Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 53-2 (2020) 11264–11270

Constrained Reinforcement Learning for Dynamic Optimization under Uncertainty

P. Petsagkourakis * **I.O. Sandoval** ** **E. Bradford** ***
D. Zhang ****,† **E.A. del Rio-Chanona** †

* *Centre for Process Systems Engineering (CPSE), University College London, Torrington Place, London, United Kingdom*
** *Instituto de Ciencias Nucleares, Universidad Nacional Autónoma de México, Ciudad de México, Mexico*
*** *Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway*
**** *Centre for Process Integration, Department of Chemical Engineering and Analytical Science,The University of Manchester, UK*
† *Centre for Process Systems Engineering (CPSE), Imperial College London, UK*

Abstract: Dynamic real-time optimization (DRTO) is a challenging task due to the fact that optimal operating conditions must be computed in real time. The main bottleneck in the industrial application of DRTO is the presence of uncertainty. Many stochastic systems present the following obstacles: 1) plant-model mismatch, 2) process disturbances, 3) risks in violation of process constraints. To accommodate these difficulties, we present a constrained reinforcement learning (RL) based approach. RL naturally handles the process uncertainty by computing an optimal feedback policy. However, no state constraints can be introduced intuitively. To address this problem, we present a chance-constrained RL methodology. We use chance constraints to guarantee the probabilistic satisfaction of process constraints, which is accomplished by introducing backoffs, such that the optimal policy and backoffs are computed simultaneously. Backoffs are adjusted using the empirical cumulative distribution function to guarantee the satisfaction of a joint chance constraint. The advantage and performance of this strategy are illustrated through a stochastic dynamic bioprocess optimization problem, to produce sustainable high-value bioproducts.

Keywords: Reinforcement learning, Uncertain dynamic systems, Stochastic control, Chemical process control, Adaptive control, Policy gradient

1. INTRODUCTION

The optimization of chemical processes presents a distinctive challenge to the process systems engineering community, given that they suffer from three conditions: 1) there is no precise model known for the process under consideration (plant-model mismatch), leading to inaccurate predictions and convergence to suboptimal solutions, 2) the process is affected by disturbances (i.e. it is stochastic), and 3) process systems can be sensitive, therefore erratic constraint specifications can be inconvenient or even dangerous.

An efficient dynamic process optimization approach needs to be able to handle both the inherent stochasticity of the system (e.g. process disturbances) and plant-model mismatches, while satisfying safety and physical constraints. To accomplish this, we exploit a method from *reinforcement learning* (RL) called *policy gradient* with the inclusion of chance constraints from sample approximations. RL has been shown to be a powerful control approach, and is one of the few control techniques able to handle nonlinear stochastic optimal control problems (Bertsekas, 2019). The inclusion of chance constraints is not new in optimal control, however this work focuses on the development of policies that can handle arbitrary stochastic systems and the constructed policy will require less than a second to be computed. It should further be noticed that recursive feasibility is out of the scope of this work. Here probabilistic guarantee is provided given the uncertain system.

Given the rapid development of machine learning technology, building a data-driven model to simulate, optimize, and control complex processes has become possible. In fact, a number of previous studies have adopted supervised learning methods (e.g. artificial neural network, Gaussian processes) to predict process behaviours and conduct open-loop process optimal control (Bradford et al., 2018; del Rio-Chanona et al., 2017). However, few studies have been conducted to investigate the applicability and efficiency of reinforcement learning in process engineering, and none of them include the efficient handling of constraints. Therefore, in this work, we propose a policy gradient reinforce-

⋆ EAC was supported by the Engineering and Physical Sciences Research Council (EPSRC) (EP/P016650/1).

ment learning algorithm with efficient sample approximation of chance constraints.

Reinforcement learning (in an approximate dynamic programming (ADP) philosophy), has received significant attention for chemical process control. Most of the approaches rely on the (approximate) solution of the Hamilton–Jacobi–Bellman equation (HJBE), and have been shown to be reliable and robust for several problem instances. For example, in Lee and Lee (2005) a model-based strategy and a model-free strategy for control of nonlinear processes were proposed, in Peroni et al. (2005) ADP strategies were used to address fed-batch reactor optimization. In Tang and Daoutidis (2018) with the inclusion of distributed optimization techniques, an input-constrained optimal control problem solution technique was presented, among other works (e.g. Shah and Gopal (2016)).

In this paper, we present another taking on RL, that of using policy gradient. Policy gradient methods directly estimate the control policy, without the need of a model, or the solution of the HJBE. We highlight their advantages next. In Policy gradient methods, the approximate policy can naturally approach a deterministic policy, whereas action-value methods (that use epsilon-greedy or Boltzmann functions) select a random control action with some heuristic rule. Policy gradient methods work directly with policies that emit probability distributions, which is much faster and does not require an online optimization step. Policy gradient methods are guaranteed to converge at least to a locally optimal policy even in high dimensional continuous state and action spaces, unlike action-value methods where convergence to local optima is not guaranteed. Hence, they also enable the selection of control actions with arbitrary probabilities. In some cases, the best approximate policy may be stochastic (Sutton and Barto, 2018). Petsagkourakis et al. (2020) seems to be the first research where these methods were applied in the context of process optimization and control.

