Nuclear Engineering and Technology 55 (2023) 3277–3290

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

journal homepage: www.elsevier.com/locate/net

Original Article

Deep reinforcement learning for a multi-objective operation in a nuclear power plant☆

Junyong Bae, Jae Min Kim, Seung Jun Lee*

Department of Nuclear Engineering, Ulsan National Institute of Science and Technology, 50 UNIST-gil, Ulsan, 44919, Republic of Korea

ARTICLE INFO

Article history:
Received 19 February 2023
Received in revised form
25 May 2023
Accepted 3 June 2023
Available online 5 June 2023

Keywords:
Nuclear power plant
Automation
Deep reinforcement learning
Soft actor-critic
Hindsight experience replay

ABSTRACT

Nuclear power plant (NPP) operations with multiple objectives and devices are still performed manually by operators despite the potential for human error. These operations could be automated to reduce the burden on operators; however, classical approaches may not be suitable for these multi-objective tasks. An alternative approach is deep reinforcement learning (DRL), which has been successful in automating various complex tasks and has been applied in automation of certain operations in NPPs. But despite the recent progress, previous studies using DRL for NPP operations have limitations to handle complex multi-objective operations with multiple devices efficiently. This study proposes a novel DRL-based approach that addresses these limitations by employing a continuous action space and straightforward binary rewards supported by the adoption of a soft actor-critic and hindsight experience replay. The feasibility of the proposed approach was evaluated for controlling the pressure and volume of the reactor coolant while heating the coolant during NPP startup. The results show that the proposed approach can train the agent with a proper strategy for effectively achieving multiple objectives through the control of multiple devices. Moreover, hands-on testing results demonstrate that the trained agent is capable of handling untrained objectives, such as cooldown, with substantial success.

1. Introduction

Nuclear power plants (NPPs) have long utilized automated systems to improve efficiency and safety. The reactor protection system, for example, is automated through rule-based logic, which generates a reactor trip and safety system actuation signals when the monitored parameters reach predefined thresholds. The steam bypass control system automatically removes excess energy from the reactor coolant system using logics and proportional-integral (PI) controllers for turbine bypass valves [1,2]. However, the operations requiring continuous multi-objective decision-making are still manually performed by human operators, despite the potential for human error.

One example is controlling the pressure, volume, and temperature of the reactor coolant during NPP startup. While the reactor coolant in a generic pressurized water reactor is maintained at extremely high pressure and temperature during normal full-power operation, it is maintained at atmospheric pressure and room temperature during overhaul periods, requiring operators in the main control room to pressurize and heat the coolant during the startup. The operating procedures for the startup instruct the operators to heat the coolant by activating and energizing reactor coolant pumps (RCPs) and pressurizer (PZR) heaters, respectively, in line with the decay heat of the reactor core. Based on this incoming heat, the operators should adjust the coolant temperature to the target temperature by regulating the amount of heat removal via the heat exchanger (Hx) of the residual heat removal (RHR) system; specifically, the operators can regulate this by adjusting the position of the RHR Hx flow control valve. Concurrently, the operators should manage the pressure increase and volume expansion due to heat-up by adjusting the positions of the valves for the charging and letdown flows and PZR spray flow. As the control of such devices can affect more than one objective [e.g., closing the letdown flow control valve can decrease the pressure, increase the PZR level (representing the volume of the coolant), and increase the heat removal], the operators should continuously determine a proper status of devices for achieving multiple objectives. In addition, this operation should be conducted under

☆ It is confirmed that this item has not been published nor is currently being submitted elsewhere.

* Corresponding author.
E-mail addresses: junyong8090@unist.ac.kr (J. Bae), jaemink@unist.ac.kr (J.M. Kim), sjlee420@unist.ac.kr (S.J. Lee).

https://doi.org/10.1016/j.net.2023.06.009

dynamically changing situations and generally takes more than 8 h in real NPPs [1]. These characteristics make this operation intricate and mentally taxing for human operators. According to an analysis conducted by Jo et al., over a recent decade, human error caused 20 unplanned reactor trips in the NPPs of the Republic of Korea (ROK), of which 20% (4 cases) were induced by mistakes of the main control room operators during startup and shutdown [3].

Automation of such an operation, meaning one that is multi-objective with continuous decision-making involving multiple devices under dynamically changing situations, can reduce the burden on the operators and ultimately prevent human error. However, classical approaches including rule-based logic and PI controllers may be inappropriate to automate these kinds of operations, since a PI controller is generally for a single-input single-output system and rule-based logic cannot cope with dynamic situations without prepared rules.

An alternative way is deep reinforcement learning (DRL). Reinforcement learning (RL) is a process where an agent (i.e., controller) explores the given environment without any explanation and exploits its experiences to optimize actions (i.e., control). This process requires functions that predict future rewards or optimal actions, etc., and DRL utilizes deep learning models as approximators of these functions. With the advances of deep learning, DRL has successfully automated a variety of complex and human-level tasks, such as the game of Go [4], *StarCraft II* [5], matrix multiplication [6], optimal path finding for multi-arm manipulators [7], and the navigation and control of unmanned aerial vehicles [8].

In the nuclear engineering field, deep learning technology has been widely applied for various purposes such as pipe thinning detection [9], a surrogate model for thermal-hydraulic system code [10–12], diagnosis of sensor faults, abnormal events, and accidents [13–18], and inference models for operator support systems [19–21]. Likewise, DRL has been implemented for the optimization of the fuel loading pattern in a boiling water reactor, where a brute-force search can lead to expensive simulation costs [22]. In the ROK, another research area being actively conducted is the automation of NPP operations. For the aforementioned reasons, such research has especially focused on automating the manual tasks of the startup and shutdown operations and has utilized compact nuclear simulators (CNSs), simplified and modifiable NPP simulators, to facilitate the implementation of the DRL algorithms. Park et al. presented a DRL-based method to automatically maintain the coolant pressure and volume within allowable ranges while the reactor is being heated up [23]. They employed an asynchronous advantage actor-critic (A3C) algorithm, one of the latest DRL algorithms, and designed the agent to choose the optimal action among multiple action candidates. Similarly, Lee et al. utilized the A3C algorithm to train a reactor power control agent that selects the optimal strategy, i.e., increase power, decrease power, or stay [24]. Harmonized with rule-based modules, the power control agent was shown to successfully automate the power-increase operation during NPP startup. However, the approaches of both these studies have limitations when applied to complex multi-objective operations with multiple devices. First, both studies employed discrete action candidates, which can lead to an excessive number of candidates when multiple devices are postulated (e.g., 5 valves with open, close, and stay actions generate $3^5 = 243$ action candidates). Second, the DRL agents in both studies were trained to follow a predefined optimal operating path, while the full potential of DRL is to investigate new strategies for achieving an operating goal.

As an alternative way to apply DRL, Lee et al. tuned the PI controllers of each operating objective (i.e., pressure and level of the coolant) using DRL and compared their performances with a DRL that selects discrete actions and a PI controller tuned in a classical way, i.e., following the Ziegler–Nichols rule [1]. However, this study did not consider the possible conflict between controllers with different objectives. Kim et al. suggested a strategy to resolve these conflicts by evaluating combinations of possible actions through a plant parameter prediction model [20,25]. Nevertheless, these previous studies used a distance-based reward function, which necessitates calibration of the reward scales for multiple objectives through burdensome trial-and-error.

In this study, we suggest a DRL-based approach for multi-objective operations with multiple devices in an NPP. To avoid action candidate explosion, we designed a DRL-trained agent to directly output the continuous status of devices, such as the percentile position and power of the valves and the PZR heater, respectively. We also employed a soft-actor critic (SAC) algorithm, which is a DRL algorithm suitable for continuous action space [26]. To simplify the reward design for multi-objectives, the proposed method incorporates hindsight experience replay (HER) [7,8,27]. The reward function in this approach can take a binary form, providing feedback only when the objective is achieved. This binary reward is thus straightforward and does not require calibration for each objective. Moreover, this form of reward allows the DRL agent to explore and establish its policy for multi-objectives, rather than following a predefined optimal operating path. The suggested DRL method was utilized in a case study to train an agent that controls the reactor coolant status during NPP startup via activation and deactivation of the RCPs and steam generators (SGs), respectively. During the training session, the agent was tasked with heating the reactor coolant while adjusting the pressure and volume to various statuses by controlling four valve positions and PZR heater power. As a result, the suggested DRL method successfully trained the agent with an operating strategy that can achieve the multiple objectives efficiently. Additionally, a hands-on test shows that the trained agent can perform not only the trained operation (i.e., heat-up) but also an untrained operation (i.e., cooldown). We believe that this research will contribute to the further application of DRL in NPPs, such as the automation of other high-level operations and the investigation into novel operating strategies.

The remaining parts of this paper are as follows. Section 2 briefly introduces the concepts of DRL, actor-critics, the SAC algorithm, and HER. Section 3 details the suggested DRL method with SAC and HER as well the designs for NPP applications. The target operation for the case study and the results are given in Section 4. Section 5 presents a discussion about the contributions of this work and further studies, and Section 6 concludes this paper.

2. Soft actor-critic (SAC) and hindsight experience replay (HER)

2.1. Reinforcement learning and actor-critic method

RL is a machine learning paradigm that trains an agent to output an action that can maximize the notion of cumulative reward by exploiting its experience in a training environment. In an RL framework, the problem needs to be modeled as a Markov decision process (MDP) consisting of the tuple $\{S, A, P, r, \gamma\}$, where S is the state space, A is the action space, P is a transition probability, r is a reward function, and $\gamma \in [0, 1]$ is a discount factor. It assumes that the state transition depends only on the current state and the chosen action, not on previous states and actions. In other words, for the current time step t, the current action $a_t \in A$ will evolve the current state $s_t \in S$ into the subsequent state $s_{t+1} \in S$ according to the transition probability $P(s_{t+1}|s_t, a_t)$. This transition can be scored by the reward function $r_{a_t}(s_t, s_{t+1})$, and a sequence of transitions (i.e., process) can also be evaluated by cumulative reward $G_t = \sum_{i=0}^{\infty} \gamma^{t+i} r_{a_{t+i}}(s_{t+i}, s_{t+i+1})$, which is the discounted summation of

future rewards. The objective of the MDP is to shape a policy that guides the selection of actions to maximize the cumulative reward. This policy can be defined mathematically as a distribution of the probability to choose an action for a given state $\pi(a|s)$.

