Constrained Q-Learning for Batch Process Optimization

Elton Pan [*] **Panagiotis Petsagkourakis** [**] **Max Mowbray** [***] **Dongda Zhang** [***] **Antonio del Rio-Chanona** [*]

[*] *Centre for Process Systems Engineering, Department of Chemical Engineering, Imperial College London, London, SW7 2BU UK (e-mail: {elton.pan17, a.del-rio-chanona}@imperial.ac.uk)*
[**] *Centre for Process Systems Engineering, Department of Chemical Engineering, University College London, London, WC1E 7JE UK (e-mail: p.petsagkourakis@ucl.ac.uk)*
[***] *Department of Chemical Engineering and Analytical Science, University of Manchester, Manchester, M1 3AL UK (e-mail: {max.mowbray, dongda.zhang}@manchester.ac.uk)*

Abstract: Chemical process optimization and control often require satisfaction of constraints for safe operation. Reinforcement learning (RL) has been shown to be a powerful control technique that can handle nonlinear stochastic optimal control problems. Despite this promise, RL has yet to see significant translation to industrial practice due to its inability to satisfy state constraints. This work aims to address this challenge. We propose an "oracle"-assisted constrained Q-learning algorithm that guarantees the satisfaction of joint chance constraints with high probability, which is required for safety critical tasks. To that end, constraint tightening (backoffs) are introduced, which are adjusted using Broyden's method, hence making the backoffs self-tuned. This results in a general methodology that can be integrated into approximate dynamic programming-based algorithms to guarantee constraint satisfaction with high probability. Finally, a case study is presented to compare the performance of the proposed approach with that of model predictive control (MPC). The superior performance of the proposed algorithm, in terms of constraint handling, signifies a step toward the use of RL in real world optimization and control of systems, where constraints are critical in ensuring safety.

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

1. INTRODUCTION

The online optimization and control of chemical and biochemical processes, provides significant improvements in operative sustainability. Currently, the optimization of nonlinear stochastic processes poses a challenge for conventional control schemes given the requirement of an accurate process model and 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 online dynamic optimization of batch processes within the chemical and biochemical industries (Singh and Kodamana (2020); Petsagkourakis et al. (2020b)). Many of these works demonstrate the capability of RL algorithms to learn a control law independently of a nominal process model, but negate proper satisfaction of state and safety constraints (Treloar et al. (2020)). In this work, we use constrained Q learning, a model-free algorithm to meet the operational and safety requirements of constraint satisfaction with high probability.

Despite the interest of the academic community in the application of RL for data-driven control, there exists relative inertia in practical and industrial implementation. 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 (Shin et al. (2019)). The literature documents a number of approaches to constraint satisfaction, which typically either add penalty to the original reward function for constraint violation (Lee and Lee (2005); Tessler et al. (2018)) or augment the original MDP to take the form of a constrained MDP (CMDP) (Altman (1999); Liu et al. (2019)). 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 by Achiam et al. (2017) and Engstrom et al. (2020). The latter approach is underpinned by the learning of surrogate cost functions for each individual constraint combined with appropriate adaptation of the policy (Achiam et al. (2017); Liu et al. (2019)) or value learning rule (Ge et al. (2019)). Both approaches ensure constraint satisfaction only in expectation, 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 (Petsagkourakis et al. (2020a)).

To our knowledge, no method has been proposed which achieves such constraint satisfaction for pure action-value based methods. In this work, we propose a Q-learning method, which guarantees constraint satisfaction with high probability. Here, we 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 (Bradford et al. (2020)) to penalize constraint violation. Conceptually, backoffs provide a policy variant shaping mechanism to ensure high probability satisfaction (Ng et al. (1999)). Tuning comprises a Monte Carlo method to estimate the probability of constraint violation under the policy combined with Broyden's root finding method. The optimal greedy constrained policy is optimized through an evolutionary strategy (Slowik and Kwasnicka (2020)) given its nonconvex nature. The work is arranged as follows; the problem description is formalised in section 2, the methodology proposed in section 3 and demonstrated empirically in section 4 via a benchmark case study.