Reinforcement learning and particularly policy gradient methods are considered to be advantageous as discussed above. However, the inclusion of constraints is not straightforward. Various approaches have been proposed in the literature, where usually penalties are applied for the constraints. Such approaches can be very problematic, easily loosing optimality or feasibility (Achiam et al., 2017) especially in the case of a fixed penalty. As it is stated in Wen (2018), existing methods cannot guarantee strict feasibility. The main approaches to incorporate constraints in this way make use of trust-region and fixed penalties (Achiam et al., 2017; Tessler et al., 2018), as well as cross entropy (Wen, 2018). Unfortunately, existing methods for constrained reinforcement learning that are based on policy gradient methods cannot guarantee strict feasibility of the output policies even when initialized with feasible initial policies. Also, in (Deisenroth et al., 2015; Kamthe and Deisenroth, 2018) a reinforcement learning approach was proposed where the constraints are taken into account but without probabilistic guarantee for the joint constraints.

To address the above challenges, we propose a method with probabilistic guarantees for the satisfaction of joint chance constraints. We assume a model with uncertainty to be available, with either parametric uncertainty or structural mismatch. The training of the policy is fully offline and can adapt to different environments as in Petsagkourakis et al. (2020). The proposed method utilizes backoffs for the tightening of the constraints. Several works have been proposed in the area of stochastic MPC including the recently proposed Koller et al. (2018); Paulson and Mesbah (2018); Bradford et al. (2019) to account for stochastic uncertainties in NMPC. These methods generally rely on generating closed-loop Monte Carlo (MC) samples offline from the plant to attain the required backoff values.

The structure of this paper is as follows: the problem statement is given in section 2, then in section 3 the details of proposed method for probabilistic satisfaction in reinforcement learning is given. An illustrative case study follows, where the framework is applied in a batch bioreactor. In the last session, the conclusions are discussed.

2. PROBLEM STATEMENT

In this work, the dynamic system is assumed to be given by a probability distribution, following a Markov process,

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

with $\mathbf{x} \in \mathbb{R}^{n_x}$, $\mathbf{u} \in \mathbb{R}^{n_u}$ and t being the states, control inputs and discrete time, respectively. This behaviour is observed in systems when stochastic disturbances are present and/or other uncertainties affect the physical system, like parametric uncertainties. The case of additive disturbance can be written as:

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

where $\mathbf{w} \in \mathbb{R}^{n_w}$ is a vector of Gaussian distributed additive disturbance. Additionally in the case of parametric uncertainty the model can be described as

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

with $\mathbf{p}$ being the uncertain parameters. Both (2) and (3) can be represented by (1). In this work we seek to maximize an objective function in expectation, using an optimal stochastic policy subject to probabilistic constraints despite the uncertainty of the system. This problem can be written as a Stochastic Optimal Control Problem (SOCP) in (4). It should be noticed that the notation from reinforcement learning is followed, where the objective is to maximize a total reward instead of minimizing a cost (Bertsekas, 2019).

$$\mathcal{P}(\pi^*(\cdot)) := \begin{cases} \max\limits_{\pi(\cdot)} \mathbb{E}[J(\mathbf{x}_t, \mathbf{u}_t)] \\ \text{s.t.} \\ \mathbf{x}_0 = \mathbf{x}(0) \\ \mathbf{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{u}_t \sim \pi(\mathbf{x}_t, D_t) \\ \mathbf{u} \in \mathbb{U} \\ \mathbb{P}(\bigcap\limits_{i=1}^{T} \{\mathbf{x}_i \in \mathbb{X}_i\}) \geq 1 - \alpha \\ \forall t \in \{0, ..., T-1\} \end{cases} \tag{4}$$

with J being the objective function, $\mathbb{U}$ the set of hard constraints for the control inputs and $\mathbb{X}_i$ constraints for states that must be satisfied with a probability $1-\alpha$. Specifically,

$$\mathbb{X}_t = \{\mathbf{x}_t \in \mathbb{R}^{n_x} | g_{j,t}(\mathbf{x}_t) \leq 0, j = 1, \ldots, n_g\}, \tag{5}$$

and the joint chance constraints are satisfied for the joint event over all $t \in \{1, \ldots, T\}$. Additionally, $\pi(\cdot)$ is the stochastic policy and D_t is a window of past inputs and states that are used by the policy. Unfortunately, this SOCP is intractable in general, and therefore, approximations must be made, some of these approximate methods come from the family of RL algorithms (Bertsekas, 2019; Schulman et al., 2017; Petsagkourakis et al., 2020).

In RL a policy $\pi_\theta(\cdot)$ parametrized by the parameters θ, is constructed. This policy, seeks to maximize the expectation of some objective function $J(\cdot)$. We can define this objective function in the finite horizon discrete-time case as

$$J = \sum_{t=0}^{T} \gamma^t R_t(\mathbf{u}_t, \mathbf{x}_t), \tag{6}$$

where $\gamma \in (0, 1]$ is the *discount factor* that allows to give more importance to the short-term actions and R_t a given reward at the time instance t for the values of $\mathbf{u}_t$, $\mathbf{x}_t$.

