Reinforcement Learning for Constrained Continuous Stirred-Tank Reactors Systems

Shunhua Jiao Tachun Chen

Abstract

Reinforcement learning (RL) faces challenges in real-world process systems due to the critical need to handle state constraints, which conventional RL frameworks do not explicitly address. This research aimed to address this limitation by integrating state-dependent control constraints into RL frameworks. The proposed method was validated on a Continuous Stirred-Tank Reactor system using the PC-Gym environment, evaluating the performance of two RL algorithms, Deep Q-Network (DQN) and Proximal Policy Optimisation (PPO), in both constrained and unconstrained settings. A significant future direction involves extending the framework to stochastic environments to incorporate uncertainties in state transitions and constraints. This study highlights the potential of constrained RL as a practical solution for optimising dynamic and real-world chemical processes.

Keywords: Constrained Reinforcement Learning, Process Control Optimisation, Deep Q-Network, Proximal Policy Optimisation

1 Introduction

Sequential decision-making plays a critical role in process systems engineering (PSE), with applications ranging from scheduling and supply chain management to process control and optimisation. These tasks require a series of optimal decisions to ensure system efficiency. However, the inherent uncertainties in real-world PSE systems pose significant challenges to traditional methods like stochastic programming, which relies on predefined probability distributions and approximations. While effective in some cases, stochastic programming often becomes computationally intractable for complex systems, motivating the exploration of reinforcement learning (RL) as a viable alternative (J. Shin et al. (2019)[14]).

RL has emerged as a promising solution for addressing uncertainty in PSE by leveraging data to learn optimal control policies (J. W. Kim et al. (2020) [8]). Its effectiveness has been demonstrated in several applications, including set-point tracking and control optimisation problems (Lee and Lee (2005) [9]; Spielberg et al. (2019) [10]). Typically, RL formulates these problems within the framework of Markov Decision Processes (MDPs), which provide a structured approach to sequential decision-making (Puterman, M.L. (2014) [11]). However, a significant limitation of this framework is its inability to explicitly handle state constraints, which are common and normally necessary in real-world settings.

Common strategies to address this include penalty functions and Lagrangian multipliers, but these methods often lead to convergence instability and require extensive tuning to align with RL architectures (H.Yu et al (2021)[12]). Alternative methods, like domain restrictions and control masking, offer partial solutions but are typically limited to discrete control settings (A. Kanervisto et al. (2020) [15]).

This work aims to extend the capability of RL by addressing state constraints within continuous control environments. Inspired by the observation that MDPs can manage control constraints in a state-dependent manner, we built RL framework upon the decomposition feasibility method by M. Mowbray et al. (2024) [6], abstracting state constraints into state-dependent control constraints. The proposed method is initially developed in a deterministic setting for clarity and validation, with the ultimate goal of extending it to probabilistic environments to incorporate uncertainty effectively. This approach represents a significant step toward robust RL-based solutions for real-world PSE challenges.

This report is organized as follows: Section 2 reviews reinforcement learning and constrained RL, highlighting key frameworks and challenges. Section 3 outlines the methodology, including state constraint transformation and RL implementation. Section 4 presents a case study on a Continuous Stirred-Tank Reactor (CSTR), with results discussed in Section 5. Section 6 concludes with key findings and future directions.

2 Background

2.1 Reinforcement Learning

RL involves an agent interacting with an environment E to determine optimal actions $\mathbf{u}_t \in \mathcal{U}$ based on observed system states $\mathbf{x}_t \in \mathcal{X}$. At each discrete time step $t \in \mathbb{Z}$, the agent selects actions based on its policy $\pi(\mathbf{u}_t|\mathbf{x}_t)$, aiming to maximize an expected cumulative return $\mathbb{E}_\pi[G_t]$ from a discount factor γ and numerical

reward $R_{t+1} \in \mathbb{R}$ received from environment E after taking control $\mathbf{u}_t$:

$$\mathbb{E}_\pi [G_t] = \mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^{t-1} R_{t+1}(\mathbf{x}_t, \mathbf{u}_t) \right] \tag{1}$$

The interaction between the RL agent and the environment is assumed to follow a Markov Decision Process (MDP), defined as a five-tuple framework $< \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \gamma >$, where the transition probability of the environment is given by:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t) \tag{2}$$

with $\mathbf{f}(\cdot)$ approximating a nonlinear state space model, $\mathbf{f}(\cdot)$, and $W \sim p(\mathbf{w})$ representing general uncertain parameters. In this study, a deterministic setting was assumed:

$$p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \approx \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \tag{3}$$