Since it is not practical to compute the exact cumulative reward in most cases, expectations of G_t can be used to guide the creation of the policy. For instance, an action-value function $q_\pi(s_t, a_t)$ is the expected cumulative reward when starting from s_t, taking a_t, and following the policy π, as shown in Eq. (1) [28]:

$$q_\pi(s_t, a_t) = \mathbb{E}_\pi[G_t | s = s_t, a = a_t] \tag{1}$$

If the policy π promises the optimal action-value function $q_*(s_t, a_t) = \max_\pi q_\pi(s_t, a_t)$, it will be an optimal policy, which is a solution for the MDP problem. Conversely, if the optimal action-value function q_* is known, the optimal policy can be derived. Equation (2) is an example of the optimal policy π_* for discrete action space [28]:

$$\pi_*(a|s) = \begin{cases} 1 & if \ \ a = \underset{a \in \mathrm{A}}{argmax}\, q_*(s, a) \\ 0 & otherwise \end{cases} \tag{2}$$

RL approximates the optimal action-value function through a recursive equation and experiences. For an action-value function approximator $Q(s_t, a_t)$, the relationship between current and successive time steps can be expressed as in Eq. (3) [28]:

$$\begin{aligned} Q(s_t,a_t) &= \mathbb{E}_{\pi_*}[G_t | s=s_t, a=a_t] = r_{a_t}(s_t,s_{t+1}) + \mathbb{E}_{\pi_*}\left[\gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots\right] \\ &= r_{a_t}(s_t,s_{t+1}) + \mathbb{E}_{\pi_*}\left[\gamma\left(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots\right)\right] \\ &= r_{a_t}(s_t,s_{t+1}) + \mathbb{E}_{\pi_*}[\gamma G_{t+1}] \approx r_{a_t}(s_t,s_{t+1}) + \gamma \mathbb{E}_{\pi_*}[Q(s_{t+1},a_{t+1})] \end{aligned} \tag{3}$$

Since the next step action a_{t+1} can also be approximated as the action with the maximum action value according to the approximator $Q(s_t, a_t)$, Eq. (3) can be rewritten as shown in Eq. (4):

$$\begin{aligned} Q(s_t, a_t) &\approx r_{a_t}(s_t, s_{t+1}) + Q\left(s_{t+1}, \underset{a_{t+1} \in \mathrm{A}}{argmax}\, Q(s_{t+1}, a_{t+1})\right) \\ &\approx r_{a_t}(s_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \end{aligned} \tag{4}$$

which suggests that the approximator $Q(s_t, a_t)$ can be updated using the experience of an RL agent specifically through the use of historical records of interactions (s_t, a_t, s_{t+1}, r_t) between the RL agent and the training environment. It has been demonstrated that through sufficient updates with enough experience, $Q(s, a)$ can converge to the optimal action-value function $q_*(s, a)$ [28]. This update can be achieved by minimizing the objective function given by Eq. (5) when the approximator is parameterized by θ:

$$J_Q(\theta) = \left[Q_\theta(s_t, a_t) - \left(r_t + \gamma \max_{a_{t+1}} Q_\theta(s_{t+1}, a_{t+1})\right)\right]^2 \tag{5}$$

In this context, there are two possible approaches: value-based and policy-based [7,28]. In value-based RL, the value function approximator estimates the expected return of each action in the action space, and the policy selects the action that maximizes the expected return, as shown in Eq. (2). Deep Q-network (DQN), which utilizes a deep-learning model as a function approximator, is an example of a value-based RL [29]. In contrast, in policy-based RL, the policy approximator directly outputs an optimal action without consulting a value function. Unlike value-based RL, which requires a set of possible actions to choose from, policy-based RL does not need to discretize the action space to generate candidates. This makes policy-based RL particularly effective for continuous and high-dimensional action spaces.

One policy-based RL approach is the actor-critic method, which combines a policy approximator (the actor) with a value function approximator (the critic). The critic is used to guide the training of the actor toward maximizing the expected returns, rather than just providing feedback on the actor's performance. There are various actor-critic methods including a deep deterministic policy gradient (DDPG), asynchronous advantage actor-critic (A3C), and proximal policy optimization (PPO), which differ in terms of the determinism of the policy, how the value function is used to update the policy, and how the training is stabilized. Currently, deep learning models are generally being used to approximate value and policy functions in RL; this approach, called DRL, has increased the performance and applicability of RL based on the benefits advanced deep learning techniques.

In DRL, accumulating experiences is critical as the value and policy functions are updated based on the experience gained. There are two approaches to how the current policy is utilized: on-policy and off-policy. On-policy methods update the value and policy functions based on the actions taken by the current policy, while off-policy methods can update them based on the actions of a different policy. Off-policy methods are more experience-efficient since they can learn from past experience; however, they can have stability issues and are more sensitive to hyperparameter choices.

2.2. Soft actor-critic (SAC)

SAC, as suggested by Haarnoja et al., in 2018 [26,30], is an actor-critic and off-policy method that optimizes a stochastic policy with a soft value function. One of the key features of SAC is entropy regularization, which means that it not only considers the expected cumulative reward but also the entropy of a stochastic policy, $\pi(a|s)$.

Off-policy methods compile experience while balancing exploration and exploitation. For example, DQN uses an ε-greedy method that adjusts the proportion of the random action (exploration) to the 'greedy' optimal action (exploitation) based on decaying ε values. However, this method does not take into account the current state when adjusting the trade-off between exploration and exploitation.

SAC addresses this issue by entropy regularization, which means augmenting the entropy of the policy for a given state as an additional term in the value functions. With entropy $\mathbb{E}[-\log \pi(\bullet|s)]$, the entropy-augmented Q-function, named soft Q-function, can be defined as shown in Eq. (6), where α is the temperature weight of the entropy term, and the action of next step is sampled based on $\pi(\bullet|s)$:

$$Q(s_t,a_t) := r_{a_t}(s_t,s_{t+1}) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q(s_{t+1},a_{t+1}) - \alpha \log \pi(a_{t+1}|s_{t+1})] \tag{6}$$

The objective function for the critic network in SAC is given by Eq. (7). The target critic network, parametrized by $\widehat{\theta}$, is used to help stabilize the training of the main critic network. This network can be either the one that is updated less frequently than the main critic network or the one that is selected from among the two separate critic networks based on its promise of a minimum action value. Equation (7) reads:

$$J_Q(\theta) = [Q_\theta(s_t, a_t) - (r_t + \gamma Q_{\widehat{\theta}}(s_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1}|s_{t+1})]^2 \tag{7}$$

In SAC, the goal of the policy update is to fit the stochastic policy

to the distribution of the soft Q-function, as measured by the Kullback–Leibler (KL) divergence $D_{KL}(P \parallel Q)$. The objective function for the actor network is shown in Eq. (8), where $Z(s_t)$ is a function normalizing $\exp(\frac{1}{\alpha}Q_\theta(s_t,\bullet))$:

$$J_\pi = D_{KL}\left(\pi(\bullet|s)\middle\|\frac{\exp(\frac{1}{\alpha}Q_\theta(s_t,\bullet))}{Z(s_t)}\right) \tag{8}$$

Since the KL divergence can also be expressed in the expectation form $D_{KL}(P \parallel Q) = \mathbb{E}_{x\sim P}[\log P(x) - \log Q(x)]$, Eq. (8) can be rewritten with ϕ, which parameterizes the actor network, as shown in Eq. (9):

$$J_\pi(\phi) = \mathbb{E}_{\hat{a}_t\sim\pi_\phi}[\alpha \log \pi_\phi(\hat{a}_t|s_t) - Q_\theta(s_t,\hat{a}_t)] \tag{9}$$

Equation (8) implies that the objective of the policy update is to not only maximize the expected sum of future rewards but also maximize the entropy of the policy, which encourages exploration. This allows SAC to adapt its exploration–exploitation strategy based on the state of the environment. When the optimal action for a given state is unknown, the actor network will increase the randomness of its action selection to explore the optimal action. Once the optimal action is known, the actor network will exploit it. As a result, SAC can perform more continuous and smooth exploration and exploitation based on the state of a given task.

2.3. Hindsight experience replay (HER)

HER, introduced in the paper "Hindsight Experience Replay" by Andrychowicz et al., is a technique that can improve RL training efficiency by avoiding complicated reward engineering [7,8,27]. Many real-world tasks only offer sparse and binary rewards, meaning that the agent receives a response about the trials only under certain specific conditions. For example, in a robotic manipulation task, the agent might only receive a reward when it successfully moves an object to a certain location. These sparse and binary rewards can make it difficult for the agent to learn from its experiences since most of the experiences will not result in a meaningful reward. HER addresses this problem by allowing the RL agents to learn from hindsight experience, where the desired goals are shifted to arbitrary ones.

In HER, the concept of the goal $g \in G$, where G is the goal space, is introduced. The reward function and states are redefined as $r_t := r_{a_t}(s_t, s_{t+1}, g)$ and $s_t := s_t|g$, respectively. Like other off-policy RL methods, HER stores every transition $(s_t|g, a_t, s_{t+1}|g, r_{a_t}(s_t, s_{t+1}, g))$ in a replay buffer after experiencing an episode $\{s_0, s_1, \ldots, s_T\}$. The stored transitions are then recomputed with a shifted goal $\hat{g}$, and new transitions $(s_t|\hat{g}, a_t, s_{t+1}|\hat{g}, r_{a_t}(s_t, s_{t+1}, \hat{g}))$ are added to the replay buffer. If we set the shifted goal $\hat{g}$ as the one achieved during the episode $\{s_0, s_1, \ldots, s_T\}$, then the modified replay buffer will contain some successful transitions that might help train the RL agent. With HER, we can use intuitive and straightforward rewards in real-world tasks without having to conduct complicated reward engineering.

3. Deep reinforcement learning for multi-objective operations in NPPs

Many operations in NPPs, such as controlling pressure, volume, and temperature during startup and shutdown, can be classified as multi-objective and multi-device, making automation of such operations challenging. In this study, we propose a DRL-based approach to address this challenge. Unlike previous approaches that utilized multiple agents for each objective [1,23–25,31], our approach trains a single agent to determine the appropriate device status to achieve multiple objectives simultaneously. Fig. 1 provides a schematic illustration of the proposed approach.