The problem that we aim to solve is challenging, since the policy must be constructed such that it satisfies the probabilistic joint state constraints with some probability, and not only in expectation. The next section discusses the methodology followed by the joint satisfaction of constraints utilizing explicit backoffs, where we can denote a tightened constraint set as:

$$\bar{\mathbb{X}}_t = \{\mathbf{x}_t \in \mathbb{R}^{n_x} | g_{j,t}(\mathbf{x}_t) + b_{j,t} \leq 0, j = 1, \ldots, n_g\}, \tag{7}$$

where variables $b_{j,t}$ represent the backoffs which tighten the original constraints $\mathbb{X}_t$ defined in (5).

3. CONSTRAINED POLICY OPTIMIZATION FOR PROBABILISTIC CONSTRAINTS

In this section, the general proposed framework for safe reinforcement learning is described. A policy $\pi(\cdot)$ is constructed to optimize in expectation an economic metric of the process (J). To accomplish this, a policy optimization is performed, by sampling the physical system (or the generative model) at each time instant. We denote $\boldsymbol{\tau}$, as the joint random variable of states, controls and rewards for a trajectory with a time horizon T:

$$\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{u}_0, R_0, ..., \mathbf{x}_{T-1}, \mathbf{u}_{T-1}, R_{T-1}, \mathbf{x}_T, R_T), \tag{8}$$

then the policy optimization can be defined as:

$$\pi_\theta^* = \arg\max_{\pi_\theta(\cdot)} \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\theta)} J(\boldsymbol{\tau}), \tag{9}$$

where $p(\boldsymbol{\tau}|\theta)$ represents the probability of the trajectory $\boldsymbol{\tau}$ given the parametrization θ of the policy. Several approximation are required in (9) to lead to satisfaction of constraints. Recently a methodology was proposed that satisfies the expected value of the constraints (Tessler et al., 2018; Wen, 2018), however this is not adequate for most engineering problems, as usually the system is subject to safety constraints, that need to be satisfied with with high probability. To account for constraint violations, in this work, probabilistic constraints are incorporated. The problem is then formulated as:

$$\begin{aligned} \pi_\theta^* = &\arg\max_{\pi_\theta(\cdot)} \mathbb{E}_{\boldsymbol{\tau}\sim p(\boldsymbol{\tau}|\theta)} J(\boldsymbol{\tau}) \\ &s.t. \\ &\mathbb{P}\Big(\bigcap_{i=1}^{T} \{\mathbf{x}_i \in \mathbb{X}_i\}\Big) \geq 1-\alpha \\ &\boldsymbol{\tau} = (\mathbf{x}_0, \mathbf{u}_0, R_0, ..., \mathbf{x}_{T-1}, \mathbf{u}_{T-1}, R_{T-1}, \mathbf{x}_T, R_T) \end{aligned} \tag{10}$$

We omit the hard constraints for the control inputs in (10) as they are inherently satisfied by the construction of the policy, e.g. the policy passes through a bounded and differential squashing function (Deisenroth et al., 2015).

In order to solve (10) three steps must be applied: 1) The policy is parameterized by a multilayer recurrent neural network that computes the mean and variance of the control actions given a state, resulting in a stochastic policy. 2) The probabilistic constraint in (10) is substituted by a surrogate set of constraints to guarantee closed-loop probabilistic constraint satisfaction. 3) The 'new' constraint is incorporated into the objective function as a penalty to be solved as an unconstrained optimization problem (Nocedal and Wright, 2006). The policy optimization is solved by a policy gradient framework (Sutton et al., 1999). In the next subsections these components are described.

3.1 Recurrent Neural Networks

Recurrent neural networks, (RNNs) (Rumelhart et al., 1986), are artificial neural networks that have recursive connections between hidden units. This allows them to obtain a 'memory' of previous data and model more accurately time-series. Let $\hat{\mathbf{x}}_t$ be the vector that contains previous realizations of the states $\mathbf{x}$ and the controls $\mathbf{u}$, i.e. $\hat{\mathbf{x}}_t = \left[\mathbf{x}_t^T, \ldots, \mathbf{x}_{t-N}^T, \mathbf{u}_{t-2}^T, \ldots, \mathbf{u}_{t-N-1}^T\right]^T$. Then the stochastic policy can be defined as:

$$\mathbf{u}_t \sim \pi_\theta(\mathbf{u}_t|\hat{\mathbf{x}}_t, \mathbf{u}_{t-1}) := \begin{cases} [\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t] = s_\theta(\hat{\mathbf{x}}_t, \mathbf{u}_{t-1}) \\ \mathbf{u}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) \end{cases}, \tag{11}$$

where s_θ is the multilayer RNN parametrized by θ. Deep structures (which means having more than one hidden layer) are employed to enhance the performance of the learning process (Mnih et al., 2013, 2015). The *actual* control inputs are drawn from the mean and variance that has been computed from (11). Having a stochastic policy could be advantageous when uncertainties are present, as a deterministic policy will always compute the same control inputs with the same states since it learns a deterministic mapping from states to control inputs. On the contrary, a stochastic policy draws a control action from a probability distribution which can account for inherent stochasticity of the uncertain environments.

