17th World Conference on Earthquake Engineering, 17WCEE
Sendai, Japan - September 13th to 18th 2020

OPTIMIZING POST-EARTHQUAKE FUNCTIONAL RESTORATION OF WATER DISTRIBUTION SYSTEMS THROUGH DEEP REINFORCEMENT LEARNING

H. Huang(1), H. V. Burton(2)

(1) *Ph.D. Candidate, Department of Civil and Environmental Engineering, University of California, Los Angeles, CA, 90095, USA, honglanhuang@ucla.edu*
(2) *Assistant Professor, Department of Civil and Environmental Engineering, University of California, Los Angeles, CA, 90095, USA, hvburton@ucla.edu*

Abstract

Following a major seismic event, multiple components within a water distribution system can undergo different levels of damage or inoperability, causing significant disruption to the water supply. Repair scheduling plays an important role in restoring the functionality of water distribution systems in a timely manner. To this end, this study formulates the post-earthquake hydraulic restoration process of a water distribution system as a Markov Decision Process, and implements deep reinforcement learning to determine the optimal repair policy. The proposed framework incorporates a simulation-based environment (pipeline damage and hydraulic serviceability) and implements the Deep Q Network and Deep Actor Critic reinforcement learning agents. Simulation-based test results on a wide range of damage cases show that the deep reinforcement learning agents outperform the basic random assignment policy by reducing the time to restore hydraulic serviceability within the water distribution network. As one of the first applications of the deep reinforcement learning methodology to optimize post-earthquake restoration of water distribution systems, this study demonstrates its capability to guide decision-making and enhance the resilience of critical infrastructure.

Keywords: water distribution system; deep reinforcement learning; post-earthquake restoration; resilience

1. Introduction

Water distribution networks (WN) have been identified as one of several critical infrastructure systems (CIS) [1]. Following a major seismic event, the physical damage to a water distribution network can cause significant disruption to its core functions (e.g. delivering water to residential commercial and industrial facilities), resulting in adverse socioeconomic impacts to the affected population [2–4]. The growing societal expectations towards infrastructure resilience urges for better post-hazard-event restoration policies [5]. Therefore, developing reliable frameworks and tools to support optimal decision making for the post-earthquake recovery of water distribution systems is one of the critical steps towards enhancing infrastructure resilience.

The post-earthquake recovery of CIS usually involves several tasks: inspection, damage assessment and repair. The repair task usually takes the longest duration and requires significant allocation of resources [6]. In the literature, the repair scheduling problem for CIS typically takes on one of two formulations. The first utilizes linear programming, whereby the flow operation equations are embedded within the optimization constraints and an approximate objective function is usually adopted [6, 7]. Another approach is to formulate the repair scheduling as a Markov decision process (MDP), which is a general framework for sequential decision making problems. Numerous studies have demonstrated the suitability of MDP for modeling infrastructure decision making problems and the capability of dynamic programming/ reinforcement learning methods to achieve a solution [5, 8–10]. However, the classical dynamic programming/reinforcement learning methods often suffer from the curse of dimensionality [8, 11, 12].

Recently, the classical reinforcement learning methods achieved significant breakthrough after being combined with deep learning methods, which is described as deep reinforcement learning (DRL). The DRL methodology has accelerated the progress of reinforcement learning agents in large-scale complex decision making problems, e.g., the games of Go [13] and StarCraft II [14]. Current applications of DRL in civil engineering are mostly on the maintenance and operation management of infrastructure systems [15, 16]. This study serves as one of the initial attempts to apply the DRL methodology to the post-earthquake restoration of water distribution systems.

2. Review of the Markov Decision Process and Reinforcement Learning Methodology

2.1 Markov decision process

The Markov decision process (MDP) is a mathematical framework that is used to represent a random process where a decision maker (agent) interacts with an environment at finite time steps $t = 0,1,2\ldots,n$ [11]. The finite MDP is characterized by following elements: $\mathcal{S}$ is the finite set of states; $\mathcal{A}$ is the finite set of actions; $\mathcal{P}:\mathcal{S}\times\mathcal{S}\times\mathcal{A}\rightarrow\left[0,1\right]$ is the state transition matrix with each entry $P(s_{t+1}=s' \mid s_t = s, a_t = a)$ defining the probability of transitioning from s to s' after taking action a; $\mathcal{R}$ is the reward function, $r\left(s,a\right)=E[r_{t+1} \mid s_t = s, a_t = a]$ represents the expected reward received after taking action a at state s, which can be stochastic or deterministic; γ is the discount factor for controlling the decay of importance for future rewards. At each time step t, the agent observes the current state s_t, takes action a_t, receives a reward r_{t+1} from the environment, which then transitions to next state s_{t+1}. The process from an initial to a terminal state is called an episode.