2.2 Constrained Reinforcement Learning

Constrained Reinforcement Learning (CRL) has gained significant attention from the research community, as real-world dynamics often demand fast and safe control mechanisms. Previous optimisation routines in CRL required either the challenging Lagrange method for constrained Markov Decision Processes (CMDPs) or mathematical simulations for probabilistic state constraints within policy optimisation. In the context of our problem setting, we purposed a solution method for the nominal optimal control problem:

$$\pi(\cdot) := \begin{cases} \max_{\pi(\cdot)} G_t \\ \text{s.t.} \\ \mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{u}_t = \pi(\mathbf{x}_t) \\ \mathbf{u}_t \in \mathbb{U} \\ \mathbf{x}_t \in \overline{\mathbb{X}_t} \\ \overline{\mathbb{X}_t} := \{\mathbf{x}_t \in \mathbb{X} \mid \mathbf{g}_t(\mathbf{x}_t) \leq 0\} \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{4}$$

Here, $\mathbf{g}_t(x_t)$ represents the state constraints imposed at time t. This, however, highlights the challenges in CRL. Despite numerous efforts in previous works to develop more reliable state constraint approximations, these overall process often exhibit inherently non-trackable method, particularly in state constraint identification or in optimising the corresponding Lagrange function (E. Pan et al.[1], P. Petsagkourakis et al.[5]).

2.3 Constraints Identification

M. Mowbray et al. (2024) [6] presented a sampling-based decomposition approach to the feasible space via a directed acyclic graph (DAG). Instead of directly solving a large problem defined on control trajectories, $\mathbb{U}_\tau \subseteq \prod_{t=0}^{T-1} \mathcal{U}$ space, a time-wise decomposition is applied, and local subproblems are solved individually to exploit problem structure. Each pair of state and control at time t is denoted by a node in the graph and the subproblems are denoted $\mathbb{XU}_t$. Nodes are connected in a way that follows the MDP. Throughout the method, the DAG is updated, through a policy of backward propagation according to a precedence ordering on the time grid, i.e. $t \prec t+1$, with nodes subproblems solved incrementally, enforcing additional constraints on the original $\mathcal{X} \times \mathcal{U}$ space. Figure 1 visualizes the DAG, with each circle representing a node in the graph.

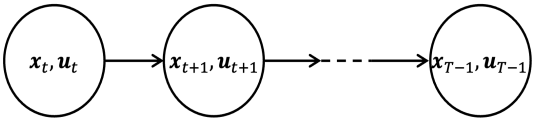

Figure 1: Example of DAG

This decomposition technique enables us to transform state constraints, which cannot be handled directly, into surrogate constraints that can be applied explicitly on control input variables $< \mathbf{x}_t, \mathbf{u}_t, t >$. In other words, it abstracts the state constraint feasibility into state dependent control constraints enforced at time t:

$$\begin{aligned} \mathbb{XU}_t &= \left\{ (\mathbf{x}_t, \mathbf{u}_t) \in \mathcal{X} \times \mathcal{U} \;\middle|\; \begin{array}{c} \mathbf{x}_t \in \overline{\mathbb{X}}_t \\ \exists (\mathbf{x}_{t+1}, \mathbf{u}_{t+1}) \in \mathbb{XU}_{t+1} : \\ \mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \end{array} \right\} \\ &\approx \{(\mathbf{x}_t, \mathbf{u}_t) \in \mathcal{X} \times \mathcal{U} | 0 \geq \overline{\mathbf{g}}_t(\mathbf{x}_t, \mathbf{u}_t)\}. \end{aligned} \tag{5}$$

In this notation, $\overline{\mathbf{g}}_t(\mathbf{x}_t, \mathbf{u}_t)$ represents a scalar function parameterising the set of safe states and controls at time t. In this work, this is identified as a neural network binary classifier.

3 Methodology

By taking advantage of the constraint transformation, state constraints can be easily imposed on any RL framework. To explore the potential and feasibility of this novel CRL framework, both Deep Q-network [3] (action-value function) and Proximal Policy Optimisation [4] (actor-critic method) were experimented with, as they are both easy to implement and widely known among researchers, while providing completely different routines in terms of agent training.