3.2 Probabilistic constraints

In this section, we introduce the surrogate inequality constraints to substitute the probabilistic (chance) constraints. To achieve a probabilistic guarantee, backoff based tightening of individual constraints will be introduced such that satisfaction for the joint chance constraint is attained. The probabilistic constraints are intractable, but they can be approximated by the empirical cumulative distribution function (ecdf), using a sample approximation and S Monte Carlo (MC) simulations:

$$F = \mathbb{P}(\bigcap_{t=1}^{T}\{\mathbf{x}_t \in \mathbb{X}_t\}) \approx \hat{F}_S = \frac{1}{S}\sum_{s=1}^{S}\mathbb{1}\{\bigcap_{t=1}^{T}\{\mathbf{x}_t^s \in \mathbb{X}_t\}\ \}, \tag{12}$$

where $\hat{F}_S$ is the approximate joint constraint satisfaction probability for a trajectory, $\mathbb{1}\{\bigcap_{t=1}^{T}\{\mathbf{x}_t^s \in \mathbb{X}_t\ \}\} = \begin{cases}1, & \mathbf{x}_t^s \in \mathbb{X}_t\, \forall t \in \{1,\dots,T\} \\ 0, & otherwise\end{cases}$. The indicator function is a Bernoulli random variable, which means that $\hat{F}_S$ follows a binomial distribution, with $\hat{F}_S \sim \frac{1}{S}\text{Bin}(S,F)$, F being the cumulative distribution function (cdf). The confidence bound for the ecdf can then be computed from the Binomial cdf (Clopper and Pearson, 1934). In fact a simplified expression can be obtained using beta distributions instead leading to the following theorem.

Theorem 3.1. (Clopper and Pearson (1934)). Assume we are given a value of the ecdf $\hat{F}_S$ (see (12)) based on S i.i.d. samples, then the true value of the cdf, F, has a lower bound F_{lb} and lies inside the confidence interval $[F_{lb}(\epsilon; S; \hat{F}_S); F_{ub}(\epsilon; S; \hat{F}_S)]$ with a confidence level of $1-\epsilon$. Therefore, the probability of F given a lower bound $F_{lb}(\epsilon; S; \hat{F}_S)$ is as follows:

$$\begin{aligned}&\mathbb{P}\{F \geq 1-\alpha | F_{lb} \geq 1-\alpha\} \geq 1-\epsilon, \\ &F_{lb} = 1 - \text{betainv}(1-\epsilon, S+1-S\hat{F}_S, S\hat{F}_S),\end{aligned} \tag{13}$$

with betainv$(\cdot,\cdot,\cdot)$ being the inverse of the beta cdf with parameters $\{S+1-S\hat{F}_S\}$ and $\{S\hat{F}_S\}$, where $1-\alpha$ is the value for the probabilistic constraint satisfaction and $1-\epsilon$ is the confidence level.

The objective now is to tighten the individual constraints using backoffs such that the probabilistic lower bound of the ecdf of the joint constraints is equal to $1-\alpha$ (see (14)), given the confidence level of $1-\epsilon$.

$$F_{lb} = 1-\alpha. \tag{14}$$

In other words, if the lower bound of the ecdf (F_{lb}) is equal to $1-\alpha$, then the probability of satisfaction of the joint chance constraints in (10) is larger than $1-\alpha$ with a certainty no smaller than $1-\epsilon$. Hence, we aim to compute the tightened constraints by backoffs $b_{j,t}$ such that (14) is satisfied when the policy optimization has finished. To find $b_{j,t}$, we first compute an initial set of backoffs ($b_{j,t}^0$), as it has been proposed in Paulson and Mesbah (2018), where

$$\bar{g}_{j,t}(\mathbf{x}_t) + b_{j,t}^0 = 0\ \forall j,t \text{ gives } \mathbb{P}(\mathbf{x}_t \in \mathbb{X}_t) \geq 1-\delta, \tag{15}$$

with δ being a tuning parameter. Additionally, the mean value for $g_{j,t}$ is the sample average approximation (SAA): $\bar{g}_{j,t} = \frac{1}{S}\sum_{s=1}^{S} g_{j,t}^s$. The initial backoff values are computed to probabilistically satisfy each constraint:

$$\mathbb{P}(\mathbf{x}_t \in \mathbb{X}_t) \geq 1-\delta \tag{16a}$$

$$b_{j,t}^0 = \hat{F}_S^{-1}(1-\delta) - (\bar{g}_{j,t}(\mathbf{x}_t)),\ \forall j,t \tag{16b}$$

with $\hat{F}_S^{-1}(1-\delta)$ being the inverse ecdf of (16a). We wish the backoffs $\mathbf{b}$ (with elements $b_{j,t}\ \forall\ j,t\ \in \{1,\dots,n_g\} \times \{1,\dots,T\}$) to be large enough to ensure the probabilistic satisfaction of constraints. However, if the backoffs are too large, unnecessary conservatism will be present, since relaxing any active state constraint can only result in improved performance. Therefore, we wish for all active constraints to be as close to zero as possible. This results in a root-finding problem using Theorem 3.1 with $F_{lb} - (1-\alpha) = 0$. Given that we do not have the derivatives with respect to the backoffs, we solve this problem by a bisection algorithm. We implement this by iterating over a design parameter γ, such that $F_{lb} - (1-\alpha) = 0$, and $b_{j,t} = \gamma\ b_{j,t}^0$ is used as an update rule. Notice that now F_{lb} is a function of the design parameter γ, since the lower bound depends on the tightening that is applied.

With the above procedure we are able to compute a surrogate for the probabilistic constraints.

