Constrained Model-Free Reinforcement Learning for Process Optimization

Elton Pan\textsuperscript{a}, Panagiotis Petsagkourakis\textsuperscript{b,\ast}, Max Mowbray\textsuperscript{c}, Dongda Zhang\textsuperscript{c}, Antonio del Rio-Chanona\textsuperscript{a,\ast}

\textsuperscript{a}Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, UK
\textsuperscript{b}Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, UK
\textsuperscript{c}Department of Chemical Engineering and Analytical Science, University of Manchester, UK

Abstract

Reinforcement learning (RL) is a control approach that can handle nonlinear stochastic optimal control problems. However, despite the promise exhibited, RL has yet to see marked translation to industrial practice primarily due to its inability to satisfy state constraints. In this work we aim to address this challenge. We propose an “oracle”-assisted constrained Q-learning algorithm that guarantees the satisfaction of joint chance constraints with a high probability, which is crucial for safety critical tasks. To achieve this, constraint tightening (backoffs) are introduced and adjusted using Broyden’s method, hence making them self-tuned. This results in a general methodology that can be imbued into approximate dynamic programming-based algorithms to ensure constraint satisfaction with high probability. Finally, we present case studies that analyze the performance of the proposed approach and compare this algorithm with model predictive control (MPC). The favorable performance of this algorithm signifies a step toward the incorporation of RL into real world optimization and control of engineering systems, where constraints are essential in ensuring safety.

Keywords: Machine Learning, Batch Optimization, Process Control, Q-learning, Dynamic Systems, Data-Driven Optimization

1. Introduction

The optimization of nonlinear stochastic processes poses a challenge for conventional control schemes given the requirement of an accurate process model and a method to simultaneously handle process stochasticity and satisfy state and safety constraints. Recent works have explored the application of model-free Reinforcement Learning (RL) methods for dynamic optimization of batch processes within the chemical and biochemical industries [34][28]. Many of these works demonstrate the capability of RL algorithms to learn a control law independently from a nominal process model, but negate proper satisfaction of state and safety constraints [40]. In this work, we propose constrained Q-learning, a model-free algorithm to meet the operational and safety requirements of constraint satisfaction with high probability.

\ast Corresponding authors

Email addresses: p.petsagkourakis@ucl.ac.uk (Panagiotis Petsagkourakis),
a.del-rio-chanona@imperial.ac.uk (Antonio del Rio-Chanona)

Preprint submitted to Elsevier August 2020
1.1. Model-Free Reinforcement Learning

1.1.1. Dynamic Programming and Reinforcement Learning

RL encompasses a subfield of machine learning, which aims to learn an optimal policy for a system that can be described as a Markov decision process (MDP). Importantly, MDPs assume the Markov property, such that the future transitions (dynamics) of the process are only dependent upon the current state and control action, and not upon the process history.

Dynamic programming (DP) methods provide exact solution to MDPs under knowledge of the exact process dynamics. A subset of RL algorithms are known as action-value (or Q-learning) methods. These methods are closely related to DP, but instead learn an approximate parameterization of the optimal action-value function independently of explicit knowledge of the exact dynamics descriptive of the underlying stochastic process. This eliminates the requirement for explicit assumption regarding the stochastic nature of the system. This is a particularly powerful concept in the domain of process control and optimization, given the inherent uncertainties and (slow) non-stationary dynamics often characteristic of process systems [36, 28].

The field of model-free reinforcement learning extends beyond action-value methods; two other approaches exist in the form of policy optimization [20, 37] and actor-critic [14, 21] methods. Where action-value methods explicitly learn a parameterization of the action-value function, policy optimisation methods implicitly learn the value space and instead learn and parameterize a policy directly. Typically, policy optimization approaches deploy a Monte Carlo method to gain estimate of the value-function corresponding to the current policy parameters. This provides a search direction for further policy improvement. However, policy optimization methods tend to follow on-policy learning rules, which means data collected under a given policy may only be used for one learning update before being discarded. On-policy learning combined with dependency on the use of a Monte Carlo method for evaluation of the search direction provides significant sample complexity. Actor-critic methods overcome the problems associated with sample complexity by explicitly learning a parameterization of the action-value function as well as a policy. Specifically, the use of a critic removes the dependency on Monte Carlo for evaluation of the search direction and hence removes the computational burden associated with on-policy policy optimization. In this work, we deploy an off-policy action-value method, which enables reuse of past experience for future learning updates. As such, this directs focus in the following.

1.2. Action-Value Function Approximation in Chemical Engineering

In recent years, there has been a growing interest in the development of action-value based RL controllers in the domain of chemical and biochemical processing and this is reflected by the rapid growth of literature in the field.

Actor-critic methods have been applied widely, given their explicit parameterisation of a policy (actor) and the associated sample efficiency induced via parameterisation of the action-value function (critic). In [13] an actor-critic method for pH control of wastewater from an industrial electroplating process is proposed and benchmarked against a proportional-integral-derivative (PID) controller. Similarly, in [17], the authors present an actor-critic based RL framework for optimal PID controller tuning, including a mechanism for antiwindup. In [45], the authors propose a framework to augment the actor-critic algorithm, DDPG, with model predictive control (MPC). The framework is then demonstrated on two case studies, both of which highlight the framework’s ability to provide marked improvements in the efficacy of offline policy training, relative to a vanilla implementation of DDPG. In [36], the authors identify the potential benefits of a data-driven RL controller with respect to ease of system re-identification and demonstrate the
application of an actor-critic algorithm to a number of case studies including control of a high purity distillation column.

