ICONHIC2026
4th International Conference on Natural Hazards & Infrastructure
29 June – 2 July 2026, Chania, Greece

Reinforcement Learning for Flood Recovery in Urban Rail Transportation Systems

Victor H. Calderon, M.S.*, Jürgen Hackl, Ph.D.
Princeton University, Princeton, USA

Wei Bi, Ph.D.
ETH Zürich, Zürich, Switzerland

Abstract

Climate resilience of infrastructure systems has become more relevant as climate change intensifies the frequency and severity of extreme weather events. Among these, heavy rains and subsequent flooding pose a significant risk to the functionality of transportation systems. Post-disaster recovery strategies seek to minimize service disruption and socio-economic impacts by efficiently allocating recovery resources. In this paper, we propose a reinforcement learning (RL) method for resource-constrained scheduling that enables complete service recovery of a partially flooded urban rail transportation system (URTS). The ability to distribute recovery resources in such a dynamic infrastructure system, where interdependence among components drives dynamic processes like passenger flow or damage propagation, in a timely and informed manner after a catastrophic event is important to minimize system downtimes, which may increase safety risks for passengers and lead to significant economic losses. In our approach, an RL agent is explicitly trained so that simulated extreme flooding events that cause large disruptions to the system will produce a good scheduling performance generalization on possible future flooding scenarios. We deploy the RL agent on a disrupted London URTS model and compare its scheduling performance with state-of-the-art heuristic methods in terms of solution effectiveness, computation time, and ability to generalize. The results highlight the potential of RL as a practical framework for the recovery of dynamic transportation systems, achieving advantages in scenario adaptability and solution accuracy compared to alternative methods.

Keywords: deep reinforcement learning, flood recovery, urban rail systems, infrastructure resilience

INTRODUCTION

Climate change effects on rainfall intensity can lead to larger pluvial flooding events that, at the same time, trigger cascading operational disruptions in metropolitan networks (ASCE, 2018), increasing the need for effective recovery planning. One of these networks is the urban rail transportation systems (URTS) where flooding incidents have occurred in cities such as New York (Impelli, 2021), Hong Kong (Mok et al., 2023), and London (London Assembly, 2021). One way to mitigate the post-disaster effects in a partially flooded UTRS is to deploy recovery teams efficiently so that flooded stations and tracks are repaired quickly, and service can be restored with minimal revenue impact. Building on prior work that adopts revenue loss as a performance measure aligned with the objectives of rail administration and operations stakeholders (Bi et al., 2024; Bi et al., 2025), we analyze London's URTS flood recovery as a decision-making problem within a deep reinforcement learning (DRL) framework.

Prior work on recovery resource scheduling in damages system has relied on static prioritization rules or heuristic dispatching rules. For example, several studies use static importance categorization based on network features (Bi et al., 2024; Candelieri et al., 2019), while others adopt heuristic optimization methods (Bi et al., 2025; Nurre et al., 2012; Tan et al., 2019). These approaches provided strong baselines, but typically do not

**Corresponding author:* Victor H. Calderon, Department of Civil and Environmental Engineering, Princeton University, Princeton, USA. Email: vhcalderon@princeton.edu

generalize rapidly to unseen damage scenarios and to large networks. More recently, DRL has emerged as a powerful family of methods for sequential decision optimization, and network resilience-oriented work has explored value-based methods such as Q-learning and Deep Q-learning (DQN) (Fan et al., 2023; Liang et al., 2025), as well as multi-agent RL formulations (Wang et al., 2024). However, these DRL applications have mostly been studied in small-sized networks, leaving many unresolved questions about the scalability of DRL methods in complex infrastructure networks. In contrast to well-documented, controlled settings such as Atari (Mnih et al., 2013) and MuJoCo (Todorov et al., 2012), the URTS recovery under flooding requires reasoning over high-dimensional graph observations and understanding how complex network features come into play when a DRL agent is trained and deployed in this setting.

In this paper, we study the effectiveness of a DRL agent in enhancing recovery team dispatch policies directly by interacting with the flooded network environment. We formulate URTS recovery under flooding as a semi-Markov decision process (SMDP) (Sutton et al., 1999) and train a single Proximal Policy Optimization (PPO) agent (Schulman et al., 2017) to learn the most efficient way to deploy recovery teams in an extreme 1000-year flood scenario, minimizing travel loss revenue while also reducing disrupted journeys. The training stability and extensive documentation of PPO make it a practical choice for this study and it can reach a new state of the art in implementing DRL in addressing complex-real-world infrastructure challenges posed by climate change.