3.3 Penalty Function and policy gradient method

In this work we reformulate the constraint using a quadratic penalty function and then solve the problem using a policy gradient method. A smooth quadratic penalty is added to the objective function of (10)

$$\max_{\pi_\theta(\cdot)} \mathbb{E}_{\boldsymbol{\tau}} \left(J(\boldsymbol{\tau}) + \mu \sum_{j=1}^{n_g}\sum_{t=1}^{T} \max(g_{j,t}(\mathbf{x}_t)), 0)^2 \right), \tag{17}$$

where $g_{j,t}$ is given in (5) and $\boldsymbol{\tau}$ in (8). It should be noted that the same framework can be implemented in most policy optimization methods (Tessler et al., 2018; Achiam et al., 2017). As it is observed in Achiam et al. (2017), when penalty methods are applied in policy optimization, depending on the value of parameter μ the behaviour of the policy may change. If a large value of μ is used, then the policy tends to be over-conservative resulting in feasible areas that are not optimal; on the other hand, when the value for μ is too small, the policy tends to ignore the constraints as in the unconstrained optimization case. Therefore, the value of μ must be carefully chosen, with the constraints substituted by the tightened constraints using backoffs. In this way, the policy optimization will compute the 'near'-optimal solution, at the same time guaranteeing the probabilistic satisfaction of the constraints for a given value of μ.

3.4 Proposed Algorithm

In this work we propose the use of policy gradient, and particularly of the Reinforce algorithm. Reinforce (Sutton et al., 1999) approximates the gradient of the policy to maximize the expected reward with respect to the parameters θ without the need of a dynamic model of the process. It should be mentioned that different algorithms may be applied (Schulman et al., 2017; Kakade, 2002). However the focus here is on the probabilistic guarantee of the constraints. We now define our reward function as: $\hat{J}(\boldsymbol{\tau}, \mathbf{b}) = J(\boldsymbol{\tau}) + \mu\sum_{j=1}^{n_g}\sum_{t=1}^{T}\max(g_{j,t}(\mathbf{x}_t)) + b_{j,t}, 0)^2$, which is now an explicit function of our probabilistic constraints. We now use the Policy Gradient theorem (Sutton et al., 1999) to obtain an explicit gradient of the reward with respect to the parameters that parameterize our policy:

$$\nabla_\theta \mathbb{E}_{\boldsymbol{\tau}}(J) \approx \frac{1}{K}\sum_{k=1}^{K}\left[\hat{J}(\boldsymbol{\tau}^k, \mathbf{b})\nabla_\theta \sum_{t=0}^{T-1}\log\left(\pi(\mathbf{u}_t^k | \hat{\mathbf{x}}_t^k, \theta)\right)\right] \tag{18}$$

where we denote the sample k as a superscript that denotes the k^{th} sampled trajectory. The variance of this estimation can be reduced with the aid of an action-independent baseline $\bar{\beta}_s$, which does not introduce a bias (Sutton and Barto, 2018). A simple but effective baseline is the expectation of reward under the current policy, approximated by the mean over the sampled paths:

$$\bar{\beta}_s = \bar{J}(\theta) \approx \frac{1}{S}\sum_{k=1}^{K} \hat{J}(\boldsymbol{\tau}^k, \mathbf{b}), \tag{19}$$

which leads to:

$$\nabla_\theta \hat{J}(\theta) \approx \frac{1}{K}\sum_{k=1}^{K}\left[(\hat{J}(\boldsymbol{\tau}^k,\mathbf{b}) - \bar{\beta}_s)\nabla_\theta \sum_{t=0}^{T-1}\log\left(\pi(\mathbf{u}_t^k|\hat{\mathbf{x}}_t^k,\theta)\right)\right] \tag{20}$$

This selection increases the log likelihood of an action by comparing it to the expected reward of the current policy. Equation (20) is the gradient that is used to update the policy in a gradient ascent fashion in policy gradient methods. For fixed values of backoffs, the policy gradient method is outlined in Algorithm 1.

Algorithm 1 Policy gradient for fixed backoff

```
Input: Initialize policy parameter $\theta = \theta_0$, with $\theta_0 \in \Theta_0$, learning rate $\alpha_0$ and its update rule, the number of episodes $K_0$ and the number of epochs $N_0$, $\mu$ parameter, tolerance $tol$, set backoffs $\mathbf{b}$
Output: policy $\pi^*(\cdot|\cdot,\theta)$ and $\Theta$.
for m = 1,..., N do
    (1) Collect $\mathbf{u}_t^k, \mathbf{x}_t^k$ for $T$ time steps for $K$ trajectories along with $\hat{J}(\mathbf{x}_T^k)$, also for $K$ trajectories.
    (2) Update the policy: $\theta_{m+1} = \theta_m + \frac{\alpha_m}{K}\sum_{k=1}^{K}\left[(\hat{J}(\boldsymbol{\tau}^k,\mathbf{b}) - \bar{\beta}_s)\nabla_\theta\sum_{t=0}^{T-1}\log\left(\pi(\mathbf{u}_t^k|\hat{\mathbf{x}}_t^k,\theta)\right)\right]$
    (3) $history(m+1) := \mathbb{E}(\hat{J})$
    (4) if $|history(m+1) - history(m)| \leq tol$ then $exit$
```

