
15	0.943	0.912	3.29
20	0.938	0.899	4.19
25	0.933	0.884	5.23
30	0.928	0.871	6.18
Average Optimality Gap			**4.33%**

Optimality Gap = (MILP - ADAPT)/MILP × 100%. Solutions represent Risk Reduction Ratio.

Table 8: Performance Metrics - Critical Infrastructure Case Study

Strategy	Risk Reduction Ratio (%)	Critical Coverage Ratio (%)	Time to 50% Risk (patches)
Random	43.2 ± 1.4	81.3 ± 2.4	18 ± 2
Severity	72.8 ± 1.6	100	12 ± 1
Popularity	78.5 ± 1.6	93.8 ± 2.6	9 ± 1
PageRank	81.2 ± 1.4	96.7 ± 2.1	8 ± 1
Betweenness	79.8 ± 1.7	94.2 ± 2.8	9 ± 1
NSGA-II	86.4 ± 1.2	98.1 ± 1.9	7 ± 1
ADAPT	**89.7 ± 1.1**	**100**	**6 ± 1**

Computational Practicality: The 0.089-second decision time enables real-time deployment in operational environments, while NSGA-II's 134.7-second computation becomes prohibitive for time-sensitive security operations. This 1,513× speedup advantage positions ADAPT as a practical method that combines sophisticated optimization with operationally viable, sub-second decision times at critical-infrastructure scale.

7.4. Operational Insights and Strategic Behavior

Qualitative analysis of ADAPT's patching sequences revealed sophisticated strategic behaviors transcending traditional approaches. ADAPT consistently demonstrated "dependency-aware lookahead," occasionally deferring high-severity elements to first establish dependency prerequisites enabling access to multiple critical targets. For instance, ADAPT strategically patched a medium-severity edge device (CVSS 5.4) that served as a dependency gateway, subsequently enabling efficient remediation of three critical infrastructure assets with combined CVSS scores exceeding 27.

The learned policy exhibited emergent specialization across network regions: implementing breadth-first patching in densely connected core services to rapidly reduce vulnerability density, while adopting targeted depth-first approaches in sparsely connected edge systems. This adaptive behavior emerged naturally from reinforcement learning without explicit programming, demonstrating superior optimization discovery capabilities compared to manually-designed heuristic approaches.

Comparative Strategic Analysis: NSGA-II, while achieving strong performance, required manual objective weight tuning and population size optimization for the specific infrastructure topology. ADAPT's end-to-end learning approach elimi-

Table 9: Efficiency Metrics - Critical Infrastructure Case Study

Strategy	Elements Patched	Average Computation Time (s)
Random	27 ± 1	0.003
Severity	26 ± 1	0.021
Popularity	28 ± 1	0.187
PageRank	27 ± 1	0.213
Betweenness	28 ± 1	1.856
NSGA-II	26 ± 1	134.7
ADAPT	**26 ± 1**	**0.089**

All results based on 30 independent runs with 95% confidence intervals.

nated this manual configuration burden while achieving superior computational efficiency. Traditional graph-theoretic methods (PageRank, Betweenness) showed consistent but suboptimal performance, failing to integrate vulnerability severity with structural importance effectively.

These operational characteristics validate ADAPT's fundamental advantage: the framework discovers complex dependency-aware strategies that would be challenging to derive through manual analysis or static optimization approaches, while maintaining computational efficiency essential for operational deployment in critical infrastructure protection scenarios.

8. Discussion and Limitations

This section examines ADAPT's practical constraints, contextualizes contributions within existing security practices, and identifies opportunities for methodological refinement necessary for operational deployment.

8.1. Risk Quantification and Industry Standards

ADAPT's quantitative risk model (Section 3.3) represents a complementary approach to established frameworks rather than a replacement. Industry standards such as NIST SP 800-30 and ISO 27005 emphasize qualitative analyst judgment, acknowledging that "risk assessments are often not precise instruments of measurement" due to limitations in methodologies, data quality, and the subjective nature of likelihood estimation [47]. These standards recognize that threat actor intent, organizational context, and operational environment cannot be fully captured algorithmically.