In the MDP, the goal is to learn the policy π, which maps states to a probabilistic distribution over the actions: $\pi(a \mid s)$ denotes the probability of choosing action a at state s. Defining the return function $G_t = \sum_{k=0}^{T-t}\gamma^k r_{t+k+1}$ as the total discounted reward from time step t to the time step T at the terminal state, the goal of the agent is to maximize the expected return [11]:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} E_{s\sim p, a\sim\pi}\left[G_0\right] \tag{1}$$

The other core component of the MDP is the value function. Specifically, the state value function $V^\pi(s)$ is defined as the expected return starting from state s and following π onwards:

$$V^\pi(s) = E_\pi[G_t \mid s_t = s] \tag{2}$$

Similarly, the state-action value function $Q^\pi(s,a)$ is the expected return after executing action a at state s and following π:

$$Q^\pi(s,a) = E_\pi[G_t \mid s_t = s,\ a_t = a] \tag{3}$$

2.2 Reinforcement learning

The value functions have a recursive property stated by the Bellman equation [11, 17]:

$$V^\pi(s) = E_\pi[r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s] \tag{4}$$

When the dynamic model of the MDP is known, i.e., the transition matrix $\mathcal{P}$ and reward function $\mathcal{R}$ are explicitly determined, then it can be solved by dynamic programming. When the properties ($\mathcal{P}$, $\mathcal{R}$) are unavailable, the solution needs to be obtained by letting the agent learn from interacting with the environment, often referred to as model-free reinforcement learning. There are two general paradigms: value-based methods and policy-based methods.

The value-based methods focus on estimating the value functions introduced in Eqs. 2 and 3. Common methods for estimating value functions include Monte Carlo (MC) and Temporal Difference (TD) methods. The key idea behind the MC method is using the empirical return to approximate the expected return [9, 11]:

$$\hat{Q}^\pi(s,a) \leftarrow \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} \left[G_{t_i} | s_{i,t_i} = s, a_{i,t_i} = a \right] \tag{5}$$

where N_{MC} is the number of generated episodes, t_i is the time step in the i^{th} episode when $s_{i,t_i} = s, a_{i,t_i} = a$ occurs for the first time, G_{t_i} is the return starting from (s_{i,t_i}, a_{i,t_i}). The policy is then updated immediately based on the current estimate of the state-action value. High variance is associated with the MC estimate [12]. TD methods do not generate samples of complete episodes, instead, the estimate $\hat{Q}^\pi(s_t, a_t)$ is updated towards the “target” Q value

$$\hat{y} = r_{t+1}(s_t, a_t) + \gamma \max_{a'} \hat{Q}^\pi(s_{t+1}, a') \tag{6}$$

which can be constructed immediately after executing $a_t \sim \pi$, thus $\delta = \hat{y} - \hat{Q}^\pi(s_t, a_t)$ is called the TD error. The update is given by

$$\hat{Q}^\pi(s_t, a_t) \leftarrow \hat{Q}^\pi(s_t, a_t) + \alpha\delta \tag{7}$$

where α denotes the step size. Eq. 6 and 7 are essentially the value function update steps in the Q-learning algorithm [18]. The TD method embeds less variance, but high bias is introduced by bootstrapping, which is likely to cause instability in the learning process [12].

In contrast to the value-based methods like Q-learning, the policy-based methods express the policy explicitly as a parametric distribution $\pi_\theta(a \mid s)$, and the agent takes actions by sampling from $\pi_\theta(a \mid s)$. The objective can be expressed as a function of θ [19]:

$$J(\theta) = E_{\pi_\theta}\left[G_0\right] \tag{8}$$

Taking the derivative with respect to θ produces the policy gradient [19]:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right] \tag{9}$$

Then θ is updated through:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \tag{10}$$

Policy-based methods have better convergence properties as the policy is parameterized. However, high variance is imposed [12], as the value needs to be estimated by the MC methods discussed earlier.

2.3 Deep reinforcement learning

The Q-learning algorithm and many other classical reinforcement learning methods need to store the value for all possible state-action pairs in a tabular form. Hence, the storage requirement grows exponentially with the expansion of the state and action spaces. Minh et al. [20] introduced the Deep Q Network (DQN) algorithm, which is a benchmark for embedding deep neural networks as function approximators for the reinforcement learning agents. Fig. 1 shows the schematic structure for the DQN model, where the solid lines represents the interaction process, and the dashed lines represent the learning process. The DQN improves upon the Q-learning method in the following main aspects:

1. Using deep neural networks to approximate the Q function: a Q network is used to approximate the Q value $Q_\theta(s,a)$ that guides the selection of actions, while a target Q network $Q_{\theta^-}(s',a')$ is used to approximately construct the "target" Q value for learning [12, 20]:

$$\hat{y} = r(s,a) + \gamma \underset{a'}{\operatorname{argmax}} Q_{\theta^-}(s',a') \tag{11}$$

The deep neural network enables processing high dimensional states such as the raw pixels of video games, and outputs the values $Q_\theta(s,a)$ for all available actions a at state s via a forward pass, as shown in Fig. 1. Moreover,only the parameters of the neural networks need to be saved and updated. The parameters θ^- of the target network are usually inherited from the Q network at delayed steps, which provides a more stable target.