For action-value methods to be deployed in continuous control spaces, they typically require augmentation with a further optimization method for determining the optimal control \cite{31}, and currently, very efficient optimization algorithms exist \cite{32, 25}. In \cite{32}, a Q-learning approach was applied for nanostructure and nanosurface design. Similarly, \cite{47} proposes a Q-learning method for the purpose of molecular design. In \cite{15}, the authors integrate a Q-learning method with Aspen Plus for control of a downstream separation process, demonstrating improved performance relative to open loop operation. In \cite{18} an action-value method is proposed in the context of process scheduling and in \cite{19} the authors demonstrate the application of action-value methods for the control of stochastic, nonlinear processes. The above have shown good success in chemical process optimization and control, however, for RL methods to be deployed in many instances of process engineering, they must satisfy constraints (with high probability in the stochastic setting). This is the challenge that the current work addresses.

1.3. Related Work

There exists relative inertia in application of RL to industrial control problems. Specifically, in the chemical and biochemical process industries, the development of methods to guarantee safe process operation and constraint satisfaction would enhance prospective deployment of RL-based systems in the scope of optimisation and control. The literature documents a number of approaches to safe constraint satisfaction that typically augment a pure model-free RL-based controller, with a separate system that has basis in direct optimal control. Such augmented systems are broadly constituted by barrier function \cite{7, 38, 6} and safety filter methods \cite{10, 41, 46}. The deployment of these approaches dictates a nominal description of the process dynamics, method to handle process stochasticity and often impose nontrivial policy or value learning rules. The aspect of model dependency particularly dampens the initial attraction of an RL approach within the context of process control. Other works have explored the development of methods for safe constraint satisfaction by leveraging the value framework provided by MDPs, preserving performance independent of a process model. These methods either add penalty to the original reward function (objective) for constraint violation \cite{19, 39} or augment the original MDP to take the form of a constrained MDP (CMDP) \cite{3}. The former approach introduces a number of hyperparmeters, which are typically chosen on the basis of heuristics and have bearing on policy optimality. This is also discussed in \cite{2, 12}. The latter approach is underpinned by the learning of surrogate cost functions for each individual constraint combined with appropriate adaptation of the policy \cite{2} or value learning rule. Both approaches ensure constraint satisfaction only in expectation \cite{33}, which is insufficient for control and optimization of (bio)chemical processes. As most engineering systems are safety critical, satisfaction of constraints with high probability is a necessity. In the same rationale, a Lyapunov-based approach is proposed in \cite{8}, where a Lyapunov function is found and the unconstrained policy is projected to a safety layer allowing the satisfaction of constraints in expectation. However this is not the optimal trade-off, as the closest action is not necessarily optimal. Additionally, the satisfaction of constraints in expectation is not sufficient for the most real-world applications.

1.4. Contribution

To our knowledge, no method has been proposed which achieves constraint satisfaction with high probability for action-value based methods. In this work, we propose a Q-learning
method, which guarantees constraint satisfaction with high probability. Here, we first learn an unconstrained actor and surrogate constraint action-value functions. We then subsequently construct a constrained actor action-value function as a superimposition of the unconstrained actor with the surrogate constraints. The constrained actor is iteratively tuned, as learning proceeds, via localised backoffs \[5, 26\] to penalize constraint violation. Conceptually, backoffs provide a policy variant shaping mechanism to ensure high probability satisfaction \[24\]. Tuning comprises a Monte Carlo method to estimate the probability of constraint violation under the policy combined with Broyden’s root finding method. Given the constrained actor action-value function and the fast inference associated with neural networks, efficient optimisation strategies may be deployed for determination of the optimal control. The work is arranged as follows; the problem description is formalized in section 2, the methodology proposed in section 3 and demonstrated empirically in section 4 via two benchmark case studies.

2. Problem Statement

2.1. Reinforcement Learning in Process Engineering

Using reinforcement learning directly on an industrial plant to construct an accurate controller would require prohibitive amounts of data. As such, process models must be used for the initial part of the training. It is also to ensure that safety violations do not occur in the real plant. The simplified workflow shown in Fig. 1 starts with either a randomly initialized policy or a policy that is warm-started by an existing controller and apprenticeship learning \[1\]. Preliminary training is performed using closed-loop simulations from the offline process model (notice that a stochastic model can be used). Here, the resulting control policy is a good approximation of the optimal policy, which is subsequently deployed in the real plant for further training online. Importantly, system stochasticity is accounted for and the controller will continue to adapt and learn to better control and optimize the process, hence addressing plant-model mismatch \[11, 36, 43\].

![Figure 1: Schematic representation of RL for chemical process optimization](image)

2.2. Stochastic Optimal Control Problem

We assume that the stochastic dynamic system in question follows a Markov process and transitions are given by

\[ x_{t+1} \sim p \left( x_{t+1} \mid x_t, u_t \right) , \]

\[ (1) \]
where \( p(x_{t+1}) \) is the probability density function of future state \( x_{t+1} \) given a current state \( x_t \in \mathbb{R}^n \) and control \( u_t \in \mathbb{R}^m \) at discrete time \( t \), and the initial state is given by \( x_0 \sim p(x_0) \). Without loss of generality we can write Eq. (1) as:

\[
x_{t+1} = f(x_t, u_t, d_t, p),
\]