METHODOLOGY

Background

The London URTS network model, data compilation and processing pipeline, and resilience assessment framework used in this study are based on the work of (Bi et al., 2024). In this framework, the infrastructure system is modelled as an undirected network $G = (\mathcal{V}, \mathcal{E})$, where nodes $v \in \mathcal{V}$ and edges $e \in \mathcal{E}$ represent the stations and tracks of the system, respectively, as can be seen in Fig. 1. The London URTS network considered in this study contains 443 nodes and 533 edges. Hourly ticket fares and travel demands between station pairs (OD-demands) are employed to compute revenue loss during the recovery process. This process is modeled as the deployment of flood emergency teams equipped with portable pumping stations to damaged elements, where the repair time depends on both element characteristics and flood depth. Disruption effects include station closures, which may still allow train passage depending on the station characteristics and flooding depth, as well as track closures and the suspension of adjacent tracks based on practical criteria. Flood hazard is modeled as a static event that causes the network disruption at the beginning of the recovery process only, and this work focuses primarily on the extreme 1000-year flood scenario.

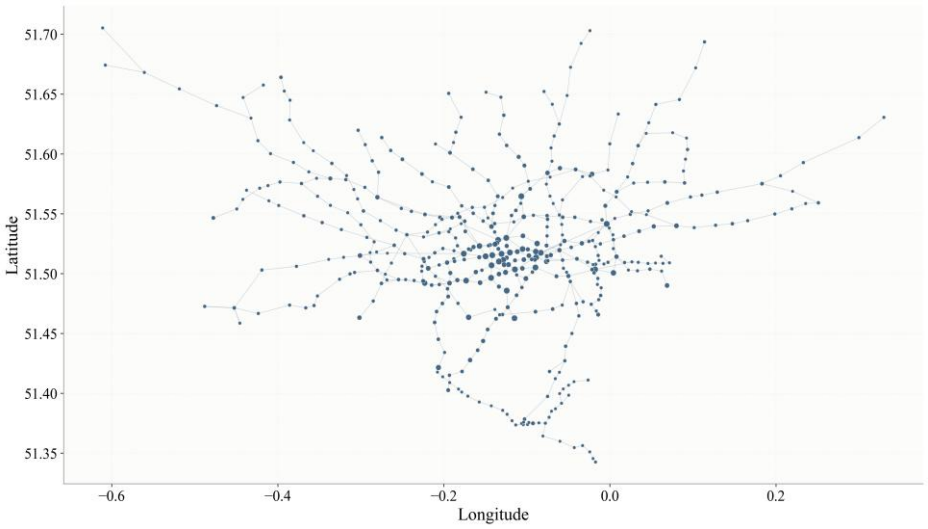

Figure 1. *London Urban Rail Transportation System (URTS)*

Reinforcement Learning

We model the URTS recovery under flooding as an SMDP where the infrastructure network changes only when a repair is completed. Let $k = 0,1,\dots,K-1$ be the index decision steps within an episode, t_k be the hour index in step k, and $\Delta t_k = t_{k+1} - t_k \geq 0$ be the variable holding time for the next step. The state $s \in \mathcal{S}$ combines a network snapshot and other relevant features: $s_k = (H, \boldsymbol{X}_k, \boldsymbol{g}_k)$, where H is the representation of the network and is kept constant in every episode, $\boldsymbol{X}_k$ is a token matrix that contains features of all nodes and edges of the network, and $\boldsymbol{g}_k$ denotes global features external to the network, such as time of day and number of idle teams. In our setup, the agent acts only when at least one team is idle; between steps, the topology and operability are constant. (Bi et al., 2024) determine that OD-demand and fares are deterministic at each time step and show a daily periodic pattern, thus conditioning on t_k suffices to make the corresponding hourly revenue drivers observable at decision time.