Proximal Policy Optimisation (PPO) [4] is an actor-critic method that utilizes neural network architectures for both its policy (actor) and value function (critic). The agent's objective is to learn an optimal policy through an approximated policy gradient descent. This approximation is necessary due to the complexity of computing the exact gradient and the inherent bias in the advantage function. Therefore, a generalized advantage function [13] is employed:

$$\begin{aligned} \hat{A}_t &= \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l} \\ \text{where} \quad \delta_t &= r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{6}$$

Where λ controls the balance between bias and variance of the approximation, $V(\cdot)$ is the value function (critic) of the PPO.

To facilitate exploitation, PPO samples actions from a normal distribution centered around the mean action.

$$\pi_\phi(u_t|x_t) \sim \mathcal{N}(\mu_\phi(x_t), \sigma) \tag{7}$$

Ultimately, PPO aims to maximize a clipped surrogate objective, which enhances stability by limiting the magnitude of policy updates while ensuring convergence to an optimal policy.

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t^n, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon))]$$
$$\text{where} \quad r_t(\theta) = \frac{\pi_\theta}{\pi_{\theta,old}} \tag{8}$$

Deep Q-network (DQN) [3] is a value function method that aims to learn an optimal Q-value function. By selecting the action with the highest Q-value in each state, DQN implicitly learns the optimal policy. To stabilize training and improve convergence, DQN uses a target network. This network is a copy of the behavior network that is updated partially. By using a more stable target, the Temporal difference (TD) updates become less noisy, leading to more efficient learning.

TD updates are applied to the behavior network to minimize the TD error. By minimizing this loss, the behavior network learns to predict accurate Q-values, which in turn leads to better decision-making

$$y = r + \gamma \max_{u'} Q_\theta^{\text{target}}(x', u') \tag{9}$$

$$\mathbb{E}_{(x,u,r,x')\sim\text{replay buffer}}\left[\left(Q_\theta^{\text{behaviour}}(x,u) - y\right)^2\right] \tag{10}$$

To avoid over-exploitation, DQN employs an epsilon-greedy exploration strategy, where a random action is chosen with probability ϵ.

3.1 Constrained PPO

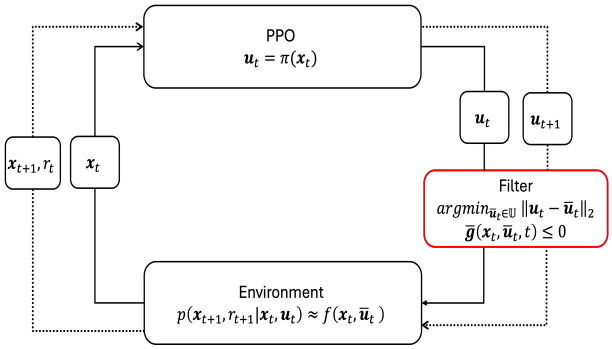

Figure 2: PPO process scheme

Vanilla PPO (J. Schulman et al. [4]) was adapted with minor changes to the logic of action selection. An additional constrained optimisation problem is solved at each action selection stage (the filter in Figure 2). The objective is to minimize the Euclidean distance between the safe control $\bar{\mathbf{u}}_t$ and the unconstrained control $\mathbf{u}_t$ generated by PPO's actor network, thereby determining the best safe control with the smallest distance using a multi-start approach. Algorithm 1 details the instruction.

Algorithm 1 PPO's Action Selection with Constraint Handling

Input: state $\mathbf{x}_t$, horizon t
Output: best safe control $\bar{\mathbf{u}}_t^*$
Sampling a list of initial guesses for $\mathbf{u}_{init} \in \mathcal{U}$.
$d^* \leftarrow \infty$.
$\bar{\mathbf{u}}_t^* \leftarrow$ None.
$\mathbf{u}_t \leftarrow \text{actor}(\mathbf{x}_t)$
for $\bar{\mathbf{u}}_t$ in $\mathbf{u}_{init}$ **do**
  Minimize$_{\bar{\mathbf{u}}_t \in \mathcal{U}} \|\mathbf{u}_t - \bar{\mathbf{u}}_t\|_2$
  subject to: $\bar{\mathbf{g}}(\mathbf{x}_t, \bar{\mathbf{u}}_t, t) \leq 0$
  $d_t \leftarrow \|\mathbf{u}_t - \bar{\mathbf{u}}_t\|_2$
  if $d_t \leq d^*$ **then**
    $d^* \leftarrow d_t$
    $\bar{\mathbf{u}}_t^* \leftarrow \bar{\mathbf{u}}_t$
  end if
end for
return $\bar{\mathbf{u}}_t^*$