(2)

where \( p \in \mathbb{R}^n \) are the uncertain parameters of the system and \( d \in \mathbb{R}^n \) are the stochastic disturbances. In this work, the goal is to maximize a predefined economic metric via an optimal policy subject to constraints. Consequently, this problem can be framed as an optimal control problem:

\[
\pi(\cdot) := \begin{cases}
\max_{\pi(t)} \mathbb{E}\left\{ J(x_0, \ldots, x_t, u_0, \ldots, u_t) \right\} \\
\text{s.t.} \\
x_0 \sim p(x_0) \\
x_{t+1} \sim p(x_{t+1} | x_t, u_t) \\
u_t = \pi(x_t) \\
u_t \in U \\
P\left( \bigcap_{t'=0}^{T_f} \{ x_t \in X_t \} \right) = 1 - \omega \\
\forall t \in \{0, \ldots, T_f\}
\end{cases}
\]

(3)

where \( J \) is the objective function, \( U \) is the set of hard constraints for the controls and \( X_t \) denotes constraints for states that must be satisfied. In other words,

\[
X_t = \left\{ x_t \in \mathbb{R}^n \mid g_j, t(x_t) \leq 0, j = 1, \ldots, n_g \right\},
\]

(4)

with \( n_g \) being the total number of constraints to be satisfied, and \( g_j, t \) being the \( j \)th constraint to be satisfied at time \( t \). Joint constraint satisfaction must occur at high probability of \( 1 - \omega \) where \( \omega \in [0, 1] \). Herein, we present a Q-learning algorithm that allows to obtain the optimal policy which satisfies joint chance constraints.

2.3. Q-learning

Q-learning is a model-free reinforcement learning algorithm which trains an agent to behave optimally in a Markov process \[44\]. The agent performs actions to maximize some expected reward given an objective function \( J(\cdot) \), which can be defined as

\[
J = \sum_{t=0}^{T_f} \gamma^t R_t(x_t, u_t),
\]

(5)

where \( \gamma \in [0, 1] \) is the discount factor and \( R_t \) represents the reward at time \( t \) given values \( x_t \) and \( u_t \). In the context of process control, the agent is akin to the controller, which uses a policy \( \pi(\cdot) \) to maximize the expected future reward through a feedback loop. Interaction between the agent (or controller) and system (in this case, a simulator) at each sampling time returns a value for the reward \( R \) that represents the performance of the policy.

In Q-learning, for a policy \( \pi \) an action-value function can be defined as

\[
Q^\pi(x_t, u_t) = R_{t+1} + \gamma \max_{u_{t+1}} Q(x_{t+1}, u_{t+1})
\]

(6)

with \( Q(x_{t+1}, u_{t+1}) \) being the expected sum of all the future rewards the agent receives in the resultant state \( x_{t+1} \). Importantly, the \( Q \)-value is the expected discounted reward for a given state.
and action, and therefore the optimal policy $\pi^*$ can be found using iterative updates with the Bellman equation. Upon convergence, the optimal $Q$-value $Q^*$ is defined as:

$$Q^*(x_t, u_t) = \mathbb{E}_{x_{t+1} \sim p} \left[ R_{t+1} + \gamma \max_{u_{t+1}} Q^*(x_{t+1}, u_{t+1}) \mid x_t, u_t \right]$$ (7)

$Q(x_t, u_t)$ can be represented by function approximators such as neural networks [23], Gaussian process [9] and tree-based regressors [29]. In this work, the Q-function is approximated with a deep Q-network (DQN) $Q_\theta$ parameterized by weights $\theta$. Here, the inputs specifically include the state $x_t$, the corresponding time step $t$ and control $u_t$. The DQN is trained with the use of a replay buffer that addresses the issue of correlated sequential samples [22]. Huber loss is used as the error function. Initial exploration is encouraged using an $\epsilon$-greedy policy starting with high $\epsilon$ values, which is decayed over the course of training to ensure eventual exploitation and convergence to the optimal policy.

3. Methodology

![Flow chart for (a) choosing action $u$ to satisfy constraints while maximizing reward (b) adjustment of backoffs](image)

Figure 2: Flow chart for (a) choosing action $u$ to satisfy constraints while maximizing reward (b) adjustment of backoffs

Our proposed approach can be found in Algorithm 1. The essence of the algorithm lies in training the Q-network ($Q_\theta$) using Q-values and the constraint networks ($G_{j,\phi}$) using the oracle $\hat{g}_j$, obtained via MC roll-outs using Algorithm 1. The concept of the oracle will be discussed in Section 3.1 with Eq. (8). In the MC roll-outs, actions are chosen as shown in Fig. 2 (a), by optimizing a fitness function (sub-problem in Algorithm 1) that comprises predictions by these neural networks to ensure that chosen actions satisfy constraints while maximizing reward. After these neural networks ($Q_\theta$ and $G_{j,\phi}$) have been trained using Algorithm 1, we perform constraint
tightening using backoffs as described in Fig. 2(b) such that constraint satisfaction occurs with desired probability of $1 - \omega$ as in Eq. (4), which will be discussed in Section 3.2.

Algorithm 1: Oracle-assisted Constrained Q-learning

1. Initialize replay buffer $D$ of size $s_D$ and constraint buffers $\hat{G}^j$ of size $s_G$, $j = 1, \ldots, n_g$
2. Initialize Q-network $Q_\theta$ and constraint networks $G_{j,\phi}$ with random weights $\theta$ and $\phi$, $j = 1, \ldots, n_g$
3. Initialize $\epsilon$ and backoffs $b_{j,t}$

for training iteration = 1, \ldots, M do

for episode = 1, \ldots, N do

1. Sample random minibatch of datapoints of size $G_j$ from $D$ and episode $E$.
2. Initialize state $x_0 \sim p_{x_0}(x_0)$ and episode $E$.
3. For $t = 0, \ldots, t_f$ do