Algorithm 1 is the base for Algorithm 2, which is discribed next. In **step (1)** the policy is trained with $\mathbf{b} = 0$, with the initial estimate of the backoffs $b_{j,t}^0$ computed in **step (2)** with i.i.d. S samples. Then, the backoff values will repeatedly change (**3**) until the desired performance is achieved, which corresponds to the satisfaction of (14). The value of γ_m is the half of a_m and c_m in **step 3i**. After construction of the new policy, (14) is evaluated in **step 3iv** using S i.i.d. samples and the relevant changes of a_m and c_m subject to bisection method are made in **step 3v**. Lastly the algorithm terminates when a desirable tolerance tol_0 has been achieved (**step 3vi**). It should be noted that every time a new backoff is computed then the policy is re-optimized. This may look inefficient at first glance, however the convergence is achieved fast, as everytime the previous policy is used as an initial guess for the next iteration. In the end of Algorithm 2, a probabilistically constrained policy will have been constructed.

3.5 Policy initialization

Reinforcement learning (including policy gradient methods) is computationally expensive; this is mainly because most of the the computational cost is shifted offline, but also because initially the agent (or controller in our case) explores the control action space randomly. In the case of process optimization and control, it is possible to use a preliminary controller, along with supervised learning to hot-start the policy, and significantly speed-up convergence.

The main idea here is to have data from some policy or state-feedback control (e.g. PID controller, (economic) model predictive controller) to compute control actions given observed states. The initial parameterization for the policy (before **step 1**) is trained in a supervised learning fashion where the states are the inputs and the control actions are the outputs. Subsequently, this parameterized policy is used to initialize the policy in **step 1** and then trained by the RL algorithm.

Algorithm 2 Backoff-Based Policy Optimization

```
Input: Initialize policy parameter $\theta = \theta_0$, with $\theta_0 \in \Theta_0$, learning rate, its update rule $\alpha$, $m := 0$, the number of episodes $K_0, K_1$ and the number of epochs $N_0, N_1$, $\mu$ parameter, set backoffs $\mathbf{b} = \mathbf{0}$, $\delta < 1$, $\alpha < 1$, $tol_0 = 1 \times 10^{-4}$, $a_0 = 0$, $c_0$, maximum number of backoff iterations $M$ and $S$ number of samples to compute $\hat{F}_S$
Output: policy $\pi^*(\cdot|\cdot,\theta)$ and $\Theta$.
(1) Perform policy optimization with $\mathbf{b} = \mathbf{0}$ using Algorithm 1 with $K_0$ episodes and $N_0$ epochs.
    Obtain policy $\pi_0(\cdot|\cdot) = \pi^*(\cdot|\cdot,\theta)$
(2) Estimate initial backoff using $S$ i.i.d. samples:
    $b_{j,t}^0 = \hat{F}_S^{-1}(1-\delta) - \bar{g}_{j,t}(\mathbf{x}_t)\ \forall j, t$
(3) for m = 0,..., $M$ do.
    i   Set $\gamma_m = \frac{a_m + c_m}{2}$
    ii  Set $\mathbf{b} = \gamma_m \mathbf{b}^0$
    iii Perform policy optimization with backoffs $\mathbf{b}$ using Algorithm 1 with $N_1$, $K_1$ and $tol_0$.
        Obtain policy $\pi_{\mathrm{m}}(\cdot|\cdot) = \pi^*(\cdot|\cdot,\theta)$.
    iv  Compute $e_m = F_{lb} - (1-\alpha)$ using $S$ i.i.d. samples.
    v   if $e_m < 0$ then: $a_{m+1} := \gamma_m$
        else $c_{m+1} := \gamma_m$
    vi  if $|e_m| \leq tol_0$ then $exit$
```

4. CASE STUDY

The case study in this paper focuses on the photo-production of phycocyanin synthesized by cyanobacterium *Arthrospira platensis*. Phycocyanin is a high-value bio-product and its biological function is to enhance the photosynthetic efficiency of cyanobacteria and red algae. It has applications as a natural colorant to replace other toxic synthetic pigments in both food and cosmetic production. Additionally, the pharmaceutical industry considers it as beneficial because of its unique antioxidant, neuroprotective, and anti-inflammatory properties.

The dynamic system consists of the following system of ODEs describing the evolution of the concentration of biomass (X), nitrate (N), and product (q) under parametric uncertainty. The dynamic model is based on Monod kinetics, which describes microorganism growth in nutrient sufficient cultures, where intracellular nutrient concentration is kept constant because of the rapid replenishment. We assume a fixed volume fed-batch. The manipulated variables as in the previous examples are the light intensity ($u_1 = I$) and inflow rate ($u_2 = F_N$). The mass balance equations are

$$\frac{dc_x}{dt} = u_m \frac{I}{I + k_s + I^2/k_i} c_x \frac{c_N}{c_N + K_N} - u_d c_X \quad (21)$$

$$\frac{dc_N}{dt} = -Y_{N/X} u_m \frac{I}{I + k_s + I^2/k_i} c_x \frac{c_N}{c_N + K_N} + F_N \quad (22)$$

$$\frac{dc_q}{dt} = k_m \frac{I}{I + k_{sq} + I^2/k_{iq}} c_x \frac{c_N}{c_N + K_N} - k_d \frac{c_q}{C_N + K_{N_q}} \quad (23)$$