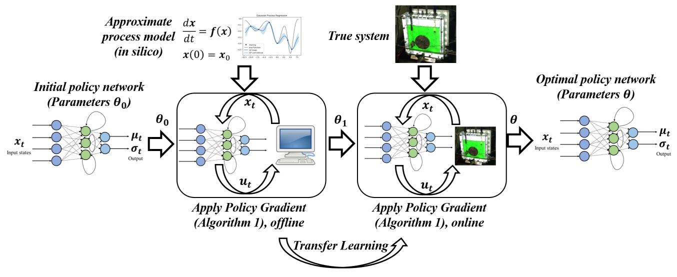
Approximate process model (in silico)
$\frac{dx}{dt} = f(x)$
$x(0) = x_0$
True system
Initial policy network (Parameters θ_0)
x_t
Input states
μ_t
σ_t
Output
θ_0
x_t
u_t
θ_1
θ
Optimal policy network (Parameters θ)
Apply Policy Gradient (Algorithm 1), offline
Apply Policy Gradient (Algorithm 1), online
Transfer Learning

Fig. 1. Schematic representation of RL for chemical process optimization

2. REINFORCEMENT LEARNING

2.1 RL in process engineering

Using RL 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. The 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 (Abbeel and Ng (2004)). Preliminary training is performed using closed-loop simulations from the offline process model. 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 (Spielberg et al. (2019); Wang et al. (2019)).

2.2 Problem statement

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

$$\mathbf{x}_{t+1} \sim p\left(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t\right), \tag{1}$$

where $p(\mathbf{x}_{t+1})$ is the probability density function of future state $\mathbf{x}_{t+1}$ given a current state $\mathbf{x}_t \in \mathbb{R}^{n_x}$ and control $\mathbf{u}_t \in \mathbb{R}^{n_u}$ at discrete time t, and the initial state is given by $\mathbf{x}_0 \sim p_{\mathbf{x}_0}(\cdot)$. Without loss of generality we can write (1) as:

$$\mathbf{x}_{t+1} = f\left(\mathbf{x}_t, \mathbf{u}_t, \mathbf{d}_t, \mathbf{p}\right), \tag{2}$$

where $\mathbf{p} \in \mathbb{R}^{n_p}$ are the uncertain parameters of the system and $\mathbf{d}_t \in \mathbb{R}^{n_d}$ 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:

$$\pi(\cdot) := \begin{cases} \max\limits_{\pi(\cdot)} \mathbb{E}\left\{J\left(\mathbf{x}_0, \ldots, \mathbf{x}_{t_f}, \mathbf{u}_0, \ldots, \mathbf{u}_{t_f}\right)\right\} \\ \text{s.t.} \\ \mathbf{x}_0 \sim p_{\mathbf{x}_0}\left(\mathbf{x}_0\right) \\ \mathbf{x}_{t+1} \sim p\left(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t\right) \\ \mathbf{u}_t = \pi\left(\mathbf{x}_t\right) \\ \mathbf{u}_t \in \mathbb{U} \\ \mathbb{P}\left(\bigcap\limits_{t=0}^{t_f} \left\{\mathbf{x}_t \in \mathbb{X}_t\right\}\right) = 1 - \omega \\ \forall t \in \left\{0, \ldots, t_f\right\} \end{cases} \tag{3}$$

where J is the objective function, $\mathbb{U}$ is the set of hard constraints for the controls and $\mathbb{X}_t$ denotes constraints for states that must be satisfied. In other words,

$$\mathbb{X}_t = \left\{\mathbf{x}_t \in \mathbb{R}^{n_x} \mid g_{j,t}\left(\mathbf{x}_t\right) \leq 0, j = 1, \ldots, n_g\right\}, \tag{4}$$