3.2 Constrained DQN

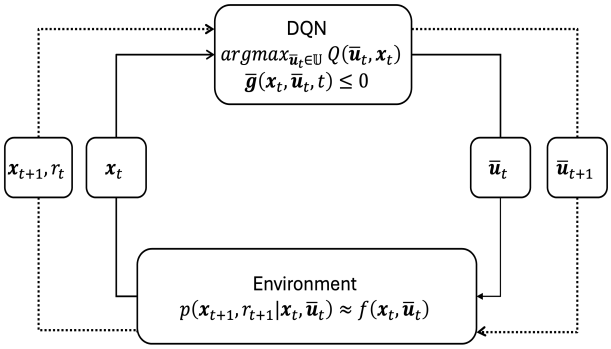

Figure 3: DQN process scheme

The general DQN framework (V. Mnih et al. [3]) requires the discretization of the continuous action space to avoid expensive computation and to identify a global optimum within the discrete action set through enumeration. However, discretized actions have inherent limitations in achieving fine control, particularly when the discrete set fails to encompass all meaningful decisions. This limitation is especially critical in the context of safe control, where precise adjustments are often necessary to ensure compliance with safety constraints. Therefore, DQN was modified to handle continuous control for this task. To select the best safe control, $\bar{\mathbf{u}}_t^*$, with the highest Q-value, a multi-start approach is employed. This involves starting from multiple initial control points and iteratively improving them using gradient-based optimisation.

Unlike PPO, which uses an actor network to directly determine the control action for a given state at

time t, DQN relies on a Q-value function $Q : f(\mathbf{u}_t, \mathbf{x}_t)$ to make decisions. This allows us to incorporate the surrogate constraint function directly into the greedy action selection process of the DQN policy. As outlined in Algorithm 2, the constrained DQN selects only safe controls when maximizing the Q function. A similar approach can be applied to the exploration phase of the epsilon-greedy method. Instead of sampling random controls from the entire designed control space, only those that satisfy the constraints are considered. Sampling can be done recursively until a feasible solution is found (see Algorithm 3).

Algorithm 2 DQN's Greedy Action Selection with Constraint Handling

Input: state $\mathbf{x}_t$, horizon t
Output: safe control $\bar{\mathbf{u}}_t$
Sampling a list of initial guesses for $\mathbf{u}_{init} \in \mathcal{U}$.
$Q^* \leftarrow -\infty$.
$\bar{\mathbf{u}}_t^* \leftarrow$ None.
for $\bar{\mathbf{u}}_t$ in $\mathbf{u}_{init}$ **do**
    $\underset{\bar{\mathbf{u}}_t \in \mathcal{U}}{\textbf{Maximize}} \quad Q(\mathbf{x}_t, \bar{\mathbf{u}}_t)$
    subject to: $\bar{\mathbf{g}}(\mathbf{x}_t, \bar{\mathbf{u}}_t, t) \leq 0$
    $Q_t \leftarrow Q(\mathbf{x}_t, \bar{\mathbf{u}}_t)$
    if $Q_t \geq Q^*$ **then**
        $Q^* \leftarrow Q_t$
        $\bar{\mathbf{u}}_t^* \leftarrow \bar{\mathbf{u}}_t$
    end if
end for
return $\bar{\mathbf{u}}_t^*$

The exploration part of the epsilon-greedy policy involves sampling from a uniform distribution within the designed control space. However, without any restrictions, a random control sampled from this distribution may lead to state-constraint violations in subsequent time steps. To address this, control samples are continuously drawn from the distribution until a sample satisfying the state constraints is obtained.