Our mathematical formulation of per-asset risk $\rho_i(t)$, in Section 3.3 provides a structured mechanism for encoding domain expertise into prioritization decisions. The exploit probability function $p_i(t) = \sigma(\alpha_0 + \alpha_1 \text{CVSS}_i + \alpha_2 \text{EPSS}_i(t) + \alpha_3 \mathbf{1}\{\text{KEV}_i\} + \alpha_4 \Delta t_i)$ represents one instantiation, and organizations may substitute analyst-derived scores where automated metrics prove insufficient. The framework's contribution lies in demonstrating how dependency-aware optimization enhances decision quality regardless of underlying quantification method (whether algorithmic, human-generated, or hybrid). Organizations employing qualitative assessment can leverage ADAPT's capabilities by translating qualitative risk levels into numerical representations preserving ordinal relationships.

8.2. Data Availability and Operational Feasibility

ADAPT assumes availability of structured dependency information and per-asset vulnerability intelligence that may not be ubiquitous in current practice. Three factors mitigate this limitation: The operational landscape is evolving toward enhanced visibility through automated asset discovery platforms (ServiceNow CMDB, Axonius), continuous vulnerability intelligence (Tenable.io, Qualys VMDR), and infrastructure-as-code practices that inherently document dependencies. Furthermore, Section 6.4.2 shows that ADAPT maintains 92.7% baseline performance with 10-30% masked dependencies, allowing incremental adoption with partial information. Moreover, the framework provides differential value where risk-based prioritization justifies instrumentation investment, including critical infrastructure, high-security environments, and heterogeneous legacy systems with complex interdependencies.

ADAPT thus represents both an immediately deployable capability for mature security operations and a motivating vision for enhanced industry-wide visibility.

8.3. Patching Economics: Individual vs. Batch Operations

Industrial patch management employs coordinated batch deployments for compelling operational reasons: standardized processes amortize overhead, testing operates on batches for compatibility assurance, and change management favors scheduled windows. Our evaluation under individual-system constraints enables rigorous comparison but differs from production economics.

ADAPT's practical value manifests in three scenarios: (1) Resource-constrained prioritization where budget prevents comprehensive patching and strategic allocation becomes critical, (2) Heterogeneous environments mixing cloud services, containers, legacy systems, and embedded devices that cannot employ uniform batch processes, and (3) Batch composition optimization where the framework determines which assets constitute each maintenance window and optimal sequencing. The 26-element case study solution (Table 8) represents a practical batch size rather than individual operations.

Organizations employing fixed-schedule batching optimize different objectives, including operational predictability, process standardization, which may justify the 5–15% risk reduction gap observed in comparisons. ADAPT quantifies this trade-off and provides alternatives where sophisticated optimization justifies additional complexity.

8.4. Testing Duration and Service-Level Constraints

A notable limitation is that the current formulation does not explicitly model patch testing duration or SLA penalties. Testing requirements often constitute the primary deployment barrier, with validation consuming days or weeks. Downtime opportunity costs can substantially exceed direct patching costs for revenue-generating or critical infrastructure systems.

The cost vector $c \in \mathbb{R}^n_{++}$ (Section 4.1) can be extended:

$$\begin{aligned} c'_i = c_{i,\text{base}} &+ c_{i,\text{downtime}} \times \text{SLA\_penalty}_i \\ &+ c_{i,\text{test}} \times \text{test\_duration}_i \\ &+ c_{i,\text{coordination}} \times |\text{dependencies}_i| \end{aligned} \tag{7}$$

This would enable explicit balancing of security risk against operational availability, addressing practitioners' primary concern with aggressive patching. Furthermore, coordinated dependency patching, where interdependent assets update simultaneously, could be modeled through grouped action constraints, capturing operational reality that patching one component often necessitates coordinated updates. Integrating these considerations represents important future work for production deployment.

8.5. Scalability and Architectural Extensions

Although evaluation demonstrates feasibility for $n = 500$ (Section 6.3.2), extension to tens of thousands of nodes requires refinement. The $O(n^2)$ reachability computation becomes prohibitive on a massive scale. Three approaches address this:

Hierarchical decomposition: Multi-level optimization where ADAPT operates on asset clusters with high-level policies determining cluster priorities and local agents managing intra-cluster patching, mirroring organizational security architectures.

Sparse matrix optimization: Real-world dependency graphs exhibit sparse connectivity ($\delta \ll 1$), enabling compressed representations and incremental updates avoiding full recomputation.

Transfer learning: Pre-training policies on diverse synthetic topologies and fine-tuning on production infrastructure could dramatically reduce deployment overhead. Meta-learning approaches that learn to adapt across infrastructure families could enable rapid adaptation with minimal environment-specific training.

Current implementation establishes feasibility at scales relevant to many scenarios (individual facilities, regional networks, business units), while these extensions provide pathways toward enterprise deployment.