   1. With probability $\epsilon$ select random control $u_t$
      otherwise select $u_t = \max_a Q_\theta (x_t, u_t) | G_{j,\phi} (x_t, u_t) + b_{j,t} \leq 0$, $j = 1, \ldots, n_g$

   (Sub-problem $^b$

   2. Execute control $u_t$ and observe reward $R_t$ and new state $x_{t+1}$

   3. Store transition $(x_t, u_t, R_t, x_{t+1})$ in $E$

1. Extract Q-values from $E$ and store datapoint $(x_t, u_t, Q_t)$ in $D$
2. Extract oracle-constraint values from $E$ using:

   $\hat{g}_{j,t} = \max_{x \geq 0} \hat{g}_{j,t'}$, $j = 1, \ldots, n_g$

3. Store datapoint $(x_t, u_t, \hat{g}_{j,t})$ in $\hat{G}^j$, $j = 1, \ldots, n_g$

1. Sample random minibatch of datapoints of size $G_j$ from $D$
2. Sample random minibatch of datapoints of size $H_j$ from $\hat{G}^j$
3. Perform a gradient-descent type step (e.g. Adam optimizer $^a$) on $Q_\theta$ and $G_{j,\phi}$ and update weights $\theta$ and $\phi$
4. Decay $\epsilon$ using $\epsilon = D_1 \epsilon$
5. Decay backoffs using $b_{j,t} = D_2 b_{j,t}$

Output: Optimal Q-network $Q_\theta^*$ and constraint networks $G_{j,\phi}$, $j = 1, \ldots, n_g$

$^a$Sub-problem: A derivative-free algorithm (e.g. evolutionary or Bayesian optimization) is used to optimize the constrained Q-function using fitness function $f(u) = Q_\theta (u) + \sum_j C_j \min (0, -(G_{j,\phi} (u) + b_{j,t}))$ where $g_{j,t}$ is the $j$th constraint violation at time $t$, and $b_{j,t}$ is the corresponding backoff. $C_j$ are large values to ensure large negative fitness values for controls that lead to constraint violation.

$^b$Any other partial, or full optimization step can be used here.

3.1. Oracle-assisted Constrained Q-learning

Q-learning, when unconstrained, may offer little practical utility in process optimization due to unbounded exploration by the RL agent. For instance, an unconstrained policy may often result in a thermal runaway leading to a safety hazard in the process. As such, herein constraints $g_{j,t}$ are incorporated through the use of an oracle $\hat{g}_{j,t}$ which is formulated as

$$\hat{g}_{j,t} = \max_{x \geq 0} \hat{g}_{j,t'}$$

(8)

with $g_{j,t}$ being the $j$th constraint to be satisfied at time $t$, and the oracle $\hat{g}_{j,t}$ is determined by the maximum level of violation to occur in all current and future time steps $t'$ in the process realization.

The intuition behind this framework is as follows: Imagine a car (agent) accelerating towards the wall with the goal of minimizing the time it takes to reach some distance from the wall.
(objective) without actually crashing into the wall (constraint). Accelerating the car without foresight causes it to go so fast that it cannot brake and stop in time, causing it to crash into the wall (constraint violated). As such, there is a need for foresight to ensure constraint satisfaction.

Effectively, the framework shown in Eq. (8) is akin to an oracle (or fortune-teller peeking into a crystal ball) advising the agent on the worst (or maximum) violation that a specific action can cause in the future given the current state. These values are easily obtained using Monte-Carlo simulations of the system. Analogous to how a Q-function that gives the sum of all future rewards, the oracle provides the worst violation in all future states if a certain action is taken by the agent, hence imbuing in the agent a sense of foresight to avoid future constraint violation.

Similar to the Q-function, constraint values are represented by neural networks $G_j$ with state and action as input features. However, the subtle difference between the two is that the state representation of the input for $G_j$ involves time-to-termination $t_f - t$ instead of time $t$ for the case of batch processes.

3.2. Constraint Tightening

To satisfy the constraints with high probability, it is required that the constraints are tightened with backoffs $b_{jt}$ as:

$$\mathcal{X}_t = \{x_t \in \mathbb{R}^n \mid g_{jt}(x_t) + b_{jt} \leq 0, j = 1, \ldots, n_g\}$$

where $b_{jt}$ are the backoffs which tighten the former feasible set $\mathcal{X}_t$ stated in Eq. (4). The result of this would be the reduction of the perceived feasible space by the agent, which consequently allows for the satisfaction of constraints. Notice that the value of the backoffs necessarily imply a trade-off: large backoff values ensure constraint satisfaction, but renders the policy over-conservative hence sacrificing performance. Conversely, smaller backoff values afford solutions with higher rewards, but may not guarantee constraint satisfaction. Therefore, the values of $b_{jt}$ are the minimum value needed to guarantee satisfaction of constraints.

To determine the desired backoffs, the cumulative distribution function (CDF) $F$ of the oracle $\hat{g}_{jt}$ is approximated using sample approximation (SSA) with $S$ Monte Carlo (MC) simulations to give its empirical cumulative distribution function (ECDF) $\hat{F}_S$, where

$$\hat{F}_S(0) \approx F(0) = \mathbb{P}(\hat{g}_{jt} \leq 0)$$

hence $\hat{F}_S(0)$ is the approximate probability for a trajectory to satisfy a constraint. We can therefore pose a root-finding problem such that we adjust the backoffs $b_{jt}$ to find:

$$\hat{F}_S(0) - (1 - \omega) = 0$$

We solve Eq. (11) via the quasi-Newton Broyden’s method [16] given its fast convergence near optimal solutions. Where $\omega$ is a tunable parameter depending on the case study, such that constraint satisfaction occurs with high probability $1 - \omega$ as shown in Eq. (3). Alternatively, the empirical lower bound of the ECDF can be forced to be $1 - \omega$, and guarantee with confidence $1 - \epsilon$ that $\mathbb{P}(\hat{g}_{jt} \leq 0) \geq 1 - \omega$.