The proposed approach composes the state s using four types of information: the current values, the changes over the past N time steps, the deviation from the target values, and the target values of the objectives. If three objectives are postulated, the state s will be represented as a vector with twelve parameters (i.e., four types of information and three objective parameters). Configuration of the state s can be varied according to the target operation. The action a consists of the continuous status of multiple devices, such as the positions of the valves and the power of heaters.

The baseline of the proposed approach is a SAC [26,30]. This algorithm is suitable for continuous action space, such as valve positions and heater power, and has high exploration–exploitation efficiency under a complex state space, such as an NPP coolant system. As an actor-critic method, our approach embeds two types of deep-learning models: actor and critic networks. The actor network (i.e., a DRL agent) consists of an input layer for the state s_t, hidden layers, and an output layer for the policy $\pi(\bullet|s)$. It is assumed that the policy $\pi(\bullet|s)$ follows a Gaussian distribution, and therefore, the output layer of the actor network outputs the mean μ and standard deviation σ for each device status. The critic network was designed to include two input channels, one for the state s_t and the other for action a_t, with each channel having an input layer and a hidden layer. These two channels are concatenated and connected to a main body composed of multiple hidden layers and an output layer that produces the action value $Q(s_t, a_t)$.

The gradients for updating the trainable parameters (i.e., ϕ and π) of the actor and critic networks are calculated according to historical transitions and the objective functions [i.e., Eq. (9) for the actor network and Eq. (7) for the critic network]. To reduce the instability of network updates, the proposed approach utilizes two methods. First, an additional critic network, named the target critic network, is constructed. This network is updated after a delay from the main critic network update and calculates the targeted values of the action values in Eqs. (7) and (9) instead of the main network. This helps to reduce the instability caused by the recursive appearance of the critic network in the objective functions. Second, the proposed approach converges the temperature weight α to the optimal value for the actor and critic updates according to the objective function shown in Fig. 1, as guided in Ref. [26].

As an off-policy algorithm, SAC employs a replay buffer to store past experiences in the form of the transitions (s_t, a_t, s_{t+1}, r_t). The proposed approach accumulates these transitions by repeatedly conducting trial episodes with randomly targeted objectives in multiple training environments. Each environment consists of an NPP simulator and an episodic actor network. The episodic actor network has the same architecture as the main actor network, and its trainable parameters (i.e., $\hat{\phi}_k$) are updated into those of the main actor network at the start of each new trial episode.

Since the action space of NPP operations is bounded within finite intervals (e.g., the position of a valve must be between 0% and 100%), the proposed approach transforms the policy $\pi(\bullet|s)$ of the episodic actor network using a squashing function of the hyperbolic tangent function and then samples the action $\hat{a}_t$ from the transformed distributions, as instructed in Ref. [26]. The sampled action $\hat{a}_t$ is provided to the proportional controllers to convert the continuous action $\hat{a}_t$ as a device status into discrete toggle signals (e.g., open or close) for matching the device status into the action $\hat{a}_t$. To prevent excessive devices controls, the action $\hat{a}_t$ is sampled every N time steps, allowing the controllers to gradually adjust the device status over N time steps. Additionally, the proposed

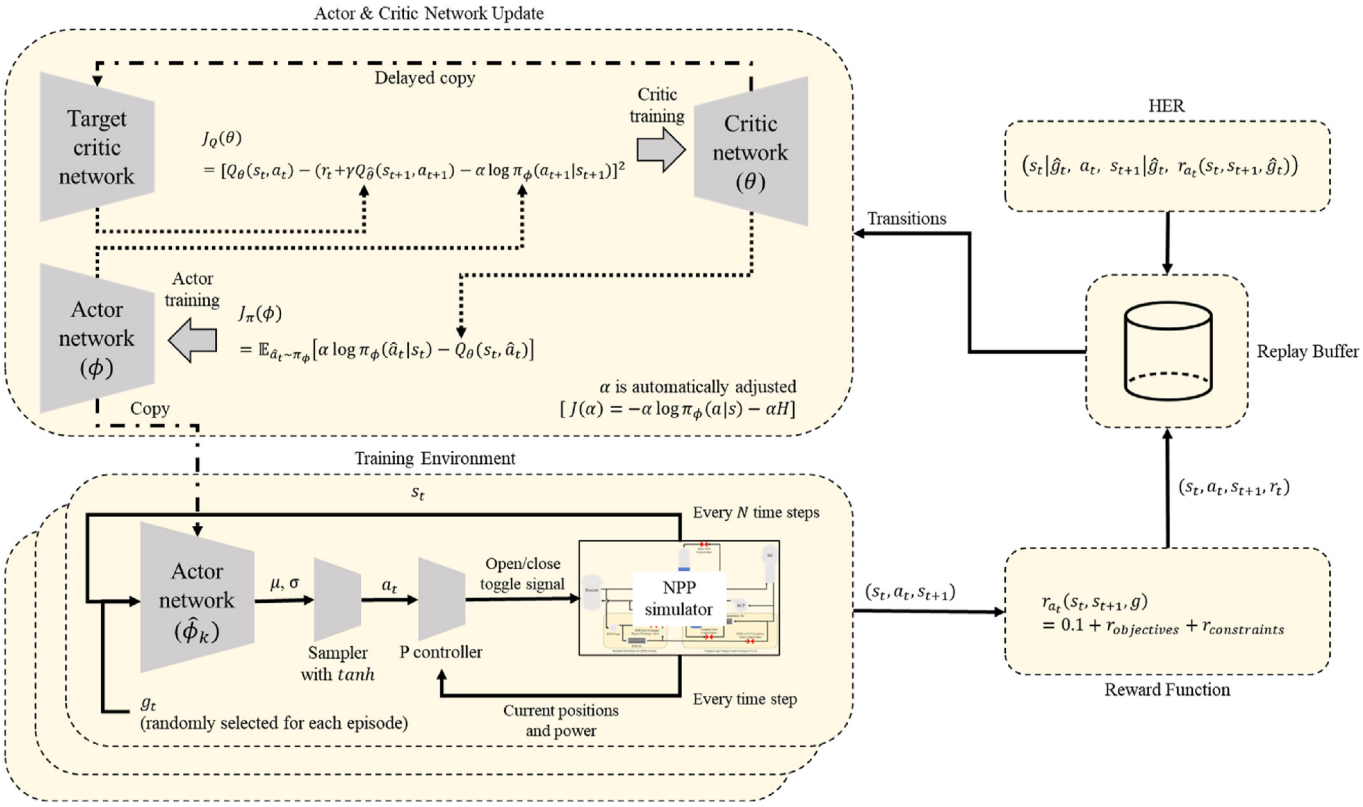

Fig. 1. Framework of DRL for multi-objective operations in NPPs.

approach allows for a device status error corresponding to changes when the toggle signals are activated for a single time step.

The transition from s_t to s_{t+1} induced by a_t is evaluated by the reward function. In contrast to previous studies [1,23–25,31], the proposed approach uses binary reward functions consisting of three terms, namely objective, constraint, and survival. The survival term provides a small positive reward for each time step that the episode continues, while the constraint term gives a negative reward in the event of constraint violations. The objective term is composed of multiple binary reward functions for each objective that provide positive rewards when each objective is achieved. This reward configuration is a straightforward way to describe multi-objective operations. Compared to distance-based rewards such as deviation from target values or the optimal path to achieve objectives, the binary reward does not require complex reward engineering, such as calibrating reward scales for each objective through trial and error as seen in previous studies [1,23–25,31]. Despite this advantage, though, adopting a binary reward can make it challenging to train the agent as it does not provide any feedback, even if the agent is moving in the correct direction. Moreover, if achieving a certain objective takes a long time compared to others, the DRL agent may struggle to establish a policy for that objective even though it satisfies the others.

To overcome this limitation, the proposed approach adapts the HER algorithm, as detailed in Section 2.3. Note that the idea of combining SAC and HER has been previously researched in Refs. [7,8,32]. The HER algorithm adds a new transition $(s_t|\widehat{g}_t, a_t, s_{t+1}|\widehat{g}_t, r_{a_t}(s_t, s_{t+1}, \widehat{g}_t))$ to the buffer by reexamining the transitions during the episode against an arbitrary goal $\widehat{g}_t$. Following the future strategy described in Ref. [27], our implementation of the HER algorithm randomly selects s_t during the episode and sets an arbitrary goal $\widehat{g}_t$ as the future state $s_{t+HER_{future}}$. Intuitively, even when these experiences are mostly failed ones, the agent is enabled to learn from its experience of achieving $\widehat{g}_t = s_{t+HER_{future}}$ from the state s_t.

4. Case study

4.1. Target operation and MDP formulation

A case study was conducted to assess the feasibility of the proposed DRL-based approach for automating multi-objective and multi-device operations in NPPs. We applied the proposed approach to train an agent that can automatically control the reactor coolant status during startup and shutdown operations. Since training a DRL agent requires many trial-and-error iterations, we employed a CNS to accelerate the training process. The CNS simulates a simplified version of a Westinghouse 1000 MWe 3-loop pressurized water reactor, and its thermal-hydraulic system was imitated by SMABRAE (which stands for small-break loss of coolant accidents) code with minimized nodalization [33,34]. We improved the cyclical speed of the simulations and ran them in parallel and asynchronously, ultimately creating multiple DRL training environments as has been described in the previous section here as well as previous studies [23,25].

Startup and shutdown operations are carried out according to procedures. The general operating procedures of the CNS include six sequentially conducted procedures for the startup operation, among which the cold-shutdown to hot-shutdown operation was chosen as the automation target in the current study. This operation involves increasing the temperature of the coolant to 176 °C after generating bubbles in the PZR. Instead of an SG and secondary system, the heat of the coolant is controlled by the RHR system. Table 1 shows the initial and final values of the pressure (P), volume

(V), and temperature (T) of the coolant during this operation. Note that the volume can be represented by the coolant level of the PZR since it locates at the top of the coolant system.