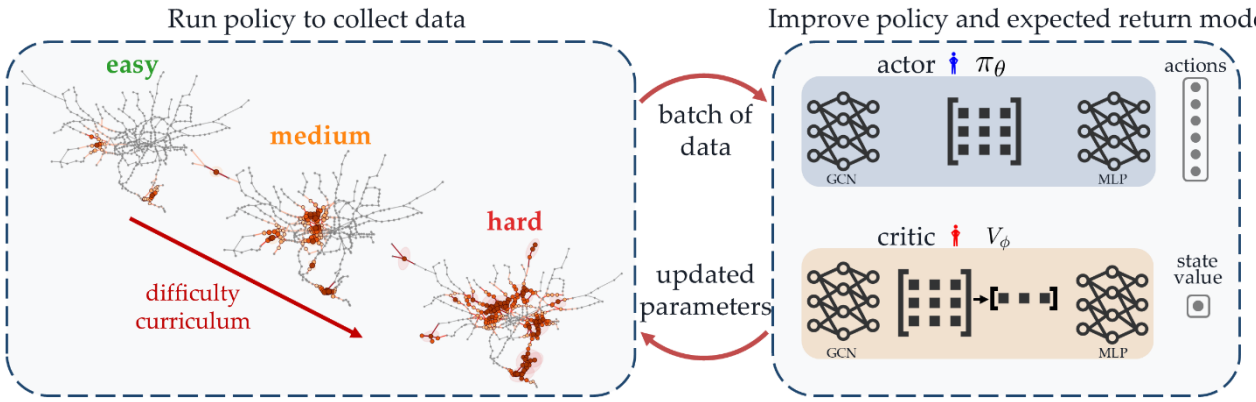

Figure 2. *PPO algorithm involves two major steps: 1) Run the policy to collect batch of data. The policy is first used to collect batches of trajectories from progressively harder recovery scenarios. 2) Improve the policy and expected return model. The batch of states and, which selects repair actions, and the critic, which estimates state value. The updated actor–critic model is iteratively reused to collect new experiences and improve scheduling performance across the curriculum.*

In any step, we define U_k as the number of idle teams and $\mathcal{C}_k$ the set of currently disrupted elements (composed of nodes and edges) under the action mask m_k^{act}. The action is a variable-length sequence $a_k = \left(a_k^{(1)}, \dots, a_k^{(U_k)}\right)$ produced autoregressively (i.e., it depends stochastically on its past values), where $a_k^{(j)} \in \mathcal{C}_k \cup \{\text{STOP}\}$ denotes the disrupted element assigned to the j-th idle recovery team, or the decision to stop assigning additional teams at that step. Therefore, the policy is defined as

$$\pi_\theta(a_k \mid s_k) = \prod_{j=1}^{U_k} \pi_\theta\left(a_k^{(j)} \mid s_k, a_k^{\text{past}}\right) = \prod_{j=1}^{U_k} \frac{\exp\left(\ell_{k,a_k^{(j)}}\right) m_{k,(j)}^{\text{act}}\left[a_k^{(j)}\right]}{\sum_{c'} \exp\left(\ell_{k,c'}\right) m_{k,(j)}^{\text{act}}[c']}. \quad (1)$$

where $\ell_{k,c}$ denotes the policy logit (score) assigned by the network to candidate element c, and $m_k{}^{\text{act}}$ restricts the softmax to feasible action choices. In this case, for instance, after selecting an element $c \in \mathcal{C}_k$ as $a_k^{(j)}$, the element becomes under repair by the j-th recovery team, then we set its mask to prevent duplicate assignment. Consequently, π_θ is the probability of allocating the available idle recovery teams to a feasible sequence of disrupted elements. STOP remains available in every slot, representing the possibility of choosing up to the number of idle teams at each step.

A simulator defines the state of the disrupted URTS, travel demand, and recovery team allocation at each decision step. The simulator is based on the flood resilience assessment framework of Bi et al. (2024), and

updates element recovery times, cascading effects, number of trips fulfilled, and other features relevant to recovery team allocation. Repair duration is integrated through an element-specific recovery timer, which evolves between decision steps and is shortened when a recovery team is assigned to the element. An element is classified as repaired when its recovery timer reaches zero and, for elements affected by cascading closures, when the element causing the cascading effect has also been restored.

The objective is to minimize the revenue loss caused by the flooding disruption. Thus, the step reward evaluates the revenue loss relative to the undamaged baseline hour-by-hour and then accumulates over the realized decision interval $[t_k, t_{k+1})$. For hour h, we define the baseline revenue $\mathrm{BR}(h)$ and the disrupted-system revenue $\mathrm{DR}(h\,; s_k)$ under the network in-repair state s_k. The value $\mathrm{DR}(h\,; s_k)$ is calculated as the sum of satisfied OD trips multiplied by the fare rate for each journey. A journey is considered satisfied if the origin and destination stations are not disrupted, a path exists between them, and the disrupted travel time does not exceed the original travel time by more than 30 minutes. Since the system remains constant within $[t_k, t_{k+1})$, $\mathrm{DR}(h; s_k)$ can be evaluated at hourly granularity using the deterministic OD-demand and fares indexed by h (with feasibility of travel-time and costs captured in $\boldsymbol{X}_k$). Formally, the step reward R_k is the negative normalized revenue loss integrated over the step:

$$\tilde{r}(h) = -\frac{\mathrm{BR}(h) - \mathrm{DR}(h\,; s_k)}{C}, \qquad R_k = \sum_{h=t_k}^{t_{k+1}-1} \tilde{r}\,(h), \tag{2}$$

where $C > 0$ is constant to control the reward magnitude. This formulation removes the necessity of "hourly decisions and rewards" that can cause sparsity in the reward signals while preserving hourly granularity in the necessary features.

Proximal Policy Optimization agent

The implemented PPO algorithm has a similar architecture to the one proposed by (Petrenko et al., 2020), in which the algorithm trains a stochastic policy $\pi_\theta(\cdot \mid s)$ (called actor) and a value function $V_\phi(s)$ (called critic) (Mnih et al., 2016; Sutton & Barto, 2020). The critic reduces the variance by providing an advantage estimate A_k that quantifies the long-term advantage of taking an action. The actor is the policy rollout to explore the environment, which in this case is how the loss revenue is affected by the performed action. On top of this, PPO uses a conservative trust-region-like surrogate objective function that clips the ratios between the current policy and the policy before the update. Because decision steps have variable time intervals, the discount factor γ is implemented at the interval level through $\Gamma_k = \gamma^{\Delta t_k}$, and computes the generalized advantage estimation (GAE) as

$$\delta_k = R_k + (1 - d_k)\Gamma_k V_\phi(s_{k+1}) - V_\phi(s_k), \qquad A_k = \delta_k + (1 - d_k)\lambda\Gamma_k A_{k+1}, \tag{3}$$

with termination flag d_k, representing when the agent reaches a terminal state (e.g., there are no flooded elements left), and $\lambda\Gamma$ is the discount of the rewards. PPO then optimizes a clipped surrogate objective that limits how much the new policy π_θ can deviate from the previous policy $\pi_{\theta_{old}}$. This is done by clipping the probability ratio between the two policies, yielding the objective

$$L_{\text{clip}}(\theta) = \mathbb{E}_k[\min(r_k(\theta)A_k,\ \text{clip}(r_k(\theta), 1-\epsilon, 1+\epsilon)A_k)], \quad r_k(\theta) = \frac{\pi_\theta(a_k \mid s_k)}{\pi_{\theta_{\text{old}}}(a_k \mid s_k)}. \tag{4}$$

In practice, we include standard auxiliary terms such as the value regression loss $\| V_\phi(s_k) - \hat{V}_k \|^2$ and entropy bonus $B(\pi_\theta(\cdot \mid s_k))$ (Mnih et al., 2016). Fig. 2 provides a schematic of the PPO framework used in this study.

Model Architecture

The disrupted network s_k is represented using mask-based features instead of removing nodes or edges. This approach ensures fixed input dimensions (tensor shapes) for the neural network across episodes and eliminates the need for repeated network encoding during training. Each disrupted element, including nodes and edges, is assigned a feature vector comprising: (i) element status (flooded or operational), (ii) operability and under-repair indicators, (iii) flood depth and deterministic repair duration, (iv) static priors such as nominal centrality, and (v) a type feature that distinguishes nodes from edges. The encoder is a graph convolutional network (GCN) (Kipf & Welling, 2017) implemented with GraphSAGE (Hamilton et al., 2018).

We perform an architecture sweep in which the GraphSAGE backbone and PPO hyperparameters are held constant, while varying several model design factors: (i) embedding capacity (hidden units of 64 versus 128), (ii) neural network depth (2, 3, or 4 layers), (iii) use of separate encoders for actor and critic, (iv) dropout regularization (0.0 versus 0.1), (v) an operability gate that attenuates messages from non-traversable components (damaged elements of the URTS cannot transmit messages), (vi) residual (skip) connections across GCN layers. Additional ablations remove the operability gate and residual pathways (resulting in a "stripped" architecture), and variants are tested that normalized the value function during training to stabilize critic optimization.

The architecture sweep aims to identify model configurations capable of reliably learning restoration behaviors. The capacity and depth probes assess whether learning is limited by representational constraints. Dropout trials evaluate whether regularization mitigate "cherry-picking" strategies that prioritize a few high-revenue repairs while neglecting large portions of the network. Tests of the operability gate and residual connections examine whether explicitly encoding infrastructure availability into message passing and maintaining information flow across layers improve restoration and optimization stability.