4. Case Studies

4.1. Case Study 1

This case study pertains to the photoproduction of phycocyanin synthesized by cyanobacterium *Arthrospira platensis*. Phycocyanin is a high-value bioproduct, and serves its biological role by
increasing the photosynthetic efficiency of cyanobacteria and red algae. In addition, it is used as a natural colorant to substitute toxic synthetic pigments in cosmetic and food manufacturing. Moreover, it possesses antioxidant, and anti-inflammatory properties.

The dynamic system comprises a system of ODEs from [5] that describe the evolution of concentration \( c \) of biomass \( x \), nitrate \( N \) and product \( q \) under parametric uncertainty. The model is based on Monod kinetics, which describes the growth of microorganism in nutrient-sufficient cultures, where intracellular nutrient concentration is kept constant because of rapid replenishment. Here, a fixed volume fed-batch is assumed. The controls are light intensity \( u_1 = I \) and inflow rate \( u_2 = F_N \). The mass balance equations are as follows:

\[
\begin{align*}
\frac{d c_x}{dt} &= \frac{I}{I + k_x + I^2/k_x} c_x c_N - u_x c_x \\
\frac{d c_N}{dt} &= -\frac{Y_{N/x}}{I} \frac{I}{I + k_x + I^2/k_x} c_x c_N + F_N \\
\frac{d q}{dt} &= \frac{k_\alpha I}{I + k_s + I^2/k_s} c_x - \frac{k_d q}{c_N + K_N}
\end{align*}
\] (12)

This case study and parameter values are adopted from [5]. Uncertainty in the system is two-fold: First, the initial concentration adopts a Gaussian distribution, where \([c_{x0}, c_{N0}] \sim N([1.0, 150.0], \text{diag}(10^{-3}, 22.5)) \) and \( c_N(0) = 0 \). Second, parametric uncertainty is assumed to be: \( k_x \) (\( \mu \text{mol}/m^2/s \)) \( \sim N(178.9, \sigma^2_{k_x}) \), \( k_s \) (\( \mu \text{mol}/L \)) \( \sim N(447.1, \sigma^2_{k_s}) \), \( k_y \) (\( \mu \text{mol}/m^2/s \)) \( \sim N(393.1, \sigma^2_{k_y}) \) where the variance \( \sigma^2 \) 10% of its corresponding mean value. This type of uncertainty is common in engineering settings, as the parameters are experimentally determined, and therefore subject to confidence intervals after being calculated through nonlinear regression techniques. The objective function is to maximize the product concentration \( c_q \) at the end of the batch, hence the reward is defined as:

\[
R_{t_f} = c_{q,t_f}
\] (13)

where \( t_f \) is the terminal time step. The two path constraints are as follows: Nitrate concentration \( (c_N) \) is to remain below 800 mg/L, and the ratio of bioproduct concentration \( (c_q) \) to biomass concentration \( (c_x) \) cannot exceed 11.0 mg/g for high density biomass cultivation. These constraints can be formulated as:

\[
\begin{align*}
g_{1,t} &= c_N - 800 \leq 0 \quad \forall t \in [0, \ldots, t_f] \\
g_{2,t} &= c_q - 0.011 c_x \leq 0 \quad \forall t \in [0, \ldots, t_f]
\end{align*}
\] (14)

The control inputs are subject to hard constraints to be in the interval \( 0 \leq F_N \leq 40 \) and \( 120 \leq I \leq 400 \). The time horizon was set to 12 with an overall batch time of 240 h, and hence giving a sampling time of 20 h. The Q-network \( Q_\theta \) consists of 2 fully connected hidden layers, each consisting of 200 neurons with a leaky rectified linear unit (LeakyReLU) as activation function. The parameters used for training the agent are: \( \epsilon = 0.99, b_{1,2} = -500, b_{2,1} = -0.05, s_D = 3000, s_g = 30000, M = 2000, N = 100, G = 100, H = 500, H_2 = 1000, D_1 = 0.99 \) and \( D_2 = 0.995 \).

Upon completion of training, validation was conducted via the trained policy with respect to the constrained Q function. The policy is generated through an evolutionary strategy [35] given its nonconvex nature, as discussed in Algorithm 1 and Fig. 2(a).

After completion of training using Algorithm 1, the backoffs are adjusted to satisfy Eq. (11) with \( S = 1000 \), with backoffs at all time-steps \( t \) being constant. For simplicity, these backoffs are adjusted to ensure satisfaction of individual constraints, but it is worth noting that methods to satisfy joint chance constraints can also be implemented as shown in [27] and [5]. The constraint
Figure 3: Case Study 1: Constraints \( g_1 \), (a) and \( g_2 \), (b) when backoffs are applied (green), and when they are absent (red) with probabilities of violation \( P_v \) within the parentheses. Inset: Zoomed-in region where violation of constraints occur. Shaded areas represent the 99th to 1st percentiles.

Figure 4: Case Study 1: Constraints \( g_1 \), (a) and \( g_2 \), (b) when backoffs are applied (green), and for MPC (blue) with probabilities of violation \( P_v \) within the parentheses. Inset: Zoomed-in region where violation of constraints occur. Shaded areas represent the 99th to 1st percentiles.