This operation can be divided into two phases: before and after bubble creation. In the first phase, RCPs and PZR heaters are activated and energized. This heats the coolant in the PZR to the saturation temperature, resulting in the formation of a space filled with saturated steam (i.e., a bubble) in the PZR. Concurrently, the operators should stabilize the pressure within the proper boundary denoted in a pressure-temperature (PT) curve. In the second phase, the overall coolant is heated up in a stepwise manner and the pressure and volume are simultaneously adjusted to the final conditions. In this study, we postulated that the DRL agent would be used after bubble creation because the second phase is a multi-objective and long-term operation, whereas the first phase has a single objective (i.e., pressure) and a relatively short-term operation. It is worth noting here that the authors' previous research suggested an autonomous pressure controller with DQN for the first phase [31].

Fig. 2 shows a simplified schematic of the reactor coolant system and auxiliary systems of the CNS. The major devices for controlling the coolant status after bubble creation are as follows: charging flow control valve, letdown flow control valve, RHR Hx bypass/discharge flow control valves, spray flow control valve, and PZR proportional and backup heaters. The positions of the RHR Hx valves are inversely proportional to each other. These devices affect more than one objective; for instance, opening the charging valve can increase both the volume and pressure of the coolant and vice versa for the letdown control valve. The position of the RHR Hx valves can influence not only the temperature but also the volume of the coolant. Consequently, there may be a conflict between the objectives when controlling the devices [25].

The MDP problem for this operation is to achieve three objectives (pressure, volume, and temperature) by controlling five devices (charging flow control valve, letdown flow control valve, RHR Hx discharge flow control valve, spray flow control valve, and PZR proportional heater). In other words, the state contains twelve values from the current values, changes over the past time steps, deviation from the target values, and target values of the pressure, volume, and temperature, and the action contains five values from the positions of the four valves and the power of the heater.

The proposed approach was implemented using NumPy [35] and TensorFlow [36], open-source Python libraries for array computation and neural network implementation, respectively. The actor network (i.e., the agent) was constructed with three hidden layers containing 256 nodes each, and the critic network comprised a hidden layer containing 256 nodes for each channel and four hidden layers with 512, 256, 128, and 64 nodes, respectively, for the main body. During training, the agent was allowed to output the action $\hat{a}_t$ every 60 time steps (i.e., $N = 60$). The proportional controllers were set to adjust the device status to the values updated every 60 s in the simulator, as one time step was calibrated as 1 s in the simulator. A 3.76% error tolerance was allowed for the positions of the charging and letdown flow control valves, as this is the amount of change observed when the open/close toggle buttons are activated for 1 s in the simulator.

The reward function of this study consisted of a survival reward of 0.1, one negative reward for the operating constraints, and three positive rewards for pressure, volume, and temperature, as shown in Eq. (10). As depicted in Eqs. (11–13), the positive rewards for each objective give a value of 10 when each objective is met:

$$r_{a_t}(s_t, s_{t+1}, g) = 0.1 + r_P + r_V + r_T + r_{constraints} \tag{10}$$

$$r_P = \begin{cases} 10 & if \quad |P_{target} - P_{t+1}| < 1.0\ kg/cm^2 \\ 0 & otherwise \end{cases} \tag{11}$$

$$r_V = \begin{cases} 10 & if \quad |V_{target} - V_{t+1}| < 2\ \% \\ 0 & otherwise \end{cases} \tag{12}$$

$$r_T = \begin{cases} 10 & if \quad |T_{target} - T_{t+1}| < 3\ ^\circ C \\ 0 & otherwise \end{cases} \tag{13}$$

Two operating constraints for coolant status control were postulated. The first constraint is that the coolant level V_t should be maintained between 17% and 99% to properly cover the heaters and maintain a space filled with saturated steam, respectively. The second constraint is that the (P_t, T_t) point should remain within the safety region as defined by the PT curve shown in Fig. 3. If the DRL agent violates these constraints, a negative reward of $r_{constraints} = -10$ was given, otherwise, $r_{constraints} = 0$.

Each training episode started with three different initial conditions corresponding to different coolant levels (90%, 50%, and 30%) after bubble creation. The target values P_{target}, V_{target}, and T_{target} were randomly selected from the following intervals: $P_{target} \in [20\ kg/cm^2,\ 35\ kg/cm^2]$, $V_{target} \in [20\ \%,\ 60\ \%]$, and $T_{target} \in [110^\circ C, 180^\circ C]$. Each episode lasted for 21600 s (i.e., 6 h) and was terminated if the DRL agent violated the constraints.

The replay buffer in this implementation was designed to store up to 2,000,000 transitions. Considering the maximum episode length (21600 s) and the time interval (60 s), the replay buffer could store the experiences from at least 5,500 episodes. A portion of the stored transitions, specifically 5% from episodes that lasted longer than 5 time steps, were reexamined by the HER algorithm with $HER_{future} = 3$. At the end of each episode, 128 transitions were randomly selected from the replay buffer and used to update the actor and critic networks.

4.2. Training results

The graph in Fig. 4 shows the moving averages of the episodic rewards throughout the training process, where episodic rewards are the summation of the rewards earned during each episode. The graph has two notable increases in episodic rewards at around the 400th and 4200th episodes. Following these increases, the behavior of the DRL agent was found to have changed significantly.

The first notable increase in episodic rewards occurring around the 400th episode was caused by the pressure reward r_V (blue line) and volume reward r_P (orange line), as depicted in Fig. 4. The operating records of the 200th and 450th episodes, shown in Fig. 5, provide insight into how the DRL agent was improved at around the 400th episode. These records show the trends and target ranges of

Table 1
Initial and final coolant P, V, T during the cold-shutdown to hot-shutdown operation.

	Parameter	Initial condition	Final condition
Pressure (P)	Pressurizer pressure	24.6–28.1 kg/cm^2	30–45 kg/cm^2
Volume (V)	Percentile level of pressurizer	100%	~ 50% (more than 22%)
Temperature (T)	Loop 1, 2, 3 average temperatures	~ 80 °C	171–176 °C

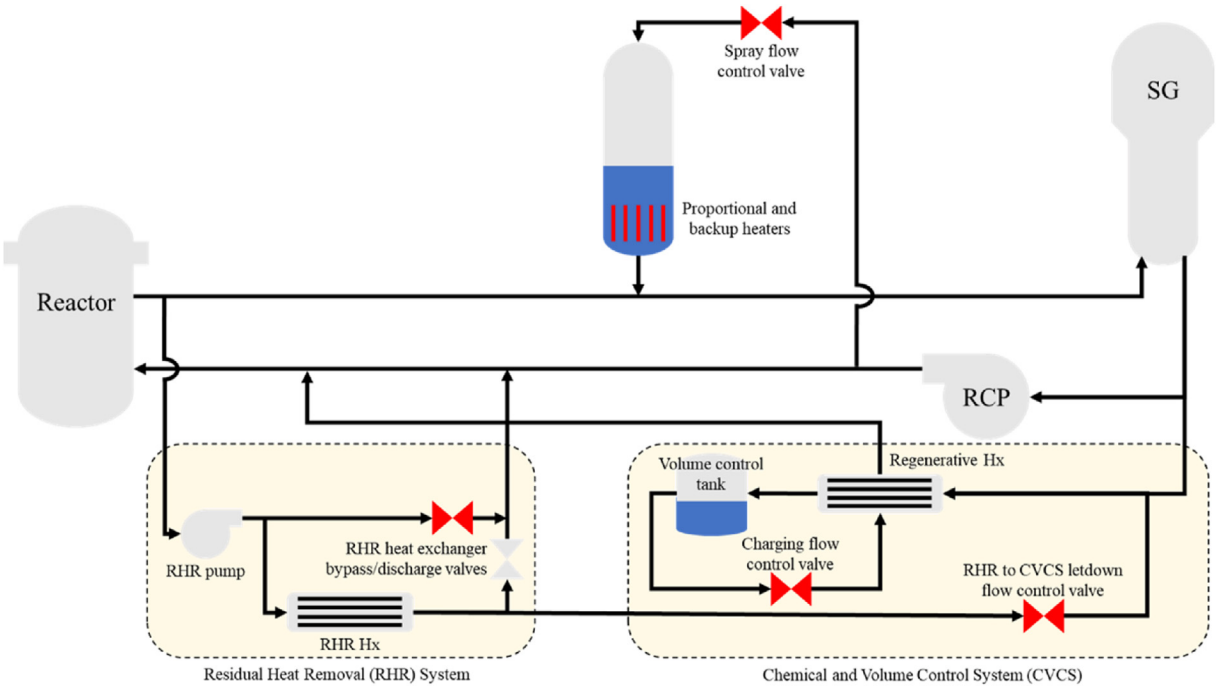

Fig. 2. Simplified schematic of the reactor coolant system and auxiliary systems (i.e., RHR and CVCS). Devices highlighted in red were manipulated by the DRL agent to control the coolant status. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

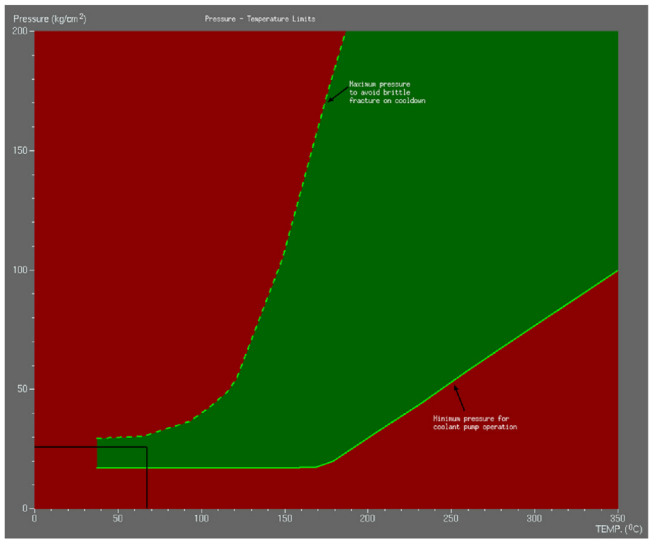

Fig. 3. Pressure–Temperature Limits, also known as the PT curve, of the CNS. (P, T) should be kept within the green area to prevent mechanical stress on the reactor coolant system. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

each objective (blue-shaded areas) as well as the positions of the valves and the power of the heater over time. At the 200th episode, the DRL agent failed to achieve any objectives and was terminated in an early stage as the pressure dropped below the limit of the PT curve. In contrast, at the 450th episode, the agent successfully achieved the pressure and volume objectives and maintained them within the target ranges. This suggests that the DRL agent had learned a policy for controlling the pressure and volume of the reactor coolant after experiencing 400 episodes. However, the agent was unable to achieve the temperature objective.