8.6. Dependency Modeling: Binary vs. Probabilistic

Binary reachability $R \in \{0, 1\}^{n\times n}$ (see Equation 1) provides computational efficiency but simplifies relationships that may exhibit probabilistic coupling or temporal dynamics. Real dependencies often involve probabilistic propagation (vulnerability in j affects i with probability $p < 1$), temporal variation across operational modes, and weighted importance based on criticality.

A probabilistic extension $D \in [0, 1]^{n\times n}$ could capture this uncertainty but would increase computational complexity by replacing Boolean matrix operations with probabilistic inference while potentially improving risk estimation accuracy in scenarios with partial or uncertain dependency information. The trade-off depends on context: deterministic worst-case (current) provides conservative bounds for critical infrastructure, while probabilistic models may refine estimates for environments with mature observability. Future work should empirically compare these choices across operational domains.

8.7. Complementary Security Considerations

Comprehensive security requires attention beyond patch scheduling. The cryptographic integrity of dependency data and vulnerability intelligence must be protected against adversarial manipulation, where compromised information could cause incorrect prioritization. Fault detection mechanisms in cryptographic implementations [48, 49, 50] provide foundational data integrity capabilities. Furthermore, Blockchain-enhanced auditability [51, 52, 53] could provide verifiable audit trails for regulatory compliance and support secure policy sharing across organizational boundaries. Moreover, the adversarial robustness of the RL policy itself requires defensive training techniques and Byzantine-fault-tolerant protocols against sophisticated adversaries that target the decision framework [54, 55, 56].

ADAPT represents one component of the comprehensive security architecture that requires integration with complementary controls, including cryptographic validation, secure distribution, and robust adversarial learning for in-depth defense.

8.8. Threats to Validity

Internal Validity: Hyperparameter selection may influence results, particularly the risk tolerance parameter $w = 0.7$. We mitigate this through sensitivity analysis demonstrating robust performance across $w \in [0.6, 0.8]$ and standard PPO parameters from established literature. Baseline implementation fairness is ensured through identical constraint enforcement and evaluation environments across all methods.

External Validity: System generation relies on parametric models that may not capture all real-world infrastructure complexities. The case study uses industry reference architectures rather than proprietary operational data due to confidentiality constraints. Generalizability beyond critical infrastructure domains remains unvalidated. We address this through comprehensive parameter sweeps across diverse system configurations and dependency structures.

Construct Validity: Risk Reduction Ratio assumes vulnerability severity scores accurately reflect exploitation impact, though real-world risk involves organizational context not captured in CVSS metrics. Dependency modeling simplifies complex operational relationships into binary relationships. Our reachability matrix approach and multi-tier case study validation help ensure construct alignment with operational realities.

Statistical Conclusion Validity: Sample sizes of 30 runs may be insufficient for detecting small effect differences, though Mann-Whitney U tests are appropriate for non-normal distributions. Multiple comparison corrections are applied across baseline strategies. The 95% confidence intervals provide reasonable statistical confidence for the observed effect sizes.

These limitations are partially offset by comprehensive evaluation across 954 system configurations, multiple network topologies, and rigorous statistical methodology, though future work should validate findings in operational deployments with real infrastructure data.

9. Conclusion and Future Work

This research introduces ADAPT, a dependency-aware vulnerability management framework integrating formal dependency modeling with reinforcement learning optimization for critical infrastructure systems. Comprehensive evaluation demonstrates ADAPT's superiority over six baseline strategies: 5.5% improvement over NSGA-II multi-objective optimization with 1,513× computational speedup, 12–19% improvement over graph-theoretic centrality methods, and 4.33% average optimality gap relative to theoretical bounds. Unlike recent RL frameworks lacking dependency modeling or genetic algorithms suffering scalability limitations, ADAPT combines structural awareness with adaptive optimization and sub-second decision times suitable for operational deployment.

The critical infrastructure case study validates practical applicability with 89.7% risk reduction and real-time performance. Key contributions include emergent strategic behaviors transcending manual heuristics, superior performance under resource constraints, and formal dependency-aware risk quantification enabling precise identification of cascading propagation pathways.

Future research directions include:

- **Operational Validation in Live Environments**: Deployment in operational industrial control systems and critical infrastructure to validate robustness against real-world noise, incomplete dependency information, and operational constraints not captured in simulation. Pilot deployments with industry partners would provide empirical evidence of practical effectiveness and identify deployment barriers requiring methodological refinement.