Algorithm 3 DQN's Random Sampling with Constraint Handling

Input: state $\mathbf{x}_t$, horizon t
Output: safe control $\bar{\mathbf{u}}_t$
$\mathbf{u}_t \sim$UniformSampling(-1,1) //For normalized control
while $\bar{\mathbf{g}}(\mathbf{x}_t, \mathbf{u}_t, t) \geq 0$ **do**
    $\mathbf{u}_t \sim$UniformSampling(-1,1)
end while
$\bar{\mathbf{u}}_t \leftarrow \mathbf{u}_t$
return $\bar{\mathbf{u}}_t$

4 Case study

4.1 PC-Gym

PC-Gym (Bloor et al., 2024 [2]) is an open-source environment that simulates complex chemical systems, such as Continuous Stirred-Tank Reactors (CSTRs), multistage extraction columns, and crystallization processes. These environments incorporate nonlinear dynamics, disturbance, and constraints, making them highly realistic for industrial process control research.

The CSTR system (APMonitor [7]) models a first-order irreversible reaction, with cooling provided by water flow. Two nonlinear differential equations govern its dynamics:

$$\frac{dC_A}{dt} = \frac{q}{V}(C_{Af} - C_A) - kC_A e^{-\frac{E_A}{RT}} \tag{11}$$

$$\frac{dT}{dt} = \frac{q}{V}(T_f - T) - \frac{\Delta H_R}{\rho C_p} kC_A e^{-\frac{E_A}{RT}} + \frac{UA}{\rho C_p V}(T_c - T) \tag{12}$$

Here, C_A is the concentration of reactant A, and T is the reactor temperature, which together form the state vector $x = [C_A, T]^\top \in \mathbb{R}^2$. The action variable, $u = T_c$, represents the cooling water temperature. Observations in this environment include the state values and their setpoints (if defined), structured as an array of shape (`(1, 2 + N_SP)`), where N_{SP} is the number of setpoints. For example, with setpoints for both states, the observation is `[CA, T, CA_Setpoint, T_Setpoint]`. The action space is defined as a continuous range $[290, 302]$, corresponding to a jacket temperature in Kelvin. The reward function computes the squared error between the state variables and their setpoints, scaled by a factor (`r_scale`), and sums the results into a single value. This setup enables training effective control policies for maintaining reactor stability and optimising performance.

4.2 Environment Setup

In this study, a customized simulation environment was configured using the PC-Gym framework to evaluate control performance for a CSTR system. A setpoint change is introduced at $t = 5$, adjusting the setpoint of `CA` from 0.80 to 0.85. The objective is to train an optimal policy for a RL agent that maximizes the episodic reward when interacting with the environment. For this relatively simple problem, such as setpoint tracking, the cost function to be minimized can be defined as the sum of the squared distances between the state of interest and its setpoint over a trajectory:

$$\text{Cost} = \sum_{i=0}^{T-1} r_{scale}(s_t - s_t^{SP})^2 \tag{13}$$

Table 1 summarizes the configuration of the unconstrained environment used within the PC-Gym framework.

By rolling out and visualizing the state and control trajectories from the model (Section 5.1), a box state constraint for the CSTR temperature (`T`) was determined. Subsequently, stage-1 constraint identification was performed, making the state-dependent control constraints (Section 2.3) available for stage-2 RL agent training.

Table 2 lists the additional parameters required to implement constrained dynamics control within the PC-Gym framework. Notably, our method directly handles state constraints using a 3-tuple input $\langle \mathbf{x}_t, \mathbf{u}_t, t \rangle$. This approach eliminates the need for penalty-based methods to address constraint violations. Instead, we can explicitly solve the constrained optimisation problem at each time step using a constrained nonlinear local solver like `COBYLA` with multi-restart or any suitable constrained global optimisation solver.

Parameter	Value
Time Step	`nsteps = 12`
Simulation Time	`T = 26`
Environment Model	`cstr_ode`
Reward Scaling	`r_scale = {'Ca': 5e2}`
Setpoints	`Ca` transitions from `0.80` (initial 5 steps) to `0.85` (next 7 steps)
Observation Space	`[CA, T, CA_Setpoint]`: low `[0.75, 310, 0.80]`, high `[0.90, 350, 0.85]`
Action Space	`T_c` in `[295, 305]` K
Initial Conditions	`[0.80, 329, 0.80]`
Normalization	Actions and states are normalized
Noise	Disabled
Integration Method	`casadi`

Table 1: Environment Parameters for the unconstrained CSTR

Parameter	Value
constraint	`{'T':[320,330]}`
constraint type	`{'T':['<=','>=']}`
constraint violation penalty	`False`
done on constraint violation	`False`

Table 2: Environment Parameters for the constrained CSTR

5 Results & Analysis

5.1 Unconstrained RL

Initially, unconstrained RL agents (PPO and DQN) were trained to assess the feasibility of the current environment configuration. As shown in Figure 4, the bottom subplot illustrates the control trajectory (cooling jacket temperature `Tc`), while the two upper subplots depict the state variables. The first subplot shows the state variable (concentration of `Ca`) where a setpoint change was applied from 0.8 to 0.85. The dotted line in the first subplot represents the setpoint at each time step, while the solid lines show the model's predictions. Notably, the second state variable, (temperature `T`), remained within a narrow range of 325 to 335 for both PPO and DQN.