Training Configuration, Curriculum, and Evaluation Case

Each architecture is trained using $7{,}680$ total environment steps, with a rollout length of 32, batch size of 32, and 4 epochs per update. Following the recommendations of (Schulman et al., 2017), the learning rate is set to 3×10^{-4} with a cosine learning-rate schedule, policy updates used a clipping coefficient of 0.2, and the value-function coefficient of 0.25. Gradients are clipped with a maximum norm of 500, and a constant of $C = 50{,}000$ is selected to the reward signal to stabilize training in the presence of large economic-loss signals. This preliminary training horizon is selected because earlier short-horizon sweeps were effective for hyperparameter selection but insufficient for model selection; several configurations that appeared promising in short runs either plateaued or collapsed under the extended training protocol.

Training employs a task-difficulty curriculum in which policies are initially exposed to simpler restoration tasks and then to progressively more challenging scenarios, facilitating learning in contexts where direct optimization on the most difficult cases may result in weak or unstable progress (Shi et al., 2023). Early stages involve smaller disruption levels and simpler repair sequences, while later stages address larger disruptions in the network. The final evaluation serves as a proxy for a 1000-year flood scenario, with approximately 35% of nodes and 65% of edges affected, resulting in a simulation with extended topology degradation.

Platforms and Deployment

Training was conducted on two CPU-based platforms: a remote workstation and the Princeton Adroit high-performance computing (HPC) environment. The remote workstation was equipped with an Intel Xeon w3-2435 processor, 8 cores, and 62GB of RAM, while the HPC system featured an Intel Skylake processor, 32 cores, and 375GB of RAM. Both platforms were included because CPU microarchitecture can result in different optimization trajectories, even when code and hyperparameters are nominally identical.

RESULTS

The proposed PPO-based recovery policy is evaluated on a fixed test case and compared with two baselines: (i) a betweenness-centrality-based rule and (ii) a random dispatch policy. Performance is summarized using cumulative revenue loss, solve time for terminated episodes, repairs per hour, and the distribution of repairs between nodes and edges. Under our evaluation protocol, an episode is considered solved only if restoration is completed before the 168-hour simulation horizon; otherwise, it is truncated.

Not all implemented architectures demonstrated learning capability during training. Initial short-budget training sweeps showed that architectures with lower hidden units, such as 64, failed to learn, and more complex architectures could also fail under limited training cycles. Therefore, comparisons were conducted only after confirming that an architecture could learn. The two strongest configurations identified were the *shared actor-critic encoder* (arch_h128_l3_stripped) and the *value-function-stabilized model with separate actor-critic encoders* (vf_separate_norm_stripped). Both utilize a 128-dimensional, 3-layer GNN actor without operability gating, residual connections, or dropout.

The results in Fig. 3 indicate that the learned policies of both leading architecture configurations are competitive with the betweenness and random baselines. The betweenness baseline solves the evaluation case in 111.8 h of simulated time with a cumulative revenue loss of £38.7 million, while the random baseline achieves 108.0 h and £38.2 million. In comparison, arch_h128_l3_stripped achieves a solve time of 111.4 h, and a cumulative revenue loss of £35.9 million. The value function stabilized model, vf_separate_norm_stripped, further improves solve time to 98.3 h, maintains a comparable cumulative loss of £36.3 million, and ensures critic stability. These preliminary results suggest that the stabilized stripped model is an effective solution in the current study, as it preserves restoration reliability and enhances the trustworthiness of the learning dynamics through a critic that reliably estimates value functions and thereby guides the policy toward better actions.