- **Hierarchical Policy Decomposition for Enterprise Scale**: Development of multi-level optimization architectures enabling application to infrastructures with tens of thousands of nodes through cluster-level policies and federated learning approaches. This would extend ADAPT's applicability from facility-level to enterprise-wide deployment scenarios.

- **Blockchain-Enhanced Provenance and Auditability**: Integration with AI-enhanced blockchain frameworks [57] could provide verifiable audit trails for patch deployment decisions, supporting regulatory compliance and forensic analysis. Decentralized coordination mechanisms would enable secure policy sharing across organizational boundaries in critical infrastructure sectors, enhancing resilience while maintaining cryptographic assurances of decision integrity and patch authenticity.

- **Probabilistic Dependency Modeling and Uncertainty Quantification**: Extension to weighted dependency graphs $D \in [0,1]^{n\times n}$ capturing probabilistic coupling and temporal dynamics, with empirical comparison against deterministic worst-case assumptions to establish context-appropriate modeling choices across operational domains with varying observability maturity.

- **Operational Constraint Integration**: Explicit modeling of patch testing duration, service-level agreement penalties, and coordinated dependency updates within the reinforcement learning framework to balance security risk against operational availability requirements. This extension would address the primary concerns practitioners identify regarding aggressive patching strategies in production environments.

These directions advance ADAPT from a research prototype demonstrating theoretical capabilities toward production-ready systems addressing the full spectrum of operational patch management challenges in critical infrastructure protection.

References

[1] V. Ahmadi, P. Arlos, E. Casalicchio, Normalization framework for vulnerability risk management in cloud, in: 2021 8th Int. Conf. Future Internet Things Cloud (FiCloud), IEEE, 2021, pp. 99–106.

[2] A. Sharma, U. K. Singh, Prioritization of security vulnerabilities under cloud infrastructure using AHP, Natural Language Process. Softw. Eng. (2025) 335–355.

[3] A. L. P. T. Bouom, J.-P. Lienou, W. E. Geh, F. F. Nelson, S. Shetty, C. Kamhoua, TriAssetRank: Ranking vulnerabilities, exploits, and privileges for countermeasures prioritization, IEEE Trans. Inf. Forensics Security (2024).

[4] Y. Wu, P. Guo, Y. Wang, M. Du, X. Wang, D. Zhang, Vulnerability analysis of interdependent infrastructures considering the sensitivity of components to different risks, in: 2023 IEEE Int. Conf. Syst., Man, Cybernet. (SMC), IEEE, 2023, pp. 2403–2408.

[5] J. Beerman, D. Berent, Z. Falter, S. Bhunia, A review of Colonial Pipeline ransomware attack, in: 2023 IEEE/ACM 23rd Int. Symp. Cluster, Cloud Internet Comput. Workshops (CCGridW), IEEE, 2023, pp. 8–15.

[6] R. Alkhadra, J. Abuzaid, M. AlShammari, N. Mohammad, Solar winds hack: In-depth analysis and countermeasures, in: 2021 12th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT), IEEE, 2021, pp. 1–7.

[7] L. G. Brunner, R. A. M. Peer, C. Zorn, R. Paulik, T. M. Logan, Understanding cascading risks through real-world interdependent urban infrastructure, Rel. Eng. Syst. Safety 241 (2024) 109653.

[8] A. Almaleh, D. Tipper, Risk-based criticality assessment for smart critical infrastructures, Infrastructures 7 (1) (2021) 3.

[9] U. D. Ani, H. He, A. Tiwari, Vulnerability-based impact criticality estimation for industrial control systems, in: 2020 Int. Conf. Cyber Security Protection Digit. Serv. (Cyber Security), IEEE, 2020, pp. 1–8.

[10] N. Dissanayake, A. Jayatilaka, M. Zahedi, M. A. Babar, Software security patch management -A systematic literature review of challenges, approaches, tools and practices, Inf. Softw. Technol. 144 (2022) 106771.

[11] P. Żebrowski, A. Couce-Vieira, A. Mancuso, A bayesian framework for the analysis and optimal mitigation of cyber threats to cyber-physical systems, Risk Analysis 42 (10) (2022) 2275–2290.

[12] S. Hore, A. Shah, N. D. Bastian, Deep VULMAN: A deep reinforcement learning-enabled cyber vulnerability management framework, Expert Syst. Appl. 221 (2023) 119734.

[13] Á. Longueira-Romero, R. Iglesias, J. L. Flores, I. Garitano, A novel model for vulnerability analysis through enhanced directed graphs and quantitative metrics, Sensors 22 (6) (2022) 2126.