Fig. 4. Trends of episodic rewards by training episode. The lines are 100 moving average values, and the shaded area shows the $\pm 1\sigma$ range of total reward.

The second increase in episodic rewards occurring around the 4200th episode was attributed to the temperature reward r_T (green line), as depicted in Fig. 4. The operating record of the 4500th episode, shown in Fig. 6, illustrates that the agent was able to learn how to increase the temperature of the coolant after undergoing more than 4200 episodes. As observed in Fig. 4, the temperature episodic rewards were smaller in comparison to the pressure and level rewards. This discrepancy was attributed to the fact that the reward function provided a non-zero value only when an objective was satisfied and satisfying the temperature objective required a longer time compared to the pressure and volume objectives. This longer time seems to be the reason the temperature objective necessitated significantly more episodic experiences compared to the others.

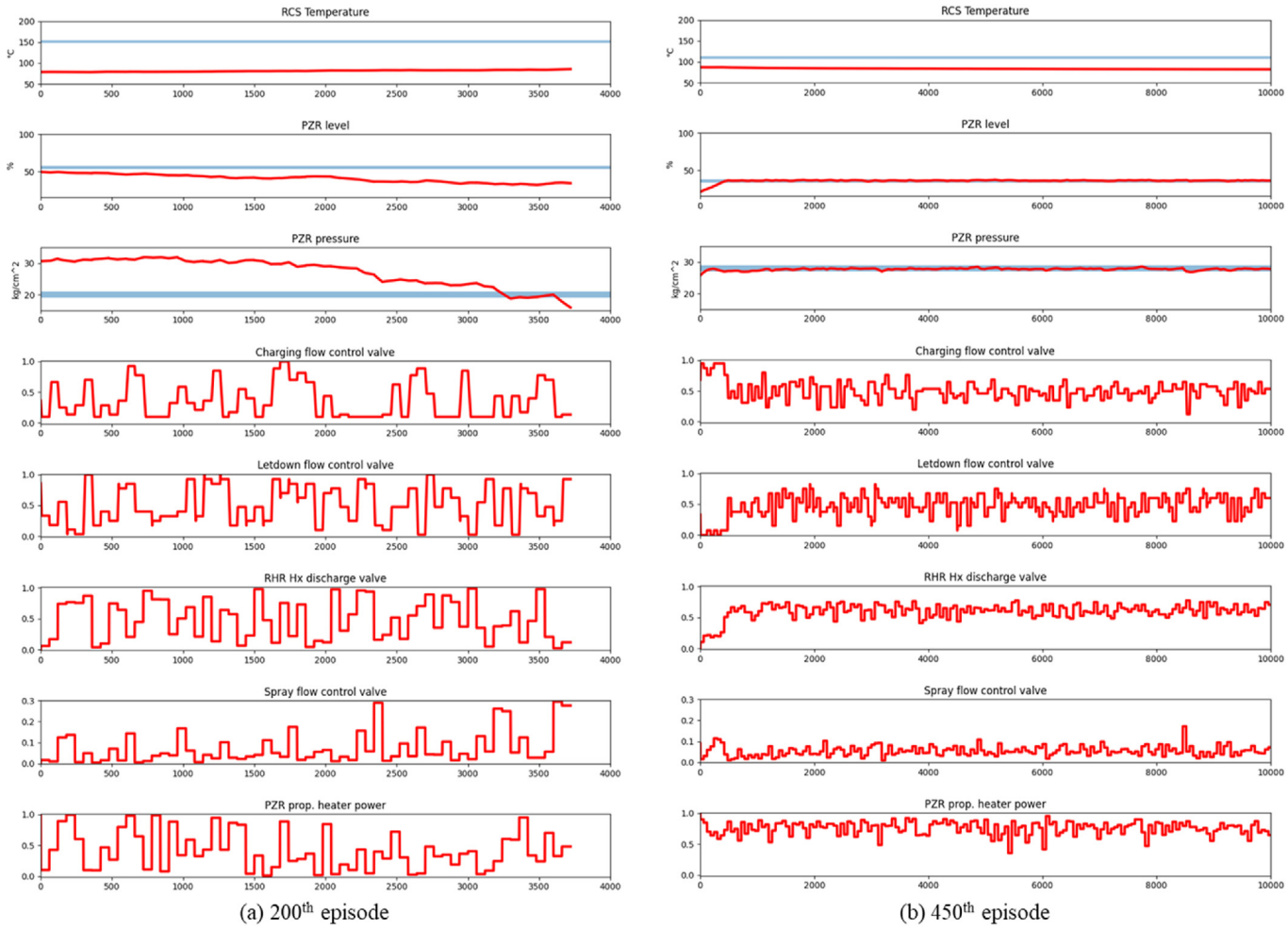

Fig. 5. Operating records of (a) the 200th training episode up to 4,000 s, and of (b) the 450th training episode up to 10,000 s. The 200th training episode was terminated prematurely due to a violation of the PT curve constraint. In contrast, the 450th training episode achieved the pressure and volume objectives but failed to meet the temperature objective.

It seems that the HER algorithm enables the trained DRL agent to achieve the temperature objective. In the previous trials, a SAC without the HER algorithm was successful in training the agent to achieve the pressure and volume objectives, but the previous DRL agents failed to reach the target temperature. Operating records of the previous trials showed that the trained agent rarely experienced a positive temperature reward, as it took a long time to reach the target through the exploration of SAC. In contrast, the framework of this study with the aid of the HER algorithm allowed the DRL agent to experience positive temperature rewards with artificial transitions that reexamined the real transitions against arbitrary temperature targets. By learning from these hindsight experiences, it appears that the DRL agent was able to learn how to achieve the temperature objective.

After experiencing more than 7000 episodes, the DRL agent mostly succeeded in satisfying all objectives. The trained DRL agent effectively addressed the multi-objective control problem by prioritizing the objectives based on the time required for their achievement. Specifically, the agent prioritized the pressure objective first, followed by the volume objective, and finally the temperature objective. This approach is the way it could maximize the episodic rewards.

The DRL agent initially focused on both the pressure and coolant level objectives simultaneously. When the pressure needed to be increased, the agent primarily closed the RHR Hx discharge valve. Conversely, the spray flow control valve was mainly opened when the pressure needed to be reduced. For the volume objective, the charging and letdown flow control valves were opened and closed, respectively, to raise the PZR level and vice versa. For instance, at the 7031th episode (Fig. 7a), the agent reduced the pressure by opening the spray valve and increased the PZR level by opening the charging valve and closing the letdown valve. In an instance of conflicting objectives, pressure was given priority. For instance, reducing the PZR level by maximizing the letdown flow and minimizing the charging flow can also reduce the PZR pressure. To resolve this conflict, at the 7011th episode, the agent first pressurized the coolant by closing the discharge valve to limit heat removal while maintaining the level by opening the charging valve to approximately 75%, as shown in Fig. 7b. Once the pressure objective was met, the agent began reducing the PZR level by closing the charging valve.

As shown in Fig. 8, after achieving the volume objective, the agent balanced the charging and letdown flows and opened the spray valve to maintain both objectives and then proceeded to heat the coolant by closing the discharge valve. Once the temperature reached the target, the agent maintained it by opening the discharge valve and increasing the flow to the Hx by opening the charging and letdown flow control valves slightly more to remove

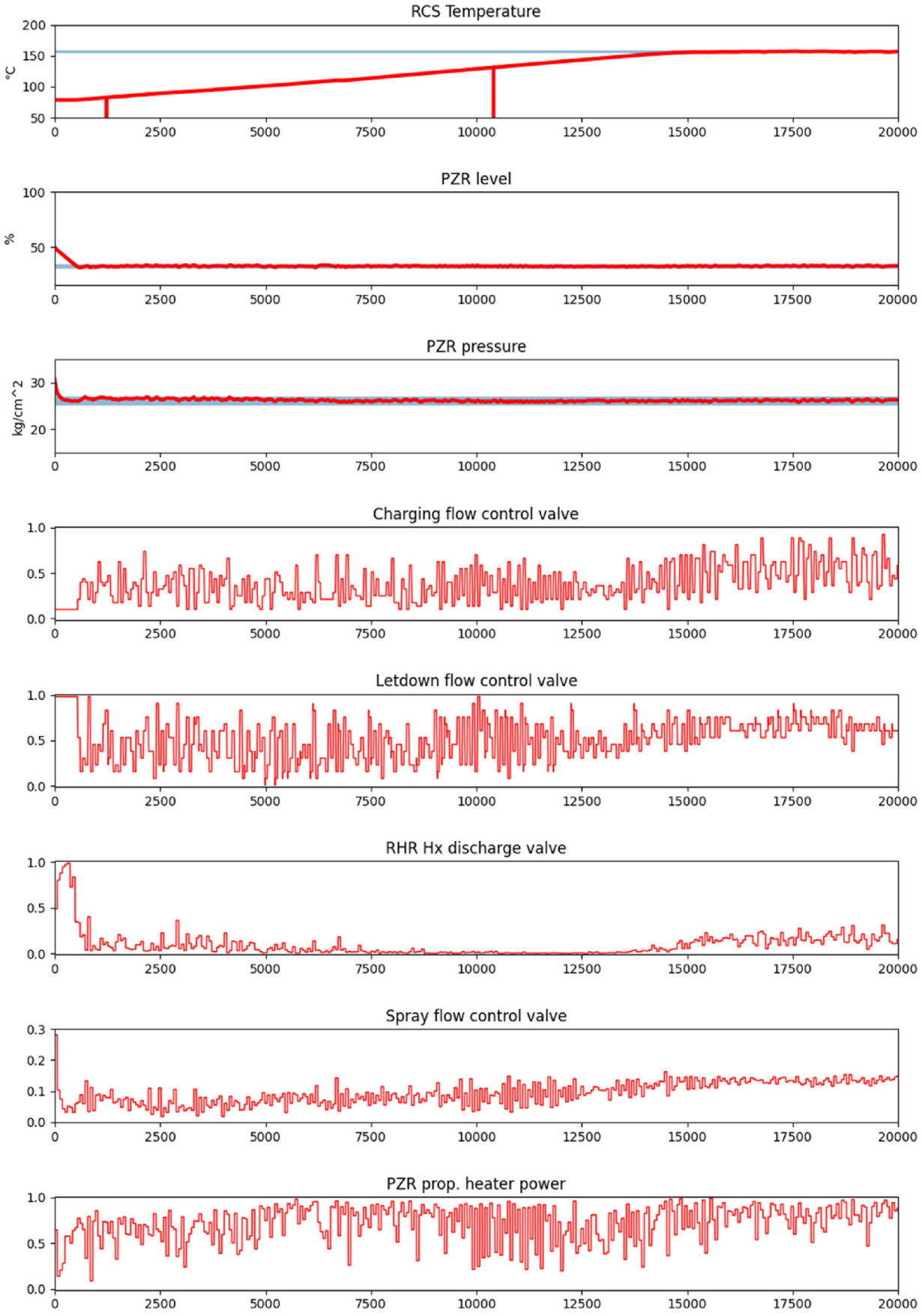

Fig. 6. Operating record up to 20000 s of the 4500th training episode. This episode succeeded in the operation. The drops in RCS temperature at 1229 s and 10411 s stem from incorrect records (i.e., 0 °C) of the simulator.