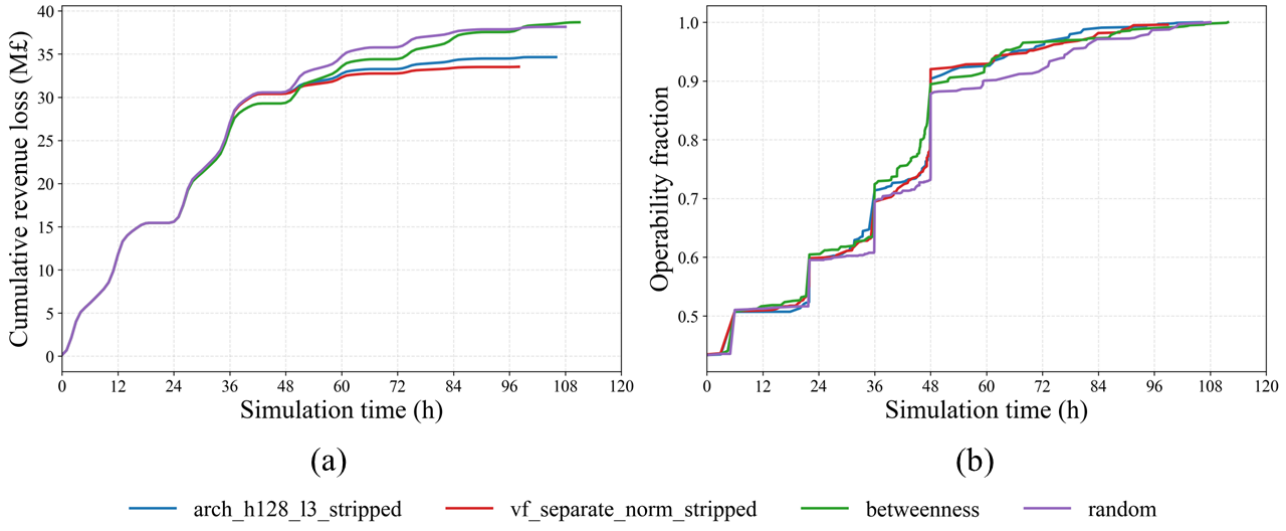

Figure 3. (a) Cumulative revenue loss and, (b) network operability fraction over the restoration horizon for the two best learned architectures and the baselines.

Different development of operability trajectories over simulation time in learned policies reveals distinct recovery strategies compared to the baselines. During the early stages of recovery, Fig. 3a and Fig. 3b show relatively small differences among the methods because the network-wide disruption initially results in long recovery times for many affected elements. As the consequences of recovery decisions unfold, particularly after approximately 36~h, differences in strategy performance become more apparent. The betweenness and random baselines show a relatively monotonic increase in operability, reflecting repair sequences determined by static topological priorities. The learned policies, in contrast, exhibit operational improvements during the

middle stages of recovery, suggesting that the policies can identify beneficial recovery actions beyond fixed topological rules.

The stepwise increases in operability, especially before 48 h, are partly caused by the natural recession of flooding in elements where recovery teams were not assigned. After approximately 48~h, most remaining flooded elements require active restoration by recovery teams because natural recession is not applicable due to physical constraints, e.g., enclosed underground tunnels.

As shown by the additional performance metrics in Fig. 4, the learned policies do not simply imitate the heuristics but instead differ from them. In the evaluation case, the learned policies prioritize nodes, with an edge-to-total repaired elements ratio of approximately 0.50 for vf_separate_norm_stripped and 0.66 for arch_h128_l3_stripped, compared to the betweenness baseline, which emphasizes restoration of topologically central edges (around 0.86). This difference suggests that the learned policy has identified an effective node-first strategy that alleviates certain recovery bottlenecks. Furthermore, this strategy appears to be more efficient in resource scheduling: although it results in fewer repairs per hour, it also reduces the number of decision steps required to restore the complete system (from 104 to 80 in the case of vf_separate_norm_stripped). This indicates a policy that identifies critical elements for network restoration, thereby requiring fewer resources to achieve complete recovery.