with n_g being the total number of constraints to be satisfied, and $g_{j,t}$ being the jth 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 (Watkins and Dayan (1992)). The agent performs actions to maximize some expected reward given an objective function $J(\cdot)$, which can be defined as

$$J = \sum_{t=0}^{t_f} \gamma^t R_t\left(\mathbf{x}_t, \mathbf{u}_t\right), \tag{5}$$

where $\gamma \in [0, 1]$ is the discount factor and R_t represents the reward at time t given values $\mathbf{x}_t$ and $\mathbf{u}_t$. 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 π an action-value function can be defined as

$$Q^\pi(\mathbf{x}_t, \mathbf{u}_t) = R_{t+1} + \gamma \sum_{\mathbf{x}_{t+1}} p_{\mathbf{x}_t \mathbf{x}_{t+1}}[\pi(\mathbf{x}_t)] V^\pi(\mathbf{x}_{t+1}), \tag{6}$$

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

$$Q^*\left(\mathbf{x}_t, \mathbf{u}_t\right) = \mathbb{E}_{\mathbf{x}_{t+1}\sim p}\left[R_{t+1} + \gamma \max_{\mathbf{u}_{t+1}} Q^*\left(\mathbf{x}_{t+1}, \mathbf{u}_{t+1}\right) \mid \mathbf{x}_t, \mathbf{u}_t\right] \tag{7}$$

$Q\left(\mathbf{x}_t, \mathbf{u}_t\right)$ can be represented by function approximators such as neural networks (Mnih et al. (2013)), Gaussian process (Chowdhary et al. (2014)) and tree-based regressors (Pyeatt et al. (2001)). In this work, the Q-function is approximated with a deep Q-network (DQN) Q_θ parameterized by weights θ. Here, the inputs specifically include the state $\mathbf{x}_t$, the corresponding time step t and control $\mathbf{u}_t$. The DQN is trained with the use of a replay buffer that addresses the issue of correlated sequential samples (Lin (1993)). Huber loss is used as the error function. Initial exploration is encouraged using an ϵ-greedy policy starting with high ϵ values, which is decayed over the course of training to ensure eventual exploitation and convergence to the optimal policy.

3. METHODOLOGY

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", where constraints are formulated as

$$g_{j,t} = \max(g_{j,t'}), t' \geq t \tag{8}$$

with $g_{j,t}$ being the jth constraint to be satisfied at time t, and 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 (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 (MC) simulations of the system. Analogous to a 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 $G_{j,\theta}$ with state and action as input features. However, the subtle difference between the two is that the state representation of the input for $G_{j,\theta}$ involves time-to-termination $t_f - t$ instead of time t.