2. Experience replay: a replay memory buffer, as shown in Fig. 1, is used to store past transitions (s,a,r,s') with storage capacity N, i.e., new transitions are taken in and old transitions are expelled. At each learning step, the agent randomly samples a mini-batch, $B=\{(s_i,a_i,r_i,s_i^{'})\}$, from the replay memory buffer and updates θ through stochastic gradient descent on the loss function [20]:

$$L(\theta) = \frac{1}{K}\sum_{i=1}^{K}\left(y_i - Q_\theta(s_i,a_i)\right)^2 = \frac{1}{K}\sum_{i=1}^{K}\left(r_i + \gamma \underset{a'}{\operatorname{argmax}} Q_{\theta^-}(s_i^{'},a_i^{'}) - Q_\theta(s_i,a_i)\right)^2 \tag{12}$$

where K denotes the batch size. This modification reduces the correlation of samples used for computing the TD error, thus reducing the bias in the update.

The Deep Actor Critic (DAC) establishes a trading-off between the bias introduced in the value-based methods and the variance in policy-based methods [12, 21]. The model consists of the two main components shown in Fig. 2: an actor network that approximates the policy function $\pi_{\theta_a}(a\,|\,s)$ and takes action by sampling from the policy; a critic network that approximates the value function $V_{\theta_c}(s)$ and evaluates the advantage of the action taken by the actor. The actor is updated through the policy gradient with respect to θ_a:

$$\theta_a \leftarrow \theta_a + \alpha_a \nabla_{\theta_a} \log \pi_{\theta_a}(a|s) A^{\pi_{\theta_a}}(s,a) \tag{13}$$

where $A^{\pi_{\theta_a}}(s,a) = Q^{\pi_{\theta_a}}(s,a) - V^{\pi_{\theta_a}}(s)$ is the advantage function, which measures the advantage of taking action a with respect to the average value of state s. Note that the advantage function can be estimated by

$\hat{A}^{\pi_{\theta_a}}(s,a) = r(s,a) + \gamma V_{\theta_c}^{\pi_{\theta_a}}(s') - V_{\theta_c}^{\pi_{\theta_a}}(s)$, where $V_{\theta_c}^{\pi_{\theta_a}}(s)$ and $V_{\theta_c}^{\pi_{\theta_a}}(s')$ are both estimates by the critic, as shown in Fig. 2.

The critic is updated through the gradient of the squared TD error with respect to θ_c:

$$\theta_c \leftarrow \theta_c + \alpha_c \delta \nabla_{\theta_c} V_{\theta_c}(s) \tag{14}$$