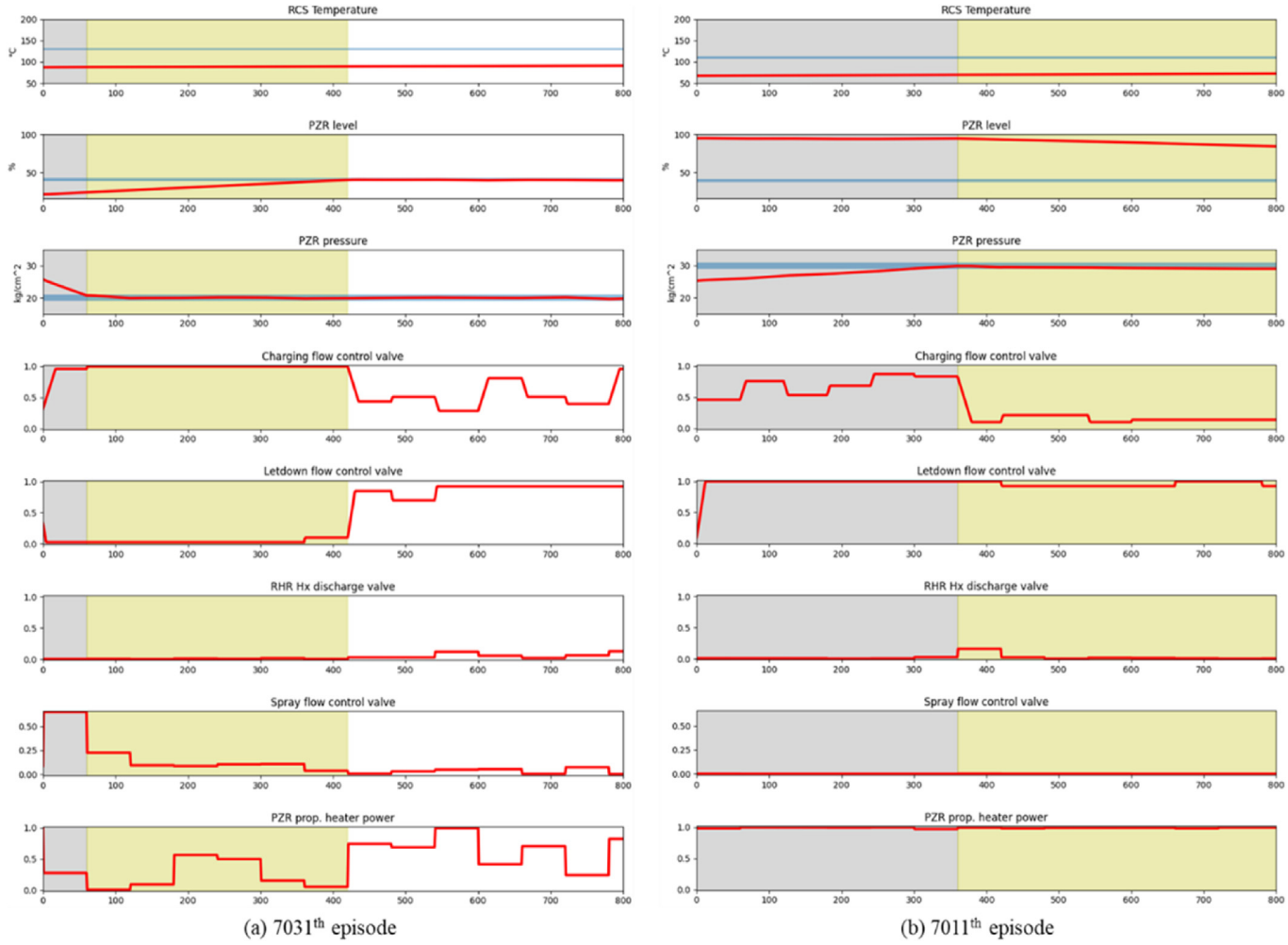

Fig. 7. Operating records up to 800 s of (a) the 7031th and (b) the 7011th episodes. The agent focused on the pressure objective (gray-shaded intervals) before satisfying the volume objective (yellow-shaded intervals). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

excess heat. As the temperature of the coolant increased, the agent also gradually opened the spray flow control valve to suppress the pressure increases resulting from the heating.

It should be noted that the trained agent occasionally violated the operating constraints even after experiencing 7000 episodes. From the 7000th to the 7194th episodes, 24 episodes were prematurely terminated due to constraints violations. Nevertheless, the operating records of the successful episodes demonstrate that DRL succeeded in training the agent to suggest optimal, continuous states of each device while managing conflicts between the multiple objectives for NPP coolant control during the startup operation.

4.3. Hands-on test for untrained situations

A demonstration program was developed using the trained DRL agent to evaluate its capability in handling previously untrained situations. This demonstration program consists of the DRL agent and a graphical user interface that enables the input of target values for each objective, as depicted in Fig. 9. The stochasticity of the action was eliminated by selecting the mean μ as the action, which fully highlights the capabilities of the agent. With this program, we conducted a hands-on test for various situations.

As shown in the test results, the DRL agent was, to a considerable extent, able to cope with untrained situations, meaning where the operating objectives had not been considered in the training session. A video of the hands-on test is available online.[1] For instance, we required the agent to reduce the temperature of the coolant. As explained in the previous sections, the DRL agent had only trained on heating the coolant. Fig. 10 is the operating record for cooldown operations. In the case of Fig. 10a, the agent was maintaining the coolant at 33 kg/cm^2, the PZR level at 80%, and the RCS temperature at 160 °C. As the objectives changed to 25 kg/cm^2, 50%, and 100 °C, the agent achieved the pressure objective first in a relatively short time by opening the spray flow valve. Concurrently, the discharge valve was also fully opened to remove the heat of the coolant. It was notable that the adjustment of the PZR level was slower, in comparison to the heating situation in Fig. 8, and was achieved at the same time as the temperature objective.

Fig. 10b illustrates the operating record for a cooldown operation when only the pressure objective was changed. It was observed that the agent utilized not only the spray flow control valve but also the charging and letdown flow control valves, resulting in a dissatisfaction of the volume objective. Nevertheless, the level was restored to the target range at the end. These cooldown operation

[1] https://github.com/JunyongBae/Soft-actor-critic-implementation-for-NPP.

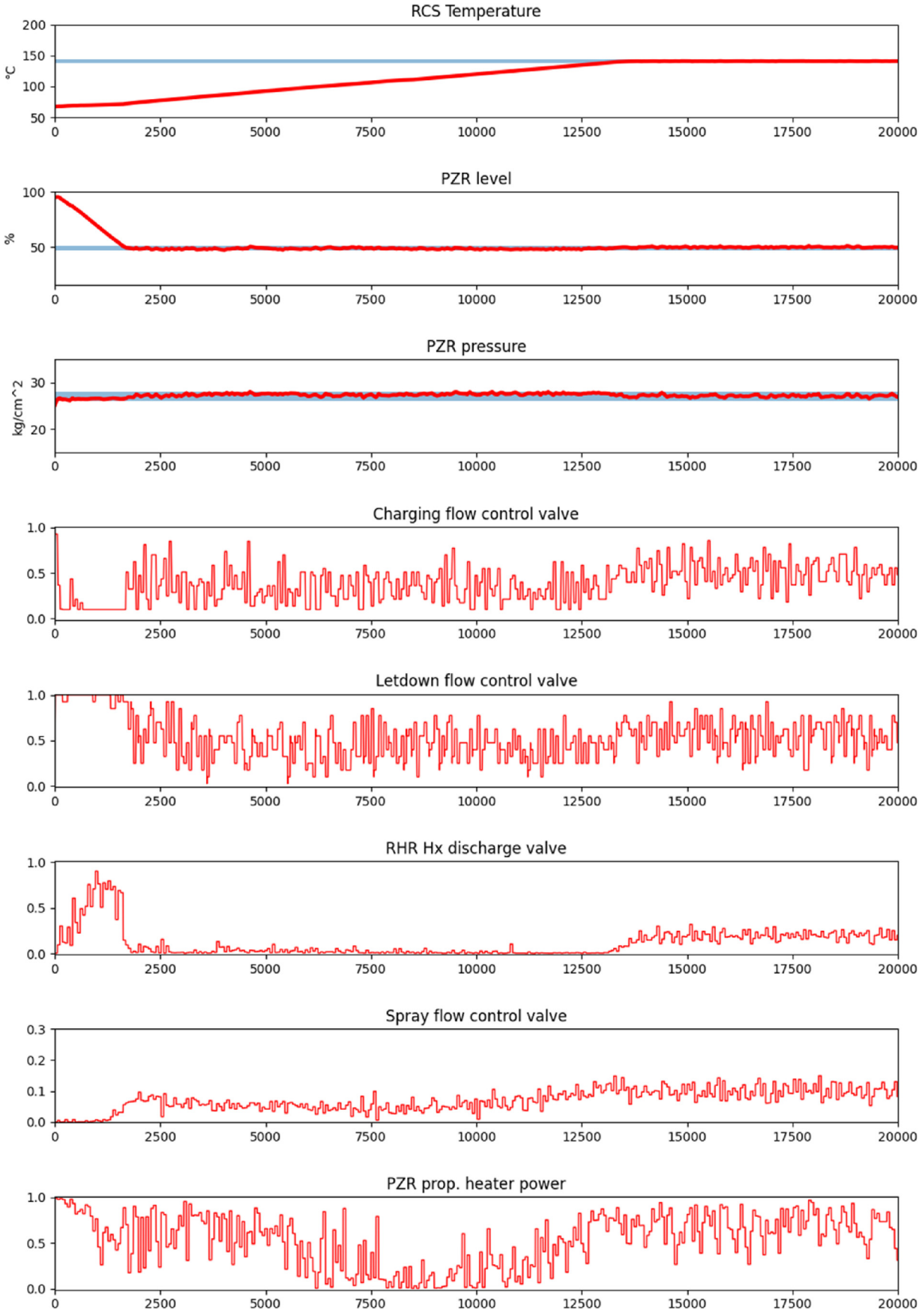

Fig. 8. Operating record up to 20000 s of the 7000th training episode. This episode succeeded in the operation.