The parameter values are adopted from (Bradford et al., 2019). Uncertainty is assumed for the initial concentration, where

$$[c_x(0)\ c_N(0)] \sim \mathcal{N}([1.\ 150.], diag(1 \times 10^{-3}, 22.5)). \quad (24)$$

$$c_q(0) = 0. \quad (25)$$

Additionally, the 10% parametric uncertainty of the system: $k_s \sim \mathcal{N}(178.9, 17.89)$, $k_i \sim \mathcal{N}(447.1, 44.71)$ and $k_N \sim \mathcal{N}(393.1, 39.31)$. The objective function(reward) in this work is to maximize the product's concentration (c_q) at the end of the batch. The objective is additionally penalized by the change of the control actions $\mathbf{u}(t) = [I, F_N]^T$. As a result the reward is given as:

$$R_t = -\Delta\mathbf{u}_t^T diag(3.125 \times 10^{-8}, 3.125 \times 10^{-6})\Delta\mathbf{u}_t, \\ t \in \{0, T-1\}, R_T = c_q(T), \quad (26)$$

where $\Delta\mathbf{u}_t = \mathbf{u}_t - \mathbf{u}_{t-1}$. The relevant constraints in this work for each time step are $g_{1,t} = c_N - 800 \leq 0$ and $g_{2,t} = c_q - 0.011c_X \leq 0$. This constraints have been normalized as:

$$\tilde{g}_{1,t} = \frac{c_N}{800} - 1 \leq 0, \quad \tilde{g}_{2,t} = \frac{c_q}{0.011c_X} - 1 \leq 0 \quad (27)$$

and the constraints are meant to be satisfied with probability 99% ($\alpha = 0.01$) and confidence level is 99% ($\epsilon = 0.01$). The constraints are added as a penalty with $\mu = 10$. The control actions are constrained to be in the interval $0 \leq F_N \leq 40$ and $120 \leq I \leq 400$, these constraints are considered to be hard. The control policy RNN is designed to contain 4 hidden layers, each of which comprises 20 neurons embedded by a leaky rectified linear unit (ReLU) activation function. A unified policy network with diagonal variance is utilized such that the control actions share memory and the previous states are used from the RNN (together the current measured states). The computational cost for each control action online is insignificant since it only requires the evaluation of the corresponding RNN. First the algorithm computes the policy for the bakcoffs to be zero ($\mathbf{b} = 0$), then the backoffs are updated according to the Algorithm 2. The parameters for the trainings are: $N_0 = 500$, $N_1 = S = 500$, $M = 100$, $K_0 = 150$, $K_1 = 50$, $tol = tol_0 = 10^{-4}$ and the two previous states and controls are used from the policy. After the completion of the training the backoffs have been computed and in Fig. 1 the convergence of the lower bound F_{lb} is shown. In a rather small number of iterations the backoff values managed to force the F_{lb} to 0.99. The figure shows F_{lb} against all the iterations performed in **step 3** including the training of policy in **step 3iii**. Now, the actual value of the constraints can be depicted in Fig 2, where the shaded areas are the 98% and 2% percentiles. The results are also compared with the case of the absence of backoffs ($\mathbf{b} = 0$). As it was expected the use of backoffs managed to remain feasible and in the case of the g_2 steer the problem to its boundary. The normalized backoff values of the $\tilde{g}_{1,t}$ and $\tilde{g}_{2,t}$ for each update are shown in Fig 3,

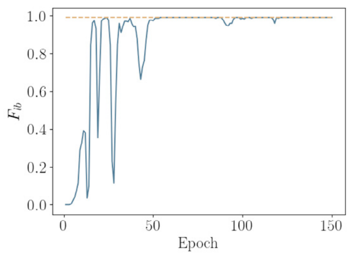

Fig. 1. Convergence of lower bound of F_{lb} to $1 - \alpha$.

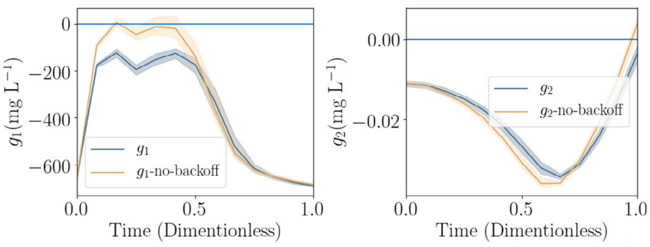

Fig. 2. Constraint satisfaction under the absence and presence of backoffs, where the shaded areas are the 98% and 2% percentiles.

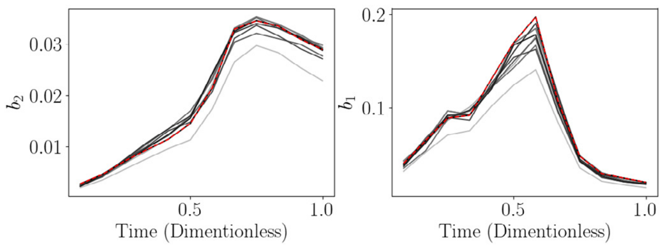

Fig. 3. The backoff values (dimensionless) are plotted over number of epochs, which are faded out towards earlier iterations.

where the red-dashed represents the converged final value. Based on the comparison, it is concluded that integrating backoffs into RL can significantly improve RL's optimal control performance when handling complex systems with high stochasticity. From Fig 2, it is seen that the current strategy can well satisfy all the practical constraints; while the RL without backoffs breaches g_1 in the middle of the operation and violates g_2 at the later stage of the process, hence resulting in an infeasible optimal policy. It should be noted that the final value for the product c_q (26) is 0.159 and 0.172 when backoffs are applied and when they are not. This difference makes sense since the case of $\mathbf{b} = 0$ does not have probabilistic guarantee for the satisfaction of the constraints and the objective can attain higher value in the infeasible area.