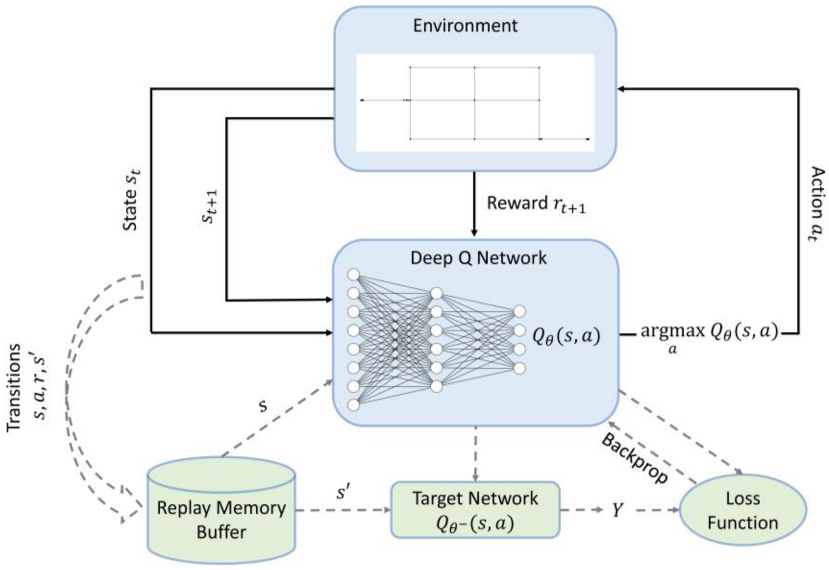

Fig. 1 – Schematic representation of the Deep Q Network

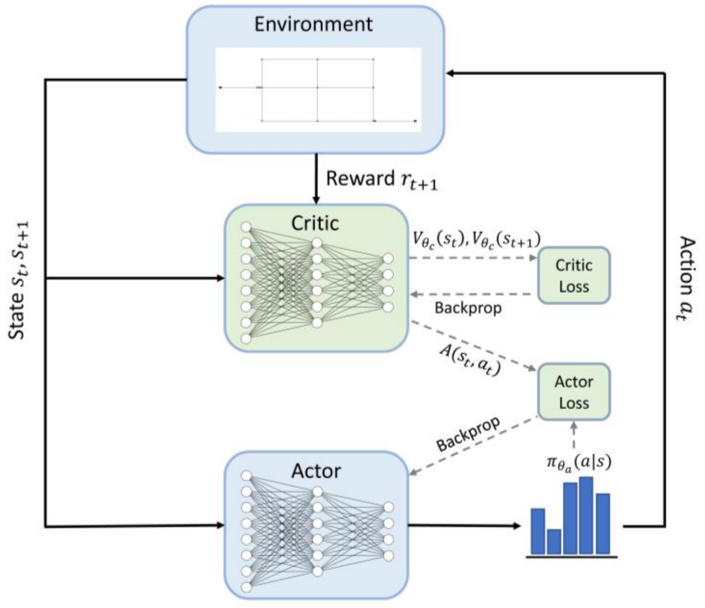

Fig. 2 – Schematic representation of the Deep Actor Critic

3. Problem Formulation and Model Implementation

3.1 Problem formulation

Following an earthquake scenario, multiple components in a water distribution network can undergo different levels of damage. The focus of this paper is on the damage to (and repair of) the pipelines in a water distribution network. More specifically, a system with N junctions and L pipelines is considered. In the formulation, $t = 0,1,2\ldots,n$ denotes the time steps of the Markov Decision Process, and $\tilde{t} \geq 0$ is the real time points (in days) following an earthquake. The goal is to find the repair scheduling policy that minimizes the cumulative loss of serviceability from the time immediately after the earthquake ($\tilde{t}_0$) to when the system recovers to full functionality ($\tilde{t}_{end}$), as shown in Fig. 3. In this paper, the System Serviceability Index (SSI) (ratio between the actual water supply to a junction to the expected demand) is adopted as the resilience metric. For the entire system, the SSI at time $\tilde{t}$ is determined by [4]:

$$SSI\left(\tilde{t}\right) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{L} q_{i,j}\left(\tilde{t}\right)}{\sum_{i}^{N} d_i\left(\tilde{t}\right)} \tag{15}$$

where $q_{i,j}\left(\tilde{t}\right)$ is the actual flow of water from pipe j into junction i, $d_i\left(\tilde{t}\right)$ is the expected demand of junction i. The objective for the repair schedule can be expressed equivalently as maximizing the area under the restoration curve (shaded area shown in Fig. 3) [22]:

$$\int_{\tilde{t}_0}^{\tilde{t}_{end}} SSI\left(\tilde{t}\right) d\tilde{t} \tag{16}$$

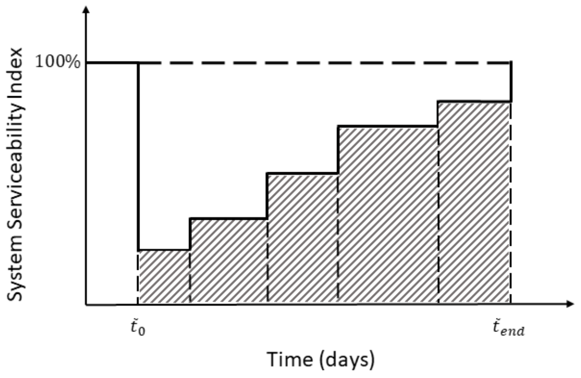

Fig. 3 – Schematic representation of a restoration curve for the system serviceability index

In this paper, two damage states are considered: DS_0 (Undamaged) and DS_1 (Major Leak) [4]. The MDP framework is adapted to model the restoration process of the water network as follows:

- State: $s_t = \left[s_{t1}, s_{t2}, \ldots, s_{tL}\right]$, where $s_{tj} \in \{0,1\}$, 0 corresponds to DS_0 and 1 corresponds to DS_1
- Action: $a_t = \left[a_{t1}, a_{t2}, \ldots, a_{tK}\right]$, where K denotes the number of available repair crews, $a_{tj} \in \{1,2,\ldots,L\}$ determines the corresponding pipe to be repaired.
- Reward: $r_{t+1}\left(s_t, a_t\right) = SSI\left(s_t, a_t\right) \cdot T\left(a_t\right)$, where $SSI\left(s_t, a_t\right)$ is the SSI after executing action a_t at state s_t (obtained from hydraulic simulation), and $T\left(a_t\right)$ denotes the duration of the repair action a_t. In this formulation, only two damage states are considered and the time needed to repair a pipeline from DS_1 to DS_0 is assumed to be the same for all teams. In future studies, this formulation can be modified to incorporate stochastic repair duration and multiple damage states.

The agent interacts with the environment in the following episodic process:

1. Episode start: the water distribution network is reset to the state s_0, which represents the initial damage case;

2. At each time step of the episode:

- The agent observes the water system state s_t and decides the next repair action a_t;
- The environment receives the repair action a_t, repairs the corresponding pipe, transitions to the next state s_{t+1} and obtains the $SSI\left(s_t, a_t\right)$ through hydraulic simulation, which is then transformed and passed to the agent as the reward r_{t+1};
- The agent learns and updates its policy.