Table 1: Case Study 1: Comparison of probabilities of constraint violation \( P_v \) and objective values of different algorithms

| Algorithm                        | Violation probability \( P_v \) | Objective \( c_{Q,T} \) |
|----------------------------------|-------------------------------|-------------------------|
| Oracle Q-learning with backoffs  | 0.01                          | 0.166                   |
| Oracle Q-learning without backoffs| 0.82                          | 0.169                   |
| MPC                              | 0.53                          | 0.168                   |

satisfaction is shown in Fig. 3 where the shaded areas represent the 99th to 1st percentiles. Here, we elucidate the importance of applying backoffs to the policy: As shown in Fig. 3 (a), even though it may seem at face value that \( g_1 \) values for both methods are similar, the zoomed-in region (in the inset) clearly shows that oracle Q-learning without backoffs (red) results in a high probability of constraint violation \( (P_v = 0.77) \), with parts of the red shaded regions exceeding zero.
The violation probabilities $P_v$ in Fig. 3 and 4 correspond to the fraction of 400 MC trajectories that violate a certain constraint. Gratifying, when backoffs are applied (green) in Fig. 3(a), all constraints are satisfied ($P_v = 0$), as shown by all the green shaded regions staying below zero.

In the same vein, in Fig. 3(b), applying backoffs resulted in a drastic reduction of constraint violation from $P_v = 0.24$ to $0.01$. This is expected since the backoffs are adjusted using $\omega = 0.01$ in Eq. (3). The objective value, represented by the final concentration of product $c_q$, are 0.166 and 0.169 for oracle Q-learning with and without backoffs, respectively. Consequently, this indicates that a small compromise in objective value can result in high probability of constraint satisfaction, where violation probability is reduced from 0.82 to 0.01 (in boldface) upon applying backoffs as shown in Table 1.

In addition, the performance of the oracle Q-learning algorithm with backoffs has been compared with that of nonlinear MPC using the nominal parameters of the model, which is one of the main process control techniques used in chemical process optimization and hence serves as an important benchmark.

Although MPC achieves a slightly higher objective value (Table 1), it fares poorly in terms of constraint satisfaction as shown in blue Fig. 4(a) and (b) where probabilities of violation are 12 and 53% for $g_1$ and $g_2$, respectively. This is unsurprising, since MPC is only able to satisfy constraints in expectation, which means that in a stochastic system, loosely speaking, violation occurs 50% of the time. On the other hand, oracle Q-learning with backoffs violated a constraint only 1% of the time (boldface in Table 1). Therefore, it is clear that this algorithm offers a more effective means of handling constraints compared to MPC.

4.2. Case Study 2

The second case study involves a challenging semi-batch reactor adopted from [4], with the following chemical reactions in the reactor catalyzed by $\text{H}_2\text{SO}_4$:

$$2 \text{A} \xrightarrow{k_1\text{A}} \text{B} \xrightarrow{k_2\text{B}} 3 \text{C}$$  \hspace{1cm} (15)

Here, the reactions are first-order. Reactions (1) and (2) are exothermic and endothermic, respectively. The temperature is controlled by a cooling jacket. The controls are the flowrate of reactant A entering the reactor and the temperature of the cooling jacket $T_0$. Therefore, the state is represented by the concentrations of A, B, and C in mol/L ($c_A$, $c_B$, $c_C$), reactor temperature in K ($T$), and the reactor volume in L ($Vol$). The model of the physical system can be found in [4].

The objective function is to maximize the amount of product ($c_C \cdot Vol$) at the end of the batch. Two path constraints exist. Firstly, the reactor temperature needs to be below 420 K due to safety reasons and secondly, the reactor volume is required to be below the maximum reactor capacity of 800 L and therefore:

$$g_{1,t} = T - 420 \leq 0 \quad \forall t \in \{0, \ldots, t_f\}$$
$$g_{2,t} = Vol - 800 \leq 0 \quad \forall t \in \{0, \ldots, t_f\}$$  \hspace{1cm} (16)

The ODEs describing the evolution of the system can be found in [4]. The time horizon is fixed to 10 with an overall batch time of 4 h, therefore the sampling time is 0.4 h. Parametric uncertainty is set as: $\theta_1 \sim \mathcal{N}(4, 0.1)$, $A_2 \sim \mathcal{N}(0.08, 1.6 \times 10^{-2})$, $\theta_4 \sim \mathcal{N}(100, 5)$. The initial concentrations of A, B and C are set to zero. The initial reactor temperature and volume are 290 K and 100 L, respectively.

In this case study, due to its more challenging nature in terms of constraint satisfaction compared to the first case study, the backoffs have been adjusted to satisfy Eq. (11) using $\omega = 0.1$.
in Eq. (3). We observe that backoffs again proved to be necessary to ensure high probability of constraint satisfaction. From the inset of Fig. 5(a), we can see that without backoffs the policy violates $g_1$ 41% of the time, and this probability is reduced to 9% when backoffs are applied. The same applies for $g_2$ in Fig. 5(b) where $P_v$ is completely eliminated from 3% to 0% using backoffs.

Figure 5: Case Study 2: Constraints $g_1$, (a) and $g_2$, (b) when backoffs are applied (green), and when they are absent (red) with probabilities of violation $P_v$ within the parentheses. Inset: Zoomed-in region where violation of constraints occur. Shaded areas represent the 95th-5th percentiles for (a) and 99th-1st percentiles for (b).

Figure 6: Case Study 2: Constraints $g_{1,t}$ (a) and $g_{2,t}$ (b) when backoffs are applied (green), and for MPC (blue) with probabilities of violation $P_v$ within the parentheses. Inset: Zoomed-in region where violation of constraints occur. Shaded areas represent the 95th-5th percentiles for (a) and 99th-1st percentiles for (b).