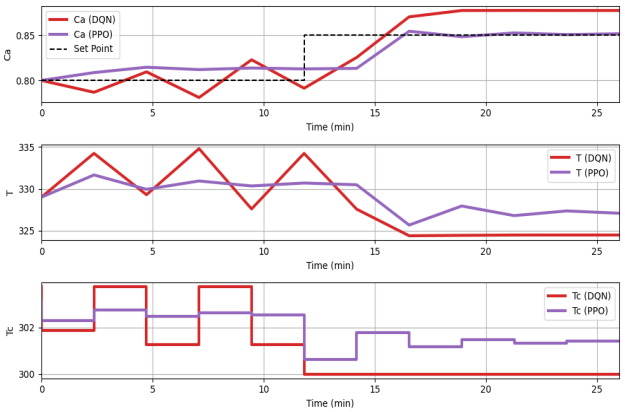

Figure 4: Unconstrained RL trajectories

To further challenge the RL agents, we introduced a box constraint on the temperature state variable `T`, limiting its range to [320, 330]. This constrained environment forced the agents to learn policies that not only achieved the desired control objective but also adhered to the imposed constraint. By imposing this constraint, we aimed to investigate how the agents would adapt their behavior and how well they could maintain state satisfaction while optimising the control objective.

5.2 Constrained RL

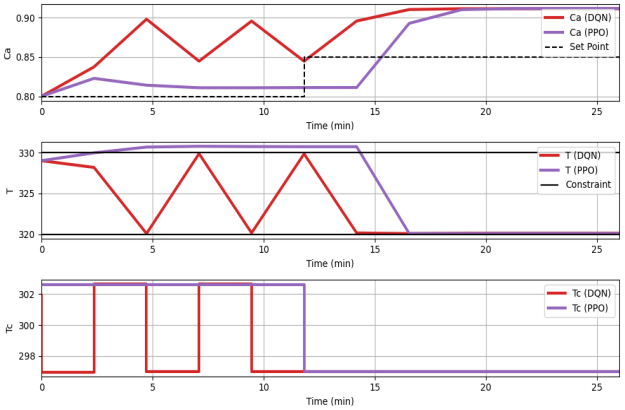

Figure 5: Constrained RL trajectories

After applying the constraint on state variable `T`, both RL agents were retrained, and their control trajectories are shown on the Figure 5. It was observed that both agents' predictions for the first state variable `Ca` deviated significantly from the setpoints. This suggests that the constraint may have been too restrictive, hindering the agents' ability to track the setpoints effectively. This is further evidenced by the trajectory of the second state variable, T, where the models frequently reached the boundary of the constraint.

Agent	Cost	Constraints violation
PPO	-5.88	5/12
DQN	-14.90	0/12

Table 3: Performance metrics for RL agents

Table 3 summarizes the agents' performance metrics in Figure 5. While PPO exhibits a slightly lower overall cost, it violates the constraint nearly half the time. This suggests that current PPO's decision-making process may not be well-suited to handling the imposed constraints, rather than a limitation in the constraint identification process itself. The primary reason for the observed constraint violations can be attributed to the exploration strategy defined defined in Eq 7. The PPO takes samples control from a normal distribution centered around the safe control $\bar{\mathbf{u}}_t$. However, there is no guarantee that this sampling process will yield control that satisfy the state constraint. When an unsafe control is obtained, the calculated gradient information can be misleading, potentially leading the PPO agent astray. This can ultimately result in the learning of a wrong policy that frequently violates the constraints.

In contrast, DQN demonstrates the effectiveness of the constraint identification process, exhibiting no constraint violations throughout the entire trajectory.