3. Episode terminates: all the pipes are repaired and the serviceability is fully restored (100%).

3.2 Model implementation

In this study, the water distribution system model is constructed using the Python-based Water Network Tool for Resilience (WNTR) package [23]. The damage state DS_1 (Major Leak) is realized by creating a leak area with a diameter that is 90% of the pipe diameter [4]. The water network environment, "WaterNet-v0", is implemented as a subclass of the OpenAI gym environments [24], which is a platform for developing and testing reinforcement learning models. The DRL models discussed in section 2.3 are implemented in Python using the TensorFlow framework [25]. The model structures are implemented as follows:

Actor network:

- Hidden layer 1: fully-connected layer with 128 units and leaky rectified linear unit activation
- Hidden layer 2: fully-connected layer with 128 units and leaky rectified linear unit activation
- Output layer: fully-connected layer with 14 units (corresponding to the number of possible actions) and softmax activation

Q and Critic networks:

- Hidden layer 1: fully-connected layer with 128 units and rectified linear unit activation
- Hidden layer 2: fully-connected layer with 128 units and rectified linear unit activation
- Output layer: fully-connected layer with 14 units for the Q network and a single unit for the Critic network and linear activation for both

A major challenge in the DRL implementation is that the action space $\mathcal{A}$ is not constant because after repairing each pipeline, the corresponding action becomes ineffective at the next time step. Two approaches are explored when implementing the models. For the DQN agent, no constraints are imposed on the action space at each time step. Instead a negative reward is introduced to punish the selected illegal action. For the DAC agent, a filter is added to the output distribution from the actor by removing the illegal actions and renormalizing the distribution by

$$\pi'\left(a|s\right) = \frac{\pi\left(a|s\right)}{\sum_{a'\in\mathcal{A}_L}\pi\left(a'|s\right)} \tag{17}$$

where $\pi'\left(a|s\right)$ is the adjusted legal policy, $\mathcal{A}_L$ is the legal action space at state *s*.

3.3 Framework

A general framework for implementing the proposed DRL methodology, which incorporates the seismic hazard model for obtaining the shaking intensity and initial damage is illustrated in Fig. 4. The main steps are as follows:

Step 1. Damage simulation: simulate a representative number of possible damage cases to the WN using a suite of earthquake scenarios.

Step 2. Model training: starting from each damage case as the initial state of the environment, let the DRL agent learn by interacting with the environment for multiple episodes.

Step 3. Obtain repair policy: after training is complete, given a damage case of interest, the model outputs the optimal repair sequence and the associated restoration curve.

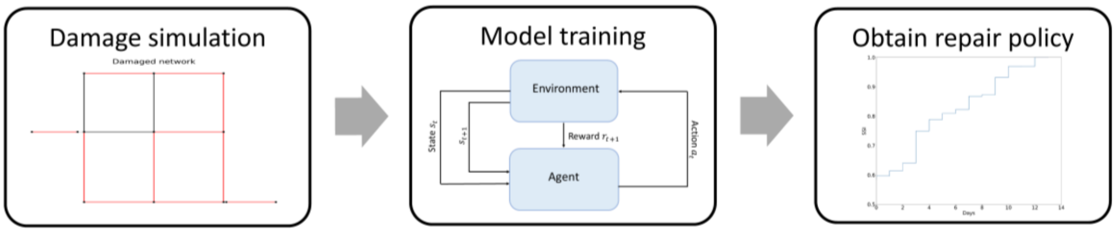

Fig. 4 – General framework for DRL model implementation

4. Case Study on a Hypothetical Water Network

In this section, a case study is performed where the DRL methodology is applied to the hypothetical water distribution network shown in Fig. 5, which has been used several prior studies [4, 26, 27].

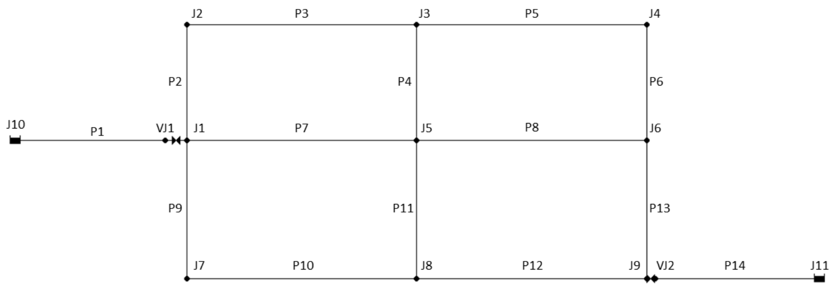

Fig. 5 – Hypothetical water distribution network (adapted from [4, 26, 27])

The hypothetical water distribution network consists of 14 distribution pipelines. Therefore, the number of possible damage cases is 2^{14}-1=16383. For this evaluation, a single repair team is assumed. random sample of damage cases is first generated, and the agents are trained using multiple episodes for each damage case. In this study, 1200 damage cases are used for the DQN agent and 1000 cases for the DAC agent. The number of episodes per case is set at 50. Once the training is completed, the performance of the agent is evaluated and compared with a "random" agent, i.e., an agent with no knowledge that selects a random pipeline to repair at each step. In this paper, the evaluation is performed on all possible (16383) cases. This evaluation does not require an earthquake ground motion simulation model. It simply compares the relative performances of the DRL agents and random agent starting from any initial damage state of the water distribution system.