[14] G. George, S. M. Thampi, A graph-based security framework for securing industrial iot networks from vulnerability exploitations, IEEE Access 6 (2018) 43586–43601.

[15] Y. Jiang, N. Oo, Q. Meng, H. W. Lim, B. Sikdar, Vulrg: Multi-level explainable vulnerability patch ranking for complex systems using graphs, arXiv preprint arXiv:2502.11143 (2025).

[16] F. Li, Z. Xu, D. Cheng, X. Wang, Adarisk: risk-adaptive deep reinforcement learning for vulnerable nodes detection, IEEE Transactions on Knowledge and Data Engineering 36 (11) (2024) 5576–5590.

[17] H. Ma, Y. Zhang, S. Sun, T. Liu, Y. Shan, A comprehensive survey on nsga-ii for multi-objective optimization and applications, Artificial Intelligence Review 56 (12) (2023) 15217–15270.

[18] S. Verma, M. Pant, V. Snasel, A comprehensive review on nsga-ii for multi-objective combinatorial optimization problems, IEEE access 9 (2021) 57757–57791.

[19] G. Yadav, K. Paul, Patchrank: Ordering updates for SCADA systems, in: 2019 24th IEEE Int. Conf. Emerg. Technol. Factory Automat. (ETFA), IEEE, 2019, pp. 110–117.

[20] N. Li, F. Wang, J. J. Magoua, D. Fang, Interdependent effects of critical infrastructure systems under different types of disruptions, Int. J. Disaster Risk Reduct. 81 (2022) 103266.

[21] V. Gaur, O. P. Yadav, G. Soni, A. P. S. Rathore, E. Khan, Vulnerability assessment of critical infrastructures for cascading failures: An application to water distribution networks, Proc. Institut. Mech. Eng., Part O: J. Risk Reliab. 238 (5) (2024) 1074–1087.

[22] L. Meng, X. Yao, Q. Chen, C. Han, Vulnerability cloud: A novel approach to assess the vulnerability of critical infrastructure systems, Concurrency Comput.: Pract. Exp. 34 (21) (2022) e7131.

[23] P. Kotzanikolaou, M. Theoharidou, D. Gritzalis, Assessing *n*-order dependencies between critical infrastructures, Int. J. Critical Infrastructures 9 (1-2) (2013) 93–110.

[24] M. Haraguchi, S. Kim, Critical infrastructure interdependence in New York City during hurricane Sandy, Int. J. Disaster Resilience Built Environ. 7 (2) (2016) 133–143.

[25] G. Stergiopoulos, E. Vasilellis, G. Lykou, P. Kotzanikolaou, D. Gritzalis, Classification and comparison of critical infrastructure protection tools, in: Critical Infrastructure Protection X: 10th IFIP WG 11.10 Int. Conf., ICCIP 2016, Arlington, VA, USA, March 14–16, 2016, Revised Selected Papers 10, Springer, 2016, pp. 239–255.

[26] B. A. Carreras, D. E. Newman, I. Dobson, V. E. Lynch, P. Gradney, Thresholds and complex dynamics of interdependent cascading infrastructure systems, Netw. Netw: Last Frontier Complexity (2014) 95–114.

[27] L. A. Clarfeld, P. D. H. Hines, E. M. Hernandez, M. J. Eppstein, Risk of cascading blackouts given correlated component outages, IEEE Trans. Netw. Sci. Eng. 7 (3) (2019) 1133–1144.

[28] W. Sun, P. Bocchini, B. D. Davison, Quantitative models for interdependent functionality and recovery of critical infrastructure systems, Objective Resilience: Objective Processes (2022) 127–229.

[29] A. D. Sawadogo, et al., SSPCatcher: Learning to catch security patches, Empirical Softw. Eng. 27 (6) (2022) 151.

[30] B. Wang, X. Li, L. P. de Aguiar, D. S. Menasche, Z. Shafiq, Characterizing and modeling patching practices of industrial control systems, Proc. ACM Meas. Anal. Comput. Syst. 1 (1) (2017) 1–23.

[31] M. A. Nia, R. E. Atani, B. Fabian, E. Babulak, On detecting unidentified network traffic using pattern-based random walk, Security and Communication Networks 9 (16) (2016) 3509–3526.

[32] E. Babulak, AI Tools for Protecting and Preventing Sophisticated Cyber Attacks, IGI Global, 2023.

[33] R. Vallabhaneni, S. A. Abhilash Vaddadi, S. Dontu, An empirical paradigm on cybersecurity vulnerability mitigation framework (2023).