6 Conclusion and Future Work

In this research, we proposed a novel approach to apply reinforcement learning algorithms to constrained environments by leveraging the feasible space decomposition method developed by M. Mowbray [6]. This method enables the decomposition of a large joint state-control trajectory space problem $\prod_{t=0}^{T-1} \mathcal{U}_t \times \mathcal{V}_t$, into a sequence of local subproblems on time-dependent state-control spaces $\mathcal{U}_t \times \mathcal{V}_t$. The feasibility of this approach is tested by a case study involving setpoint tracking in a CSTR system within the PC-gym framework [2]. Both DQN and PPO were experimented with, and two naive ways, which are applicable to many other RL algorithms, of incorporating this state-dependent control constraint were investigated to assess the compatibility of action-value-based RL and actor-critic methods with this new approach. The success of DQN in making safe control in the constrained case study demonstrates the feasibility of the work, while further research is needed to adapt PPO to make safe control.

Despite its promise, current constrained PPO framework does not guarantee the feasibility of sampled controls. When an unsafe control is sampled, it can lead to the calculation of problematic gradient information, which hinders the learning process. A potential solution involves recursively sampling controls around a safe control until a feasible (safe) exploratory control is obtained. Future work will focus on exploring unbiased sampling strategies to further enhance the performance.

Another limitation of the current constrained PPO approach is its reliance on a biased policy gradient approximation. The filtering mechanism applied to the policy's output can be reinterpreted as an additional layer within the neural network architecture. To ensure accurate gradient information, future work will focus on identifying the exact form of this reparameterization and incorporating it as a regularization term in the loss function.

Code & Data Availability

The code used in this research is available online for public access. Specific implementations for the employed algorithms are provided in the following repositories:

`PPO` https://github.com/TACHUNC/PPO

`DQN` https://github.com/TACHUNC/DQN

Acknowledgement

The authors would like to acknowledge Dr. Max Mowbray for his consistent assistance, supervision, and insightful suggestions throughout the research process. The authors would also like to thank Prof. Benoit Chachuat, Prof. Christos Markides, and Ph.D. candidate Max Bloor for their valuable suggestions.

References

[1] E. Pan et al., "Constrained model-free reinforcement learning for process optimization," vol. 154, pp. 107462–107462, Nov. 2021.

[2] M. Bloor et al., "PC-Gym: Benchmark Environments For Process Control Problems," arXiv.org, 2024.

[3] V. Mnih et al., "Human-level Control through Deep Reinforcement Learning," Nature, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv.org, 2017.

[5] P. Petsagkourakis et al, "Chance constrained policy optimization for process control and optimization," Journal of Process Control, vol. 111, pp. 35–45, Mar. 2022.

[6] M. Mowbray et al, "A Decomposition Approach to Characterizing Feasibility in Acyclic Multi-Unit Processes," IFAC-PapersOnLine, vol. 58, no. 14, pp. 216–221, Sep. 2024.

[7] APMonitor, "pdc/CSTR_Control.ipynb at master · APMonitor/pdc," *GitHub*, 2019.

[8] H. Yoo, B. Kim, J. W. Kim, and J. H. Lee, "Reinforcement learning based optimal control of batch processes using Monte-Carlo deep deterministic policy gradient with phase segmentation," Computers & Chemical Engineering, vol. 144, p. 107133, Jan. 2021.

[9] J. M. Lee and J. H. Lee, "Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes," Automatica, vol. 41, no. 7, pp. 1281–1288, Jul. 2005.

[10] S. Spielberg, A. Tulsyan, N. P. Lawrence, P. D. Loewen, and R. Bhushan Gopaluni, "Toward self-driving processes: A deep reinforcement learning approach to control," AIChE Journal, vol. 65, no. 10, Jun. 2019.

[11] L. M. Puterman, Markov decision processes : discrete stochastic dynamic programming. New York Etc.: John Wiley & Sons, 2005.

[12] H. Yoo, V. M. Zavala, and J. H. Lee, "A Dynamic Penalty Function Approach for Constraint-Handling in Reinforcement Learning," IFAC-PapersOnLine, vol. 54, no. 3, pp. 487–491, Jan. 2021.

[13] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," arXiv:1506.02438 [cs], Oct. 2018.

[14] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, "Reinforcement Learning – Overview of recent progress and implications for process control," Computers & Chemical Engineering, vol. 127, pp. 282–294, Aug. 2019.

[15] A. Kanervisto, C. Scheller, and V. Hautamäki, "Action Space Shaping in Deep Reinforcement Learning," arXiv.org, 2020.