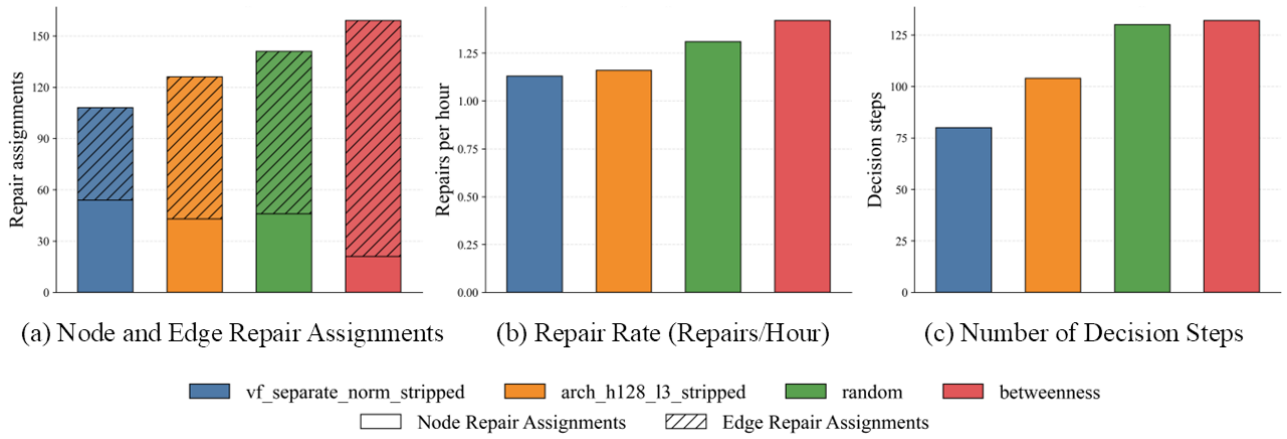

Figure 4. *Node and edge repair assignments, number of repairs per simulated hour, and number of decision steps.*

An additional observation is that policy training is not fully platform-invariant. Although inference solution times were similar (less than 5 minutes) for the evaluation case, workstation training runs were approximately twice as fast per PPO update as Adroit CPU runs (5 hours compared to approximately 10 hours per model). However, this computational advantage did not guarantee improved policy quality. Differences in results between architecture configurations trained on different platforms are attributed to the well-known numerical sensitivity of deep reinforcement learning training (Munikoti et al., 2022), where small variations in floating-point accumulation, threading, or library implementations can lead to divergent optimization trajectories. Therefore, future experiments should treat the training platform as an experimental factor when assessing robustness.

CONCLUSIONS

This paper formulated post-flood recovery in an URTS as a semi-Markov decision process and demonstrated that deep reinforcement learning can serve as a practical framework for resource-constrained scheduling. Using a PPO agent with graph-structured observations and an autoregressive action formulation, we showed that learned scheduling policies can produce reliable full-restoration behavior in an extreme-flood proxy case and achieve performance that is competitive with topology-based methods on key resilience metrics. Under the modeling assumptions adopted in this study, inference is computationally lightweight relative to the simulation

horizon, suggesting feasibility for real-time decision support once trained. The results suggest the following: (i) learned policies may converge to repair strategies distinct from those of topology-based methods, such as node-heavy prioritization and fewer decision steps, which indicates that a deep reinforcement learning (DRL) agent can identify novel scheduling strategies optimal for network-flooding recovery; (ii) although different proximal policy optimization (PPO) implementations exhibit varying performance, their policy effectiveness is strongly influenced by the stability of critic learning.

This study, together with related efforts to transfer deep reinforcement learning (DRL) to infrastructure systems, aims to advance the design of disaster-resilient systems. Traditional resilience analysis methods are constrained by limited data on rare extreme events and the complexity of large decision spaces. DRL and other machine learning approaches offer formal alternatives, such as scenario generation, data augmentation, and pattern recognition, to address these challenges and enable more adaptive recovery strategies as hazards and priorities change. Specifically in complex infrastructure networks which relational data and complex dynamics analyses can be leveraged using network representation learning techniques.

Future work will strengthen the empirical and methodological basis for deployment. First, we plan to expand evaluation beyond the fixed proxy case by running real flooding scenarios and then comparing against a more robust heuristic and optimization baselines. Second, we plan to improve robustness by increasing the number of training cycles while randomize disruption realizations, and performing PPO fine-tuning.

The proposed PPO methodology is generally applicable to other rail transportation systems with different sizes and layouts. However, it should be noted that its applicability comes at the cost of the flood-risk assessment assumptions and computational costs. Assumptions such as the relationship between repair time and flood depth, cascading closure rules, and the time required to relocate repair teams may vary across systems and contexts. Additionally, the effectiveness of PPO depends on the agent-environment sampling budget, producing a more generalized model only when a large computation budget is available.

REFERENCES

ASCE. (2018). *Climate-Resilient Infrastructure: Adaptive Design and Risk Management* (B. M. Ayyub, Ed.). American Society of Civil Engineers. https://doi.org/10.1061/9780784415191

Bi, W., Hackl, J., & MacAskill, K. (2025). Enhancing flood resilience of urban rail transit systems through recovery resource scheduling optimisation: A case study of London. *Sustainable Cities and Society*, *128*, 106437. https://doi.org/10.1016/j.scs.2025.106437

Bi, W., Schooling, J., & MacAskill, K. (2024). Assessing flood resilience of urban rail transit systems: Complex network modelling and stress testing in a case study of London. *Transportation Research Part D: Transport and Environment*, *134*, 104263. https://doi.org/10.1016/j.trd.2024.104263