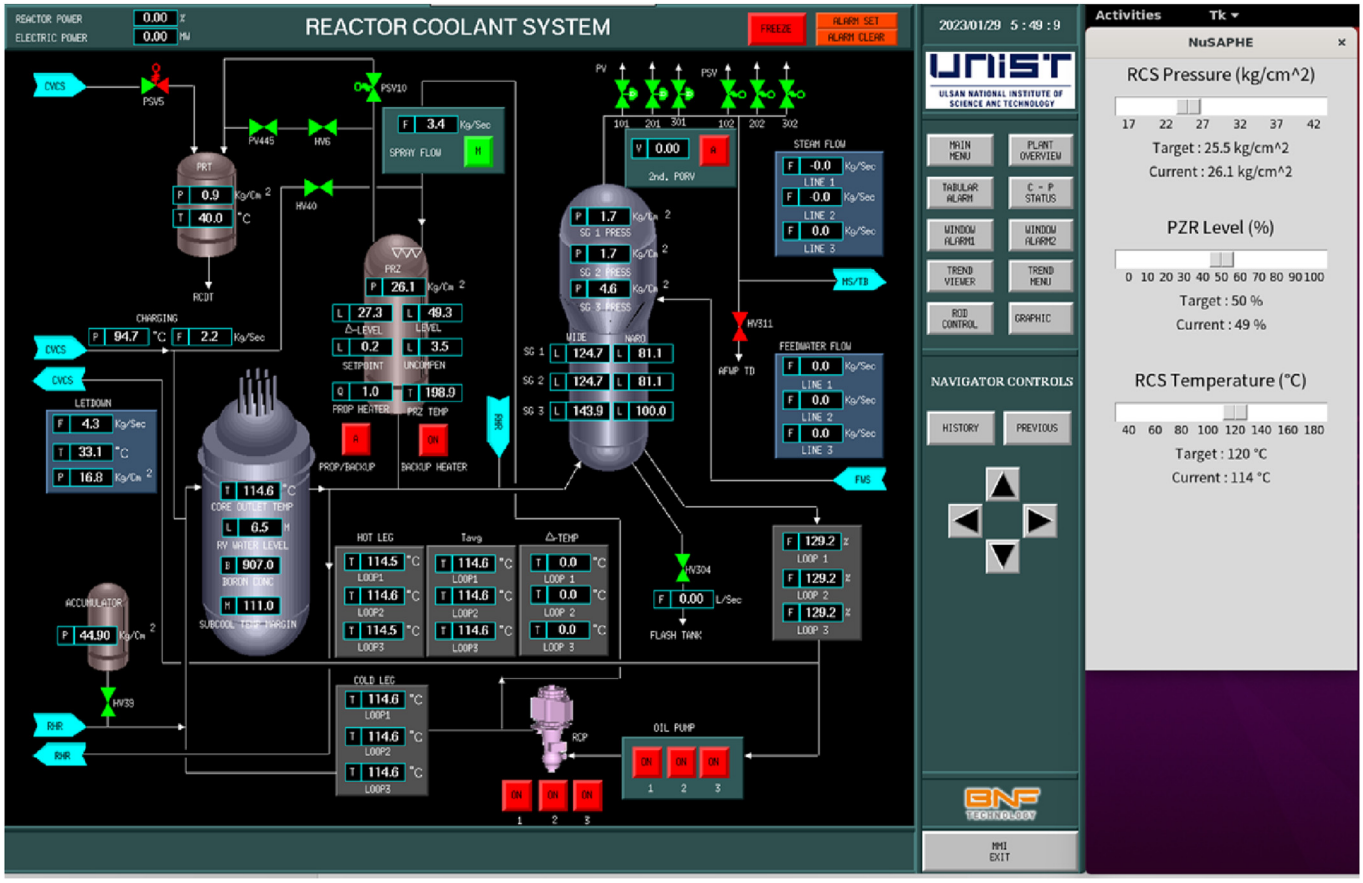

Fig. 9. Demonstration program with the trained DRL agent and user interface (right gray dialogue) connected to the CNS in a real-time manner.

examples demonstrate that the DRL agent is capable of handling previously untrained situations, but also has some limitations. It was also observed that the DRL agent experienced difficulties in adjusting both pressure and volume under high-temperature conditions.

5. Discussion

This study presents a DRL-based approach for automating operations of NPPs. Building on previous attempts to employ DRL in NPPs, the proposed approach combines SAC and HER to handle the challenges of multi-objective and multi-device operations. The use of SAC with HER eliminates the need for discrete action space and intricate reward designs for describing multiple objectives by facilitating the application of continuous action space and straightforward binary rewards. The results from the case study with pressure, volume, and temperature controls during the startup operation demonstrate the feasibility of the suggested approach. The trained DRL agent could achieve multiple objectives efficiently by prioritizing the objectives based on the time required to achieve them. Additionally, the agent showed significant success in handling untrained operating objectives.

Despite the demonstrated good performance, some substantial problems remain that should be solved before practical applications can be realized. One of the problems is verification and validation. As a safety-critical infrastructure, any software in NPPs must undergo extensive verification and validation. In this process, explaining the DRL-trained agent may be difficult due to its use of advanced deep learning models and intricate training processes. This problem could limit its application for automating NPP operations from a regulatory perspective. Therefore, it is necessary to apply techniques for explaining deep learning models, such as explainable artificial intelligence (XAI) [37], and the training process.

Additionally, the training of constraints remains a challenge in this field. As any violation of constraints in NPPs can be critical, a DRL agent for an NPP should strictly keep the constraints even if it fails to achieve the objectives. However, the current approach to training constraints relies on sparse negative feedback, given only when the constraints are violated, while positive feedback is provided more frequently as the DRL agent is trained. This may result in the agent prioritizing the objectives over the constraints. This issue was demonstrated in our case study, where a significant number of episodes were terminated prematurely due to constraint violations even after scoring high episodic rewards in other episodes. In our case study, the majority of the replay buffer consisted of transitions with positive feedback. For instance, in the later 194 episodes, 71,400 transitions that had partially or fully positive feedback were added to the replay memory (170 *episodes* [*i.e.*, 194 − 24] × 420 *transitions*/*episode* [*i.e.*, 7 *hr* × 60 *min* /*hr* × 1 *transition* /*min*]), whereas only 24 transitions with negative feedback resulting from instances of violating the constraints were added to the memory. To address this issue, further research is needed to ensure that the constraints are not violated, such as by implementing methods from the field of safe RL [38].

This study and previous studies have focused on applying DRL for the automation of NPPs. Beyond this goal, though, DRL can also