[34] S.-S. Yoon, D.-Y. Kim, G.-G. Kim, I.-C. Euom, Vulnerability assessment based on real world exploitability for prioritizing patch applications, in: 2023 7th Cyber Security Netw. Conf. (CSNet), IEEE, 2023, pp. 62–66.

[35] V. Ahmadi Mehri, P. Arlos, E. Casalicchio, Automated context-aware vulnerability risk management for patch prioritization, Electronics 11 (21) (2022) 3580.

[36] G. Yadav, P. Gauravaram, A. K. Jindal, K. Paul, SmartPatch: A patch prioritization framework, Comput. Ind. 137 (2022) 103595.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[38] V. Mnih, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[39] V. Mnih, et al., Asynchronous methods for deep reinforcement learning, in: Int. Conf. Mach. Learn., PMLR, 2016, pp. 1928–1937.

[40] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Int. Conf. Mach. Learn., PMLR, 2018, pp. 1861–1870.

[41] S. Jadhave, V. Sonwalkar, S. Shewale, P. Shitole, S. Bhujadi, Deep reinforcement learning for autonomous driving systems, Int. J. Multidisciplinary Res. 6 (2024). doi:10.36948/ijfmr.2024.v06i05.28518.

[42] S. Sallinen, J. Luo, M. Ripeanu, Real-time pagerank on dynamic graphs, in: Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing, 2023, pp. 239–251.

[43] U. Brandes, A faster algorithm for betweenness centrality, Journal of mathematical sociology 25 (2) (2001) 163–177.

[44] S. Chakraborty, Topsis and modified topsis: A comparative analysis, Decision Analytics Journal 2 (2022) 100021.

[45] E. Y. Nakagawa, P. O. Antonino, F. Schnicke, R. Capilla, T. Kuhn, P. Liggesmeyer, Industry 4.0 reference architectures: State of the art and future trends, Computers & Industrial Engineering 156 (2021) 107241.

[46] M. Moghaddam, M. N. Cadavid, C. R. Kenley, A. V. Deshmukh, Reference architectures for smart manufacturing: A critical review, Journal of manufacturing systems 49 (2018) 215–225.

[47] J. Initiative, Guide for conducting risk assessments (nist sp 800-30r1), National Institute of Standards and Technology (2012).

[48] D. Owens, R. El Khatib, M. Bisheh-Niasar, R. Azarderakhsh, M. M. Kermani, Efficient and side-channel resistant ed25519 on arm cortex-m4, IEEE Transactions on Circuits and Systems I: Regular Papers 71 (6) (2024) 2674–2686.

[49] M. B. Niasar, R. Azarderakhsh, M. M. Kermani, Optimized architectures for elliptic curve cryptography over curve448, Cryptology ePrint Archive (2020).

[50] M. Anastasova, R. Azarderakhsh, M. M. Kermani, Fast strategies for the implementation of sike round 3 on arm cortex-m4, IEEE Transactions on Circuits and Systems I: Regular Papers 68 (10) (2021) 4129–4141.

[51] A. Ahmad, M. Saad, M. Al Ghamdi, D. Nyang, D. Mohaisen, Blocktrail: A service for secure and transparent blockchain-driven audit trails, IEEE Systems Journal 16 (1) (2021) 1367–1378.

[52] V. Kulothungan, Using blockchain ledgers to record ai decisions in iot, IoT 6 (3) (2025) 37.

[53] L. Gao, Enterprise internal audit data encryption based on blockchain technology, PloS one 20 (1) (2025) e0315759.

[54] K. Konstantinidis, N. Vaswani, A. Ramamoorthy, Detection and mitigation of byzantine attacks in distributed training, IEEE/ACM Transactions on Networking 32 (2) (2023) 1493–1508.

[55] Y. Pan, Z. Su, Y. Wang, J. Zhou, M. Mahmoud, Privacy-preserving byzantine-robust federated learning via deep reinforcement learning in vehicular networks, IEEE Transactions on Vehicular Technology (2025).

[56] L. Ye, M. Figura, Y. Lin, M. Pal, P. Das, J. Liu, V. Gupta, Resilient multiagent reinforcement learning with function approximation, IEEE Transactions on Automatic Control 69 (12) (2024) 8497–8512.

[57] D. Ressi, R. Romanello, C. Piazza, S. Rossi, Ai-enhanced blockchain technology: A review of advancements and opportunities, Journal of Network and Computer Applications 225 (2024) 103858.