The algorithm is implemented in Pytorch version 0.4.1. Adam (Kingma and Ba, 2014) is employed to compute the network's parameter values using a step size of 10^{-2} with the rest of hyperparameters at their default values.

5. CONCLUSIONS

For fermentation and pharmaceutical processes, even a transitory violation of hard constraints may directly damage the product quality and result in an early termination of ongoing batch operation. As a result, choosing a robust

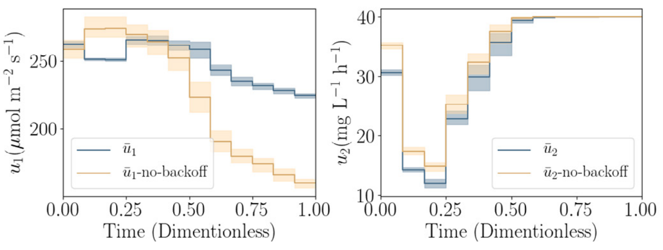

Fig. 4. Comparison of the time trajectories of the piecewise constant control actions, where the shaded areas are the 98% and 2% percentiles.

online optimization method is of critical importance when uncertainty is present in a process. The current results further reveal that it is possible to obtain a near optimal and feasible policy given a general uncertain system. In real systems with the absence of a true model, it is impossible to generate highly accurate datasets to train the policy network. As a result, a method that estimates offline backoffs should be thoroughly investigated as it can offer additional advantages to guarantee the probabilistic feasibility of hard constraints when computing an optimal policy. In terms of future work, experimental verification will be conducted to test the efficiency of this strategy and provide suggestions to further improve the current method.

REFERENCES

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained Policy Optimization. *arXiv preprint 1705.10528*.

Bertsekas, D.P. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific, 1st edition.

Bradford, E., Imsland, L., Zhang, D., and Chanona, E.A.d.R. (2019). Stochastic data-driven model predictive control using Gaussian processes. *arXiv preprint 1908.01786*.

Bradford, E., Schweidtmann, A.M., Zhang, D., Jing, K., and del Rio-Chanona, E.A. (2018). Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes. *Computers & Chemical Engineering*, 37.

Clopper, C.J. and Pearson, E.S. (1934). The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4), 404–413.

Deisenroth, M.P., Fox, D., and Rasmussen, C.E. (2015). Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on pattern analysis and machine intelligence*, 37(2), 408 – 423.

del Rio-Chanona, E.A., Fiorelli, F., Zhang, D., rashid Ahmed, N., Jing, K., and Shah, N. (2017). An efficient model construction strategy to simulate microalgal lutein photo-production dynamic process. *Biotechnology and Bioengineering*, 114(11), 2518–2527.

Kakade, S. (2002). A natural policy gradient. *Advances in Neural Information Processing Systems*.

Kamthe, S. and Deisenroth, M.P. (2018). Data-efficient reinforcement learning with probabilistic model predictive control. volume 84 of *Proceedings of Machine Learning Research*, 1701–1710.

Kingma, D.P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.

Koller, R.W., Ricardez-Sandoval, L.A., and Biegler, L.T. (2018). Stochastic back-off algorithm for simultaneous design, control, and scheduling of multiproduct systems under uncertainty. *AIChE Journal*, 64(7), 2379–2389.

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.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv preprint 1312.5602*, 1–9.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518, 529.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*.

Paulson, J.A. and Mesbah, A. (2018). Nonlinear Model Predictive Control with Explicit Backoffs for Stochastic Systems under Arbitrary Uncertainty. *IFAC-PapersOnLine*, 51(20), 523–534.

Peroni, C., Kaisare, N., and Lee, J. (2005). Optimal control of a fed-batch bioreactor using simulation-based approximate dynamic programming. *IEEE Transactions on Control Systems Technology*, 13(5), 786–790.

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

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shah, H. and Gopal, M. (2016). Model-Free Predictive Control of Nonlinear Processes Based on Reinforcement Learning. *IFAC-PapersOnLine*, 49(1), 89–94.

Sutton, R. and Barto, A. (2018). *Reinforcement Learning: An Introduction Second Edition*. MIT Press.

Sutton, R.S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, 1057–1063. MIT Press, Cambridge, MA, USA.

Tang, W. and Daoutidis, P. (2018). Distributed adaptive dynamic programming for data-driven optimal control. *Systems & Control Letters*, 120, 36–43.

Tessler, C., Mankowitz, D.J., and Mannor, S. (2018). Reward Constrained Policy Optimization. *arXiv preprint 1805.11074*, (2016), 1–15.

Wen, M. (2018). Constrained Cross-Entropy Method for Safe Reinforcement Learning. *Neural Information Processing Systems (NIPS)*, (Nips).