Algorithm 1: Oracle-assisted constrained Q-learning

```
1. Initialize replay buffer $\mathcal{D}$ of size $s_\mathcal{D}$ and constraint
   buffers $\mathcal{G}_j$ of size $s_\mathcal{G}$, $j = 1, \ldots, n_g$
2. Initialize Q-network $Q_\theta$ and constraint networks
   $G_{j,\theta}$ with random weights, $j = 1, \ldots, n_g$
3. Initialize $\epsilon$ and backoffs $b_{j,t}$
for training iteration = 1, ..., $M$ do
    for episode = 1, ..., $N$ do
        Initialize state $\mathbf{x}_0 \sim p_{\mathbf{x}_0}(\mathbf{x}_0)$ and episode $\mathcal{E}$
        for $t$ = 0, ..., $t_f$ do
            1. With probability $\epsilon$ select random control
               $\mathbf{u}_t$
               otherwise select
               $\mathbf{u}_t = \max_\mathbf{u} Q_\theta(\mathbf{x}_t, \mathbf{u}_t) \mid G_{j,\theta}(\mathbf{x}_t, \mathbf{u}_t) +$
               $b_{j,t} \leq 0, j = 1, \ldots, n_g$ (Sub-problem *)
            2. Execute control $\mathbf{u}_t$ and observe reward
               $R_t$ and new state $\mathbf{x}_{t+1}$
            3. Store transition $(\mathbf{x}_t, \mathbf{u}_t, R_t, \mathbf{x}_{t+1})$ in $\mathcal{E}$
        end
        1. Extract Q-values from $\mathcal{E}$ and store
           datapoint $(\mathbf{x}_t, \mathbf{u}_t, Q_t)$ in $\mathcal{D}$
        2. Extract oracle-constraint values from $\mathcal{E}$
           using: $g_{j,t} = \max(g_{j,t'}), t' \geq t, j = 1, \ldots, n_g$
        3. Store datapoint $(\mathbf{x}_t, \mathbf{u}_t, g_{j,t})$ in
           $\mathcal{G}_j, j = 1, \ldots, n_g$
    end
    1. Sample random minibatch of datapoints of size
       $G$ $(\mathbf{x}_t, \mathbf{u}_t, Q_t)$ from $\mathcal{D}$
    2. Sample random minibatch of datapoints of size
       $H_j$ $(\mathbf{x}_t, \mathbf{u}_t, g_{j,t})$ from $\mathcal{G}_j$
    3. Perform gradient descent on $Q_\theta$ and $G_{j,\theta}$ using
       Adam optimizer ** with step size of $10^{-3}$
    4. Decay $\epsilon$ using $\epsilon = D_1\epsilon$
    5. Decay backoffs using $b_{j,t} = D_2 b_{j,t}$
end
Output: Optimal Q-network $Q^*_\theta$ and constraint
   networks $G_{j,\theta}, j = 1, \ldots, n_g$
```

* Sub-problem: An evolutionary algorithm is used to optimize the constrained Q-function using fitness function $f(\mathbf{u}) = Q_\theta(\mathbf{u}) + \sum_j C_j \min\left(0, -(g_{j,t}(\mathbf{u}) + b_{j,t})\right)$ where $g_{j,t}$ is the jth 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.

** Any other full optimization step can be used here

3.2 Constraint tightening

To satisfy the constraints with high probability, it is required that the constraints are tightened with backoffs (Bradford et al. (2020); Rafiei and Ricardez-Sandoval (2018)) $b_{j,t}$ as:

$$\overline{\mathbb{X}}_t = \left\{\mathbf{x}_t \in \mathbb{R}^{n_x} \mid g_{j,t}\left(\mathbf{x}_t\right) + b_{j,t} \leq 0, j = 1, \ldots, n_g\right\} \tag{9}$$

where $b_{j,t}$ are the backoffs which tighten the former feasible set $\mathbb{X}_t$ stated in (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 $b_{j,t}$

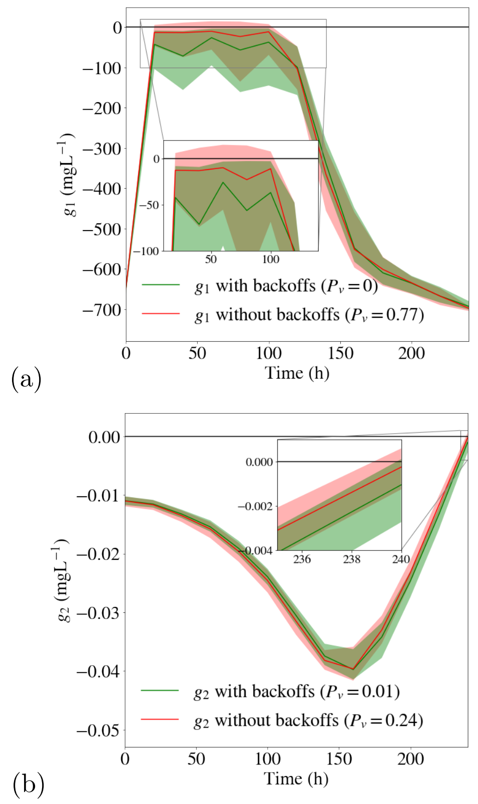

Fig. 2. Constraints $g_{1,t}$ (a) and $g_{2,t}$ (b) when backoffs are applied (green), and when they are absent (red) with probabilities of violation P_v within the parentheses. Inset: Zoomed-in region where violation of constraints occur. Shaded areas represent the 99th to 1st percentiles.