Candelieri, A., Galuzzi, B. G., Giordani, I., & Archetti, F. (2019). Vulnerability of public transportation networks against directed attacks and cascading failures. *Public Transport*, *11*(1), 27–49. https://doi.org/10.1007/s12469-018-00193-7

Fan, X., Zhang, X., Wang, X., & Yu, X. (2023). A deep reinforcement learning model for resilient road network recovery under earthquake or flooding hazards. *Journal of Infrastructure Preservation and Resilience*, *4*(1), 8. https://doi.org/10.1186/s43065-023-00072-x

Hamilton, W. L., Ying, R., & Leskovec, J. (2018). *Inductive Representation Learning on Large Graphs*. https://doi.org/10.48550/arXiv.1706.02216

Impelli, M. (2021). Ida Sent 75 Million Gallons of Water Into NY Subway System, Caused $75M in Damages. *Newsweek*. https://www.newsweek.com/ida-sent-75-million-gallons-water-ny-subway-system-caused-75m-damages-1629826

Kipf, T. N., & Welling, M. (2017). *Semi-Supervised Classification with Graph Convolutional Networks*. https://doi.org/10.48550/arXiv.1609.02907

Liang, H., Moya, B., Chinesta, F., & Chatzi, E. (2025). *Resilience-based post disaster recovery optimization for infrastructure system via Deep Reinforcement Learning*. https://doi.org/10.48550/arXiv.2410.18577

London Assembly. (2021). *News from Zack Polanski: Stations closed for 141 hours this year after flooding*. https://www.london.gov.uk/press-releases/assembly/zack-polanski/stations-closed-for-141-hours-after-flooding

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). *Asynchronous Methods for Deep Reinforcement Learning*. https://doi.org/10.48550/arXiv.1602.01783

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). *Playing Atari with Deep Reinforcement Learning*. https://doi.org/10.48550/arXiv.1312.5602

Mok, D., Kong, H., & Tsang, D. (2023). 132 hongkongers sent to hospitals, all rainstorm alerts cancelled after deluge. *South China Morning Post*. https://www.scmp.com/news/hong-kong/health-environment/article/3233782/hong-kong-observatory-issues-red-rainstorm-alert-warns-significant-road-flooding

Munikoti, S., Agarwal, D., Das, L., Halappanavar, M., & Natarajan, B. (2022). *Challenges and Opportunities in Deep Reinforcement Learning with Graph Neural Networks: A Comprehensive review of Algorithms and Applications*. https://doi.org/10.48550/arXiv.2206.07922

Nurre, S. G., Cavdaroglu, B., Mitchell, J. E., Sharkey, T. C., & Wallace, W. A. (2012). Restoring infrastructure systems: An integrated network design and scheduling (INDS) problem. *European Journal of Operational Research*, *223*(3), 794–806. https://doi.org/10.1016/j.ejor.2012.07.010

Petrenko, A., Huang, Z., Kumar, T., Sukhatme, G., & Koltun, V. (2020). *Sample Factory: Egocentric 3D Control from Pixels at 100000 FPS with Asynchronous Reinforcement Learning*. https://doi.org/10.48550/arXiv.2006.11751

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. https://doi.org/10.48550/arXiv.1707.06347

Shi, L. X., Jiang, Y., Grigsby, J., Fan, L. "Jim"., & Zhu, Y. (2023). *Cross-Episodic Curriculum for Transformer Agents*. https://doi.org/10.48550/arXiv.2310.08549

Sutton, R. S., & Barto, A. (2020). *Reinforcement learning: An introduction* (Second edition). The MIT Press.

Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*(1–2), 181–211. https://doi.org/10.1016/S0004-3702(99)00052-1

Tan, Y., Qiu, F., Das, A. K., Kirschen, D. S., Arabshahi, P., & Wang, J. (2019). Scheduling Post-Disaster Repairs in Electricity Distribution Networks. *IEEE Transactions on Power Systems*, *34*(4), 2611–2621. https://doi.org/10.1109/TPWRS.2019.2898966

Todorov, E., Erez, T., & Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

Wang, Y., Qiu, D., Teng, F., & Strbac, G. (2024). Towards Microgrid Resilience Enhancement via Mobile Power Sources and Repair Crews: A Multi-Agent Reinforcement Learning Approach. *IEEE*

Transactions on Power Systems, *39*(1), 1329–1345. https://doi.org/10.1109/TPWRS.2023.3240479