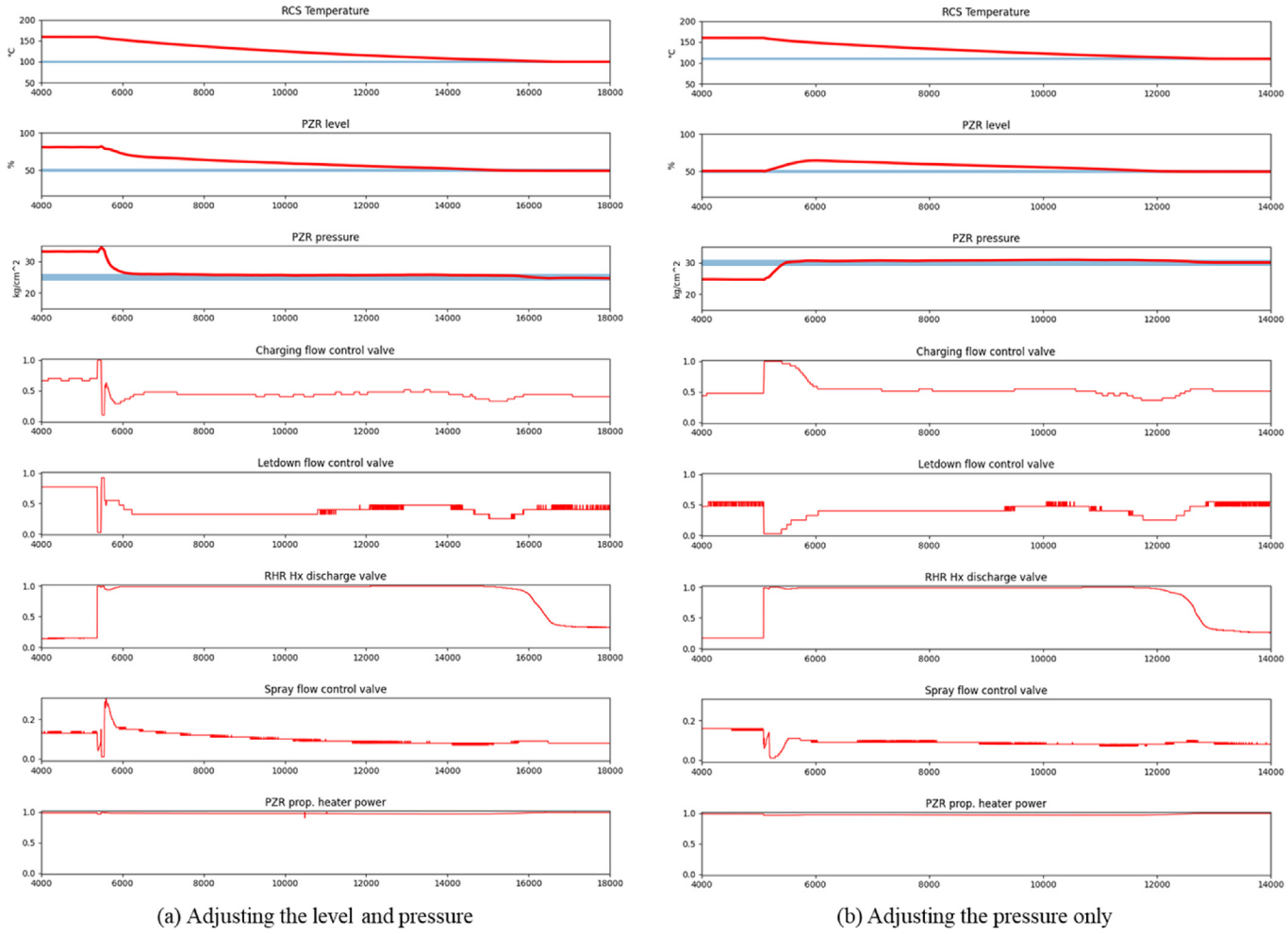

Fig. 10. Operating records of cooldown operations while being required to (a) adjust the PZR level and pressure together and (b) adjust the pressure only.

be used for various other purposes. For instance, novel operating strategies for an NPP can be explored by DRL, similar to how the game strategy in Go has evolved based on the play of a DRL-trained agent [4]. To do so, the DRL model must be capable of handling high-dimensional operations with large action and state spaces. In this study, we expanded the applicability of DRL by incorporating it in continuous action space. However, further research is needed to simplify complex high-level operations. While one possible approach is to decompose an operation into multiple task blocks and train an agent for each block, this can lead to conflicts between agents for different tasks. Kim et al. addressed this issue by prioritizing actions and estimating the consequences of possible action combinations [25] using a deep learning-based surrogate model [11,20]. As this indicates, more research is needed to effectively coordinate conflicts and fully utilize the potential of DRL.

6. Conclusion

In this study, we proposed an approach of combining SAC and HER algorithms to train a DRL agent for multi-objective and multi-device operations in NPPs. SAC allows for the adoption of continuous action space while also promising more adaptive exploration and exploitations of action and state spaces by utilizing the entropy of the policy. HER enables the use of straightforward binary reward functions while also helping the training of long-term objectives. In the case study, the agent was trained using the proposed approach and was able to successfully learn how to control the coolant pressure, volume, and temperature simultaneously by adjusting five devices. The trained agent established its own strategy for managing the multi-objectives and performed well in a hands-on test even with untrained objectives. We believe that this study extends the potential for utilizing DRL in NPP operations. The source code of our SAC and HER implementation for the NPP startup operation and hands-on test videos are available online.[1]

Author contributions

Junyong Bae – Conceptualization, Methodology, Software, Data Curation, Formal analysis, Investigation, Writing - Original Draft, Visualization. Jae Min Kim – Software, Formal analysis. Seung Jun Lee - Supervision, Funding acquisition, Writing - Review & Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2022-00165231).

References

[1] D. Lee, S. Koo, I. Jang, J. Kim, Comparison of deep reinforcement learning and PID controllers for automatic cold shutdown operation, Energies 15 (2022) 2834.
[2] C.K. Lee, D.H. Kim, I.S. Oh, J.B. Han, J.W. Shin, Y.B. Kim, Functional Descriptions for Instrumentation and Control System in Nuclear Power Plants, 2003.
[3] W. Jo, S. Jun Lee, Bayesian belief network-based human reliability analysis methodology for start-up and shutdown operations in nuclear power plants, Ann. Nucl. Energy 179 (2022), 109403.
[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge, Nature 550 (2017) 354–359.
[5] O. Vinyals, I. Babuschkin, W.M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D.H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J.P. Agapiou, M. Jaderberg, A.S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T.L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver, Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature 575 (2019) 350–354.
[6] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F.J.R. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, P. Kohli, Discovering faster matrix multiplication algorithms with reinforcement learning, Nature 610 (2022) 47–53.
[7] E. Prianto, M. Kim, J.-H. Park, J.-H. Bae, J.-S. Kim, Path planning for multi-arm manipulators using deep reinforcement learning: soft actor–critic with hindsight experience replay, Sensors 20 (2020) 5911.
[8] M.H. Lee, J. Moon, Deep reinforcement learning-based UAV navigation and control: a soft actor-critic with hindsight experience replay approach, arXiv preprint arXiv:2106.01016 (2021).
[9] Y.H. Chae, S.G. Kim, H. Kim, J.T. Kim, P.H. Seong, A methodology for diagnosing FAC induced pipe thinning using accelerometers and deep learning models, Ann. Nucl. Energy 143 (2020), 107501.
[10] J. Bae, J.W. Park, S.J. Lee, Limit surface/states searching algorithm with a deep neural network and Monte Carlo dropout for nuclear power plant safety assessment, Appl. Soft Comput. 124 (2022), 109007.
[11] S. Ryu, H. Kim, S.G. Kim, K. Jin, J. Cho, J. Park, Probabilistic deep learning model as a tool for supporting the fast simulation of a thermal–hydraulic code, Expert Syst. Appl. (2022) 200.
[12] H. Kim, J. Cho, J. Park, Application of a deep learning technique to the development of a fast accident scenario identifier, IEEE Access 8 (2020) 177363–177373.
[13] J.H. Shin, J.M. Kim, S.J. Lee, Abnormal state diagnosis model tolerant to noise in plant data, Nucl. Eng. Technol. 53 (2021) 1181–1188.
[14] G. Lee, S.J. Lee, C. Lee, A convolutional neural network model for abnormality diagnosis in a nuclear power plant, Appl. Soft Comput. 99 (2021).
[15] J. Choi, S.J. Lee, A sensor fault-tolerant accident diagnosis system, Sensors 20 (2020) 1–17.
[16] J.M. Kim, G. Lee, C. Lee, S.J. Lee, Abnormality diagnosis model for nuclear power plants using two-stage gated recurrent units, Nucl. Eng. Technol. 52 (2020) 2009–2016.
[17] Y. Chae, C. Lee, S. Han, P. Seong, Graph neural network based multiple accident diagnosis in nuclear power plants: data optimization to represent the system configuration, Nucl. Eng. Technol. (2022) 54.
[18] S. Ryu, B. Jeon, H. Seo, M. Lee, J.-W. Shin, Y. Yu, Development of Deep Autoencoder-Based Anomaly Detection System for HANARO, Nuclear Engineering and Technology, 2022.
[19] J. Ahn, S.J. Lee, Deep learning-based procedure compliance check system for nuclear power plant emergency operation, Nucl. Eng. Des. (2020) 370.
[20] J. Bae, G. Kim, S.J. Lee, Real-time prediction of nuclear power plant parameter trends following operator actions, Expert Syst. Appl. (2021) 186.
[21] J.S. Kang, S.J. Lee, Concept of an intelligent operator support system for initial emergency responses in nuclear power plants, Nucl. Eng. Technol. 54 (2022) 2453–2466.
[22] M.I. Radaideh, I. Wolverton, J. Joseph, J.J. Tusar, U. Otgonbaatar, N. Roy, B. Forget, K. Shirvan, Physics-informed reinforcement learning optimization of nuclear assembly design, Nucl. Eng. Des. 372 (2021), 110966.
[23] J. Park, T. Kim, S. Seong, S. Koo, Control automation in the heat-up mode of a nuclear power plant using reinforcement learning, Prog. Nucl. Energy 145 (2022), 104107.
[24] D. Lee, A.M. Arigi, J. Kim, Algorithm for autonomous power-increase operation using deep reinforcement learning and a rule-based system, IEEE Access 8 (2020) 196727–196746.
[25] J.M. Kim, J. Bae, S.J. Lee, Strategy to Coordinate Actions through a Plant Parameter Prediction Model during Startup Operation of a Nuclear Power Plant, Nuclear Engineering and Technology, 2022.
[26] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, Soft actor-critic algorithms and applications, arXiv preprint arXiv:1812.05905 (2018).
[27] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, W. Zaremba, Hindsight experience replay, Adv. Neural Inf. Process. Syst. (2017) 30.
[28] R.S. Sutton, A.G. Barto, Reinforcement Learning: an Introduction, MIT press, 2018.
[29] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, Human-level control through deep reinforcement learning, nature 518 (2015) 529–533.
[30] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[31] J. Bae, J. Kim, S.J. Lee, An Autonomous Pressure Controller Based on Approximation of Action Value Function, Transactions of the Korean Nuclear Society, 2020.
[32] W. Zhang, S.X. Yang, L. Yu, Soft ACTOR-CRITIC reinforcement learning for robotic manipulator with hindsight experience replay Tao yan, Int. J. Robot Autom. 34 (2019).
[33] K.-C. Kwon, J.-C. Park, C.-H. Jung, J.-S. Lee, J.-Y. Kim, Compact Nuclear Simulator and its Upgrade Plan, 1997.
[34] J. Miettinen, Development and assessment of the SBLOCA code SMABRE, in: Proceedings of the CSNI Specialists' Meeting on Small Break LOCA Analyses in LWRs, 1985, pp. 23–27. Pisa, Italy.
[35] C.R. Harris, K.J. Millman, S.J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, Array programming with NumPy, Nature 585 (2020) 357–362.
[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, Tensorflow: large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467 (2016).
[37] J.H. Shin, J. Bae, J.M. Kim, S.J. Lee, An interpretable convolutional neural network for nuclear power plant abnormal events, Appl. Soft Comput. 132 (2023), 109792.
[38] J. Garcıa, F. Fernández, A comprehensive survey on safe reinforcement learning, J. Mach. Learn. Res. 16 (2015) 1437–1480.