are the minimum value needed to guarantee satisfaction of constraints, such that:

$$P_{1-\omega}(g_{j,t}) + b_{j,t} = 0 \tag{10}$$

where $P_{1-\omega}(g_{j,t})$ is the $100(1-\omega)$th percentile of $g_{j,t}$. Here, ω is a tunable parameter depending on the case study, such that constraint satisfaction occurs with high probability $1-\omega$ as shown in (3). $P_{1-\omega}(g_{j,t})$ can be computed using sample approximation by S MC simulations. Here, we solve a root-finding problem in (10) using the Broyden's method to find the desired backoffs $b_{j,t}$ (Kelley (1995)). This way the constraints are satisfied with a desired probability and the policy is also trained to maximize the reward.

4. CASE STUDY

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 Bradford et al. (2020) that describes the evolution of concentration (c) of biomass (x), nitrate (N) and product (q) under parametric uncertainty. The model is based on

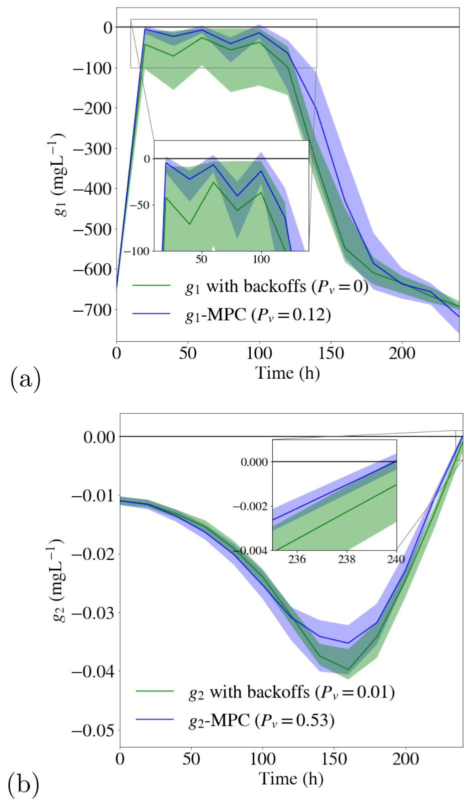

Fig. 3. Constraints $g_{1,t}$ (a) and $g_{2,t}$ (b) when backoffs are applied (green), and for MPC (blue) with probabilities of violation P_v within the parentheses. Inset: Zoomed-in region where violation of constraints occur. Shaded areas represent the 99th to 1st percentiles.

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 ($u_2 = F_N$).

This case study and parameter values are adopted from Bradford et al. (2020). Uncertainty in the system is two-fold: First, the initial concentration adopts a Gaussian distribution, where $[c_{x,0}, c_{N,0}] \sim \mathcal{N}([1.0, 150.0], \mathrm{diag}(10^{-3}, 22.5))$ and $c_q(0) = 0$. Second, parametric uncertainty is assumed to be: $\frac{k_s}{(\mu mol/m^2/s)} \sim \mathcal{N}(178.9, \sigma^2_{k_s})$, $\frac{k_i}{(mg/L)} \sim \mathcal{N}(447.1, \sigma^2_{k_i})$, $\frac{k_N}{(\mu mol/m^2/s)} \sim \mathcal{N}(393.1, \sigma^2_{k_N})$ where the variance $\sigma^2_i = 10\%$ 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 extracted using 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:

$$R_{t_f} = c_{q,t_f} \tag{11}$$

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:

$$g_{1,t} = c_N - 800 \leq 0 \quad \forall t \in \{0, \dots, t_f\} \tag{12}$$

$$g_{2,t} = c_q - 0.011c_x \leq 0 \quad \forall t \in \{0, \dots, t_f\} \tag{13}$$

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_θ consists 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: $\epsilon = 0.99$, $b_{1,t} = -500$, $b_{2,t} = -0.05$, $s_{\mathcal{D}} = 3000$, $s_{\mathcal{G}} = 30000$, $M = 2000$, $N = 100$, $G = 100$, $H_1 = 500$, $H_2 = 1000$, $D_1 = 0.99$ and $D_2 = 0.995$.