4.1 Evaluation results

Fig. 6 presents the evaluation results for the DRL and random agents considering all possible cases, where the light grey lines are the restoration curves for individual cases, and the blue line presents an average restoration curve computed by averaging the required time to reach a given SSI level (at an increment of 0.01). It can be shown that the restoration curves following the repair policy by the DQN agent have much smaller dispersion and the area under the restoration curves are all greater than half of the area of full functionality, i.e., exhibits a "concave" shape. The restoration curves from the DAC agent exhibit slightly larger dispersion than those of the DQN, which is expected since the Actor outputs a distribution over actions and samples from the distribution, while DQN takes greedy actions with respect to the Q value. Nevertheless, the DAC has much better performance than the random agent.

A comparison of the average performances of the three agents is shown in Fig. 7. Fig. 7 (a) presents the result given by averaging the required time to reach a given SSI level. On average, the performance of the DAC is similar to DQN. The average time required to recover to full serviceability following the policy from the DRL agents are approximately 5 days, while 6 days are needed on average when following the random agent policy. Despite the small difference in terms of absolute values (because of the small network), the average recovery time for the random agent is approximately higher than the DRL agents. Fig. 7 (b) presents the restoration curves computed by averaging the SSI at specific time-points after damage has occurred. The average SSIs achieved following the DQN and DAC are much higher than the random agent case. It is worth noting that, since this is a small network, the "worst" restoration curve for any damage case is not expected to be much different from the "best" one.

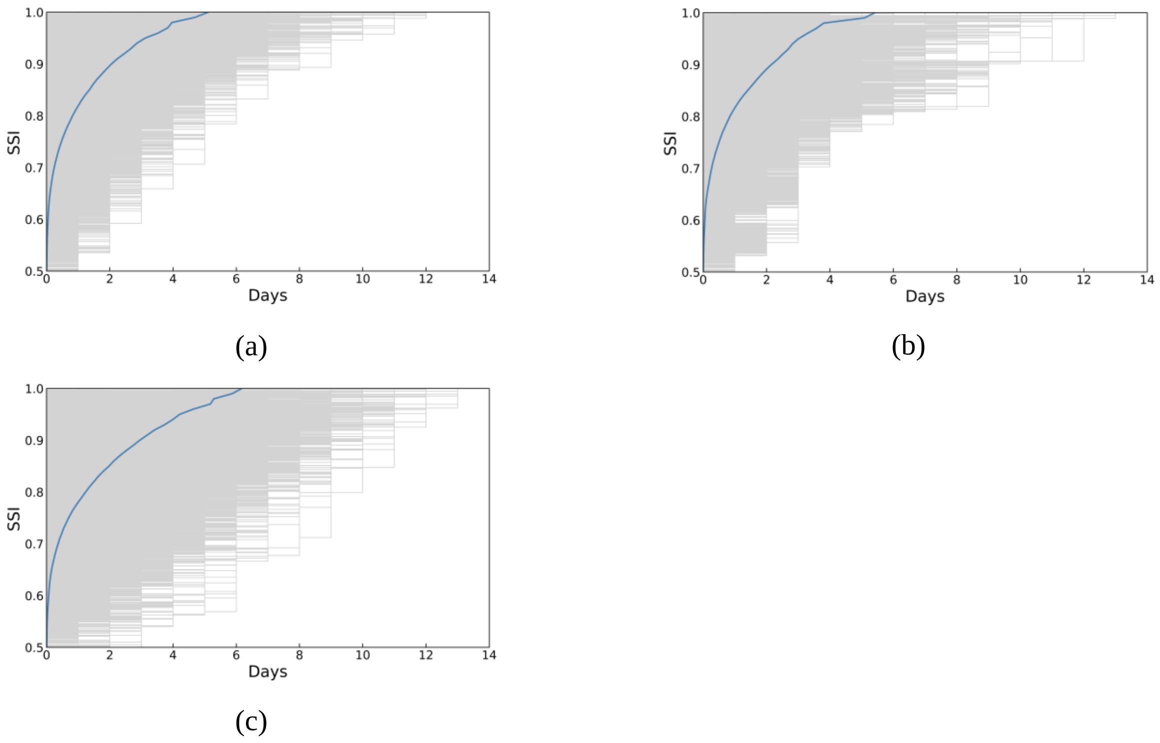

Fig. 6 – SSI-based restoration curves for all possible damage cases: (a) Deep Q Network; (b) Deep Actor Critic; and (c) Random Agent