To compare the performance of MPC with oracle Q-learning with backoffs in the context of this case study, we consider two cases: First, in a deterministic system, MPC is found to be more efficient and gives solutions of much higher objective values. However, chemical systems are rarely deterministic in nature, hence limiting the applicability of MPC. Second, in the stochastic system, MPC often struggles in terms of constraint handling. This can be clearly seen in Fig. 6(a), where the MPC trajectories only satisfy $g_1$ in expectation (blue line), hence resulting in
Table 2: Case Study 2: Comparison of probabilities of constraint violation $P_v$ and objective values of different algorithms

| Algorithm                  | Violation probability ($P_v$) | Objective ($c_{C,t} \cdot Vol_t$) |
|----------------------------|-------------------------------|-----------------------------------|
| Oracle Q-learning with backoffs | 0.09                          | 532                               |
| Oracle Q-learning without backoffs | 0.44                          | 680                               |
| MPC                        | 0.66                          | 714                               |

High levels of violations (66%). Intriguingly, for $g_2$ the MPC trajectory in Fig. 6 displayed little variation, resulting in only small probability of violation (6%).

In terms of objective values, unlike the first case study, oracle Q-learning with backoffs saw a significant decrease in objective value in Table 2 after applying backoffs. This is expected because we further restrict the feasible space of the controller leading to a more conservative solution, hence exhibiting a trade-off between constraint satisfaction and objective value.

This trade-off is justified as the MPC solution results in 66% probability of constraint violation (Table 2). In the context of a chemical plant, the MPC solution is infeasible due to the high risk. The adoption of RL in such industries necessitates that these probabilities are minimized as safety is of utmost importance in chemical engineering.

On the other hand, for oracle Q-learning, it can be seen that the probability of constraint violation has been significantly improved from 66% (for MPC) to 9% (boldface in Table 2). Clearly, oracle Q-learning offers an effective means of not only satisfying constraints in expectation (green lines in Fig. 5), but more importantly with high probability (all green shaded areas below zero).

However, it is worth noting that this algorithm is based on Q-learning, which is expected to take longer time to train, particularly because it requires backoffs to be tuned. This is a direct consequence of shifting the computation time from online to offline. Indeed, such a tradeoff can be justified as this guarantees robust constraint satisfaction online with fast computation time, which is crucial in many safety critical engineering applications.

5. Conclusions

In this paper we propose a new reinforcement learning methodology for finding a controller policy that can satisfy constraints with high probability in stochastic and complex process systems. The proposed algorithm - oracle-assisted constrained Q-learning - uses constraint tightening by applying backoffs to the original feasible set. Backoffs restrict the perceived feasible space by the controller, hence allowing guarantees on the satisfaction of chance constraints. Here, we find the smallest backoffs (least conservative) that still guarantee the desired probability of satisfaction by solving a root-finding problem using Broyden’s method. Results show that our proposed methodology compares favorably to model predictive control (MPC), a benchmark control technique commonly used in the industry, in terms of constraint handling. This is expected since MPC guarantees constraint satisfaction only in expectation (loosely speaking constraints are satisfied only 50% of the time), while our algorithm ensures constraint satisfaction with probabilities as high as 99% as shown in the case studies. Being able to solve constraint policy optimization problems with high probability constraint satisfaction has been one of the main hurdles of the widespread use of RL in engineering applications. The promising performance of this algorithm is an encouraging step towards applying RL to the real world, where constraints on policies are absolutely critical due to safety reasons.
References

[1] P. Abbeel, A. Y. Ng, 2004. Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the twenty-first international conference on Machine learning. p. 1.

[2] J. Achiam, D. Held, A. Tamar, P. Abbeel, 2017. Constrained policy optimization. arXiv preprint arXiv:1705.10528.

[3] E. Altman, 1999. Constrained Markov decision processes. Vol. 7. CRC Press.

[4] E. Bradford, L. Insland, 2018. Economic stochastic model predictive control using the unscented kalman filter. IFAC-PapersOnLine 51 (18), 417–422.

[5] E. Bradford, L. Insland, D. Zhang, E. A. del Rio Chanona, 2020. Stochastic data-driven model predictive control using gaussian processes. Computers & Chemical Engineering 139, 106844.

[6] R. Cheng, G. Orosz, R. M. Murray, J. W. Burdick, 2019. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. pp. 3387–3395.

[7] J. Choi, F. Castaño, C. J. Tomlin, K. Sreenath, 2020. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions. arXiv preprint arXiv:2004.07584.

[8] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, M. Ghavamzadeh, 2019. Lyapunov-based safe policy optimization for continuous control.

URL http://arxiv.org/abs/1901.10031

[9] G. Chowdhary, M. Lin, R. Grande, Y. Walsh, J. How, L. Carin, 2014. Off-policy reinforcement learning with gaussian processes. IEEE/CAA Journal of Automatica Sinica 1 (3), 227–238.

[10] G. Dalal, K. Djivotham, M. Vecerík, T. Hester, C. Padararu, Y. Tassa, 2018. Safe exploration in continuous action spaces. CoRR abs/1801.08757.

URL http://arxiv.org/abs/1801.08757

[11] E. A. del Rio-Chanona, P. Petsagkourakis, E. Bradford, J. E. A. Graciano, B. Chachuat, 2020. Modifier adaptation meets bayesian optimization and derivative-free optimization. arXiv preprint arXiv:2009.08819.

[12] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, A. Madry, 2020. Implementation matters in deep policy gradients: A case study on ppo and trpo. arXiv preprint arXiv:2005.12729.