After completion of training using Algorithm 1, the backoffs are adjusted to satisfy (10), with backoffs at all timesteps 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 Petsagkourakis et al. (2020a) and Bradford et al. (2020). The constraint satisfaction is shown in Fig. 2, 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. 2 (a), even though it may seem at face value that $g_{1,t}$ 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 ($P_v = 0.77$). The violation probabilities P_v in Fig. 2 and 3 correspond to the fraction of 400 MC trajectories that violate a certain constraint. Gratifying, when backoffs are applied (green) in Fig. 2 (a), all constraints are satisfied ($P_v = 0$).

In the same vein, in Fig. 2 (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 the 99th percentile of $g_{j,t}$ values as shown in (10) where ω is set to 0.01. 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 MPC, 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. 3 (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). Clearly, compared to MPC, oracle Q-learning offers a more effective means of not only satisfying constraints in expectation (green lines in Fig. 2), but more importantly with high probability (all green shaded areas below zero), which is crucial in safety critical engineering applications.

Table 1. Comparison of probabilities of constraint violation P_v and objective values of different algorithms

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

5. CONCLUSIONS

In this paper we propose a new RL methodology for finding a controller policy that can satisfy constraints with high probability in stochastic and complex bioprocess 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 is superior compared 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

Abbeel, P. and Ng, A.Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. *arXiv preprint arXiv:1705.10528.*

Altman, E. (1999). *Constrained Markov decision processes*, volume 7. CRC Press.

Bradford, E., Imsland, L., Zhang, D., and del Rio Chanona, E.A. (2020). Stochastic data-driven model predictive control using gaussian processes. *Computers & Chemical Engineering*, 139, 106844.

Chowdhary, G., Liu, M., Grande, R., Walsh, T., How, J., and Carin, L. (2014). Off-policy reinforcement learning with gaussian processes. *IEEE/CAA Journal of Automatica Sinica*, 1(3), 227–238.

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

Ge, Y., Zhu, F., Ling, X., and Liu, Q. (2019). Safe q-learning method based on constrained markov decision processes. *IEEE Access*, 7, 165007–165017.

Kelley, C.T. (1995). *Iterative methods for linear and nonlinear equations*. SIAM.

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

Lin, L.J. (1993). Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Liu, Y., Ding, J., and Liu, X. (2019). IPO: Interior-point Policy Optimization under Constraints. URL `http://arxiv.org/abs/1910.09615`.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Ng, A.Y., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

Petsagkourakis, P., Sandoval, I.O., Bradford, E., Galvanin, F., Zhang, D., and del Rio-Chanona, E.A. (2020a). Chance constrained policy optimization for process control and optimization. *arXiv preprint arXiv:2008.00030*.

Petsagkourakis, P., Sandoval, I.O., Bradford, E., Zhang, D., and del Rio-Chanona, E.A. (2020b). Reinforcement learning for batch bioprocess optimization. *Computers & Chemical Engineering*, 133, 106649.

Pyeatt, L.D., Howe, A.E., et al. (2001). Decision tree function approximation in reinforcement learning. In *Proceedings of the third international symposium on adaptive systems: evolutionary computation and probabilistic graphical models*, volume 2, 70–77. Cuba.

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

Shin, J., Badgwell, T.A., Liu, K.H., and Lee, J.H. (2019). Reinforcement learning–overview of recent progress and implications for process control. *Computers & Chemical Engineering*, 127, 282–294.

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

Slowik, A. and Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, 1–17.

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

Tessler, C., Mankowitz, D.J., and Mannor, S. (2018). Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*.

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

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

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