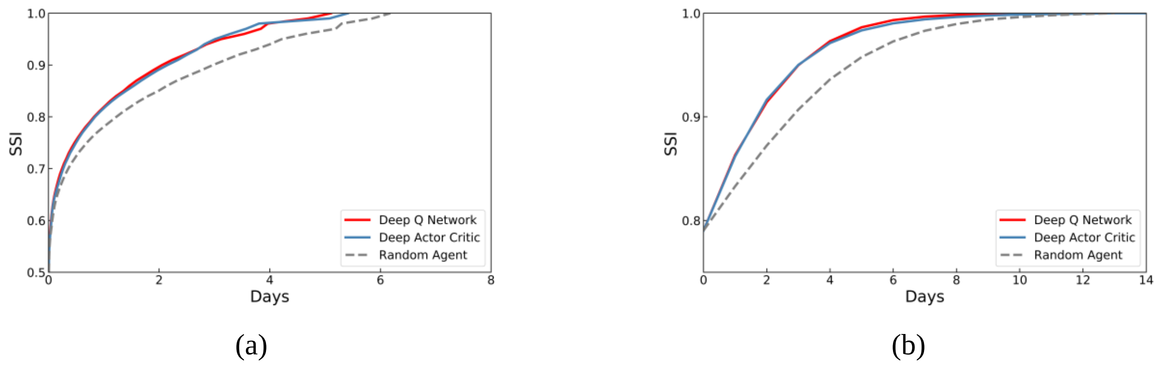

Fig. 7 – Results from all possible cases: (a) average days required to achieve a given SSI level; (b) average SSI achieved at a given day

The proposed method using the Deep Q Network can be applied to small to moderate-sized networks and when the number of repair teams is relatively small, since the Q network needs to evaluate the value of all possible actions at a given state. For a WN with L pipelines and K repair teams, the output dimension of the Q network is $\binom{L}{K}$, thus the computation could become intractable when L and K become large. In general, the actor critic model structure would be a more suitable choice when scaling to a large discrete action space.

4.2 Sensitivity analysis of the training scheme

Fig. 8 presents a sensitivity analysis, where the horizontal axis is the number of random damage cases sampled for training, and the vertical axis is the proportion of DQN outperforming the random agent in the 16383 test cases. The blue solid line represents the performance of the DQN agent trained using different numbers of training cases with 50 episodes per case. As shown in Fig. 8, the proportion of the DQN winning generally increases with the number of training cases at the beginning and reaches the highest point (85.4%) corresponding to 1200 training cases, then starts decreasing. Additionally, fixing the number of training cases at 1200, if the number of episodes is increased to 100 (the red starred point), the winning proportion of the DQN becomes 75%.

The following interpretation of the results presented in the previous paragraph is offered. In the DQN model, the key component is the deep neural network, which approximates the Q value of a given state-action pair. As a general principle for statistical models, there is a need to strike a balance between bias and variance. Consider the situation where the agent trained with 1200 cases and 50 episodes per case reach a stable point (e.g., a local optima). If the agent is trained using additional cases, the stochastic exploration mechanism of the agent (mentioned in Section 2.2) now adds noise to the near-optimal policy of the agent, thus increasing the variance in the estimation. On the other hand, if the number of episodes is increased to 100 (fixing number of training cases to 1200), the Deep Q network overfits on each case, which adds bias to the estimation. Using 1200 cases and 50 episodes per case achieves a reasonable trade-off between bias and variance.

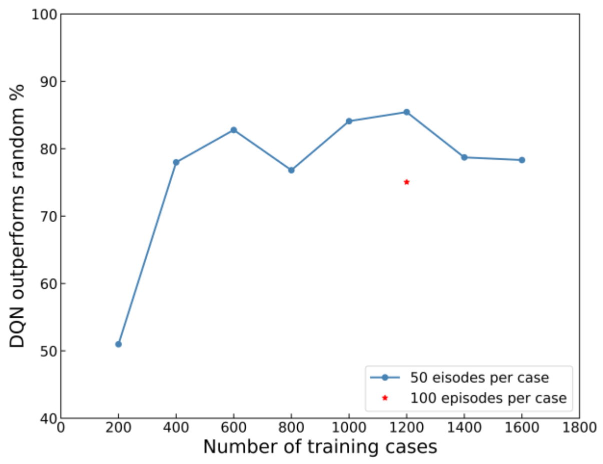

Fig. 8 – Sensitivity analysis on the number of training cases and episodes

5. Conclusion

This study explores the application of the deep reinforcement learning methodology to address the challenge of post-earthquake repair scheduling for water distribution systems. The Markov Decision Process is adapted to define the restoration process. A system model is constructed using the Water Network Tool for Resilience (WNTR) package and used to perform real-time hydraulic analysis. The network model is then embedded in a customized reinforcement learning environment written using the OpenAI gym platform.

Two types of deep reinforcement learning algorithms, namely Deep Q Network (DQN) and Deep Actor Critic (DAC) are adopted to learn the near optimal repair policy by interacting with the water network environment. A general framework for applying the deep reinforcement learning methodology is proposed which can be extended to large water networks while incorporating scenario-based earthquake simulation.

The proposed methodology is applied to a hypothetical network consisting of 14 pipelines. During the model training phase, 1200 and 1000 damage cases are randomly sampled from the entire space (16383 damage cases) for the DQN and DAC agents respectively, and 50 episodes are considered in each case. The trained model is then evaluated on all possible damage cases for the water network and compared with an agent that has no knowledge and employs a random assignment policy of repair actions. Evaluation results demonstrate that the deep reinforcement learning agents successfully learn better repair policies than the random assignment policy, both in terms of application across a wide range of damage scenarios and the average performance. A sensitivity analysis is conducted to explore the influence of the number of cases and episodes per case used for training on the performance of the DQN-based DRL agent. It was determined that using 1200 training cases and 50 episodes per case results in the best performance.