[13] D. A. Goulart, R. D. Pereira, 2020. Autonomous ph control by reinforcement learning for electroplating industry wastewater. Computers & Chemical Engineering, 106909.

[14] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290.

[15] S. Hwangbo, G. Sin, 2020. Design of control framework based on deep reinforcement learning and monte-carlo sampling in downstream separation. Computers & Chemical Engineering, 106910.

[16] C. T. Kelley, 1995. Iterative methods for linear and nonlinear equations. SIAM.

[17] N. P. Lawrence, G. E. Stewart, P. D. Loewen, M. G. Forbes, J. U. Backstrom, R. B. Gopaluni, 2020. Optimal pid and antwindup control design as a reinforcement learning problem. arXiv preprint arXiv:2005.04539.

[18] J. H. Lee, J. M. Lee, 2006. Approximate dynamic programming-based approach to process control and scheduling. Computers & chemical engineering 30 (10-12), 1603–1618.

[19] J. M. Lee, J. H. Lee, 2005. Approximate dynamic programming-based approaches for input-output data-driven control of nonlinear processes. Automatica 41 (7), 1281–1288.

[20] J. Lehman, J. Chen, J. Clune, K. O. Stanley, 2018. Es is more than just a traditional finite-diﬀerence approximator. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 450–457.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[22] L. J. Lin, 1993. Reinforcement learning for robots using neural networks. Tech. rep., Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

[24] A. Y. Ng, D. Harada, S. Russell, 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In: ICML. Vol. 99. pp. 278–287.

[25] J. Nocedal, S. Wright, 2006. Numerical optimization. Springer Science & Business Media.

[26] J. A. Paulson, A. Mesbah, 2018. Nonlinear model predictive control with explicit backoffs for stochastic systems under arbitrary uncertainty. IFAC-PapersOnLine 51 (20), 523–534.

[27] P. Petsagkourakis, I. O. Sandoval, E. Bradford, F. Galvinin, D. Zhang, E. A. del Rio-Chanona, 2020. Chance constrained policy optimization for process control and optimization. arXiv preprint arXiv:2008.00030.

[28] P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, E. A. del Rio-Chanona, 2020. Reinforcement learning for batch bioprocess optimization. Computers & Chemical Engineering 133, 106649.

[29] L. D. Pyeatt, A. E. Howe, et al., 2001. Decision tree function approximation in reinforcement learning. In: Proceed-
nings of the third international symposium on adaptive systems: evolutionary computation and probabilistic graphical models. Vol. 2. Cuba, pp. 70–77.

[30] M. Rafiei, L. A. Ricardez-Sandoval, 2018. Stochastic back-off approach for integration of design and control under uncertainty. Industrial & Engineering Chemistry Research 57 (12), 4351–4365.

[31] M. Ryu, Y. Chow, R. Anderson, C. Tjandraatmadja, C. Boutilier, 2019. Caql: Continuous action q-learning. arXiv preprint arXiv:1909.12397.

[32] I. Sajedian, T. Badloe, J. Rho, 2019. Optimisation of colour generation from dielectric nanostructures using reinforcement learning. Optics express 27 (4), 5874–5883.

[33] H. Satija, P. Amortila, J. Pineau, 2020. Constrained markov decision processes via backward value functions. arXiv preprint arXiv:2008.11811.

[34] V. Singh, H. Kodamana, 2020. Reinforcement learning based control of batch polymerisation processes. IFAC-PapersOnLine 53 (1), 667–672.

[35] A. Slowik, H. Kwasnicka, 2020. Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications, 1–17.

[36] S. Spielberg, A. Turysan, N. P. Lawrence, P. D. Loewen, R. Bhushan Gopaluni, 2019. Toward self-driving processes: A deep reinforcement learning approach to control. AIChE Journal 65 (10), e16689.

[37] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, 2000. Policy gradient methods for reinforcement learning with function approximation. In: Advances in neural information processing systems. pp. 1057–1063.

[38] A. Taylor, A. Singletary, Y. Yue, A. Ames, 2020. Learning for safety-critical control with control barrier functions. In: Learning for Dynamics and Control. PMLR, pp. 708–717.

[39] C. Tessler, D. J. Mankowitz, S. Mannor, 2018. Reward constrained policy optimization. arXiv preprint arXiv:1805.11074.

[40] N. J. Treloar, A. J. Fedorec, B. Ingalls, C. P. Barnes, 2020. Deep reinforcement learning for the control of microbial co-cultures in bioreactors. PLOS Computational Biology 16 (4), e1007783.

[41] K. P. Wabersich, M. N. Zeilinger, 2018. Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning. arXiv preprint arXiv:1812.05506.

[42] A. Wächter, 2002. An interior point algorithm for large-scale nonlinear optimization with applications in process engineering. Ph.D. thesis, PhD thesis, Carnegie Mellon University.

[43] Z. Wang, H.-X. Li, C. Chen, 2019. Incremental reinforcement learning in continuous spaces via policy relaxation and importance weighting. IEEE Transactions on Neural Networks and Learning Systems.

[44] C. J. Watkins, P. Dayan, 1992. Q-learning. Machine learning 8 (3-4), 279–292.

[45] H. Xie, X. Xu, Y. Li, W. Hong, J. Shi, 2020. Model predictive control guided reinforcement learning control scheme. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

[46] M. Zanon, S. Gros, 2019. Safe reinforcement learning using robust mpc. arXiv preprint arXiv:1906.04005.

[47] Z. Zhou, S. Kearnes, L. Li, R. N. Zare, P. Riley, 2019. Optimization of molecules via deep reinforcement learning. Scientific reports 9 (1), 1–10.