As one of the initial applications of the deep reinforcement learning methodology in the decision making of water network systems, this study shows its capability to support decision making and enhance infrastructure resilience. Nevertheless, there remain several limitations that need to be addressed in future studies. The proposed framework is evaluated on a simple hypothetical water network for demonstration purposes and only pipeline damage and a single repair team are considered. Evaluations on larger networks should be performed with realistic consideration of the possible damage states, number of repair crews and repair time. In addition, theoretical modifications of the proposed Actor Critic model need to be carefully studied to further enhance its scalability for restoration decision making on large water networks and other infrastructure systems.

6. References

[1] United States Department of Homeland Security (DHS) (2013): National Infrastructure Preparedness Plan (NIPP) 2013.

[2] Tabucchi T, Davidson R, Brink S (2010): Simulation of post-earthquake water supply system restoration. *Civil Engineering and Environmental Systems*, **27** (4), 263–79.

[3] Tabucchi T, Davidson R, Brink S (2008): Restoring the Los Angeles water supply system following an earthquake. *14th World Conference on Earthquake Engineering*, Beijing, China.

[4] Lee JY, Tomar A, Burton H (2018): A Framework for Water Distribution Systems Exposed to Seismic Events and Evolving Conditions. *11th US National Conference on Earthquake Engineering*, Los Angeles, California, USA.

[5] Sarkale Y, Nozhati S, Chong EKP, Ellingwood BR, Mahmoud H (2018): Solving Markov decision processes for network-level post-hazard recovery via simulation optimization and rollout. *IEEE International Conference on Automation Science and Engineering*, Munich, Germany.

[6] Xu M, Ouyang M, Mao Z, Xu X (2019): Improving repair sequence scheduling methods for postdisaster critical infrastructure systems. *Computer-Aided Civil and Infrastructure Engineering*, **34** (6), 506–22.

[7] Gomez C, Baker JW (2019): An optimization-based decision support framework for coupled pre- and post-earthquake infrastructure risk management. *Structural Safety*, **77**, 1–9.

[8] Nozhati S, Sarkale Y, Ellingwood B, K.P. Chong E, Mahmoud H (2019): Near-optimal planning using approximate dynamic programming to enhance post-hazard community resilience management. *Reliability Engineering and System Safety*, **181**, 116–26.

[9] Nozhati S, Sarkale Y, Chong EKP, Ellingwood BR (2020): Optimal stochastic dynamic scheduling for managing community recovery from natural hazards. *Reliability Engineering and System Safety*, **193**.

[10] Lee JH, Labadie JW (2007): Stochastic optimization of multireservoir systems via reinforcement learning. *Water Resources Research*, **43** (11).

[11] Sutton RS, Barto AG (2018): *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

[12] Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017): Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, **34** (6), 26–38.

[13] Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, et al. (2016): Mastering the game of Go with deep neural networks and tree search. *Nature*, **529**, 484–489.

[14] Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets AS, Yeo M, et al. (2017): StarCraft II: A New Challenge for Reinforcement Learning. *arXiv preprint*, arXiv: 1708.04782.

[15] Andriotis CP, Papakonstantinou KG (2019): Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering and System Safety*, **191**.

[16] Memarzadeh M, Pozzi M (2019): Model-free reinforcement learning with model-based safe exploration: Optimizing adaptive recovery process of infrastructure systems. *Structural Safety*, **80**, 46–55.

[17] Bellman RE (2003): *Dynamic Programming*. Dover Publications.

[18] Watkins C (1989): *Learning From Delayed Rewards*. PhD Thesis, King's College, Cambridge, UK.

[19] Sutton RS, McAllester D, Singh S, Mansour Y (1999): Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems 12*, 1057-1063, MIT Press.

[20] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. (2015): Human-level control through deep reinforcement learning. *Nature*, **518**, 529–533.

[21] Grondman I, Busoniu L, Lopes GAD, Babuška R (2012): A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, **42** (6), 1291–307.

[22] Cimellaro GP, Reinhorn AM, Bruneau M (2010): Framework for analytical quantification of disaster resilience. *Engineering Structures*, **32** (11), 3639–49.

[23] Klise KA, Murray R, Haxton T (2018): An Overview of the Water Network Tool for Resilience (WNTR). *1st International WDSA/CCWI Joint Conference*, Kingston, Ontario, Canada.

[24] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. (2016): OpenAI Gym. *arXiv preprint*, arXiv: 1606.01540.

[25] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. (2016): TensorFlow: A System for Large-Scale Machine Learning. *12th USENIX Symposium on Operating Systems Design and Implementation*, Savannah, Georgia, USA.

[26] Larock B, Jeppson R, Watters G (2010): *Hydraulics of Pipeline Systems*. CRC Press.

[27] Jayaram N, Srinivasan K (2008): Performance-based optimal design and rehabilitation of water distribution networks using life cycle costing. *Water Resources Research*, **43** (11).