12th IFAC International Symposium on
Advanced Control of Chemical Processes
July 14-17, 2024. Toronto, Canada

Enhancing Reinforcement Learning Robustness via Integrated Multiple-Model Adaptive Control

Soroush Rastegarpour, Hamid Reza Feyzmahdavian, and Alf J. Isaksson

ABB Corporate Research, Västerås, Sweden
(e-mails: soroush.rastegarpour, hamid.feyzmahdavian, alf.isaksson@se.abb.com).

Abstract: Reinforcement learning (RL) has attracted considerable attention from both industry and academia for its success in solving complex problems. However, the performance of RL algorithms often decreases in environments characterized by uncertainties, unmodeled dynamics, and nonlinearities. This paper presents a novel robust RL algorithm designed to ensure closed-loop stability for industrial processes. The algorithm considers a wide range of potential scenarios across various operating conditions and different ranges of parameter uncertainties. Using the multiple-model adaptive control methodology, the algorithm evaluates all scenarios and ranks them based on their likelihood of accurately characterizing the actual process. The validity of the results is demonstrated using a benchmark continuous stirred tank reactor (CSTR) problem.

Keywords: Reinforcement learning control, Robust control, Process control applications

1. INTRODUCTION

Real-Time Optimization (RTO) and Advanced Process Control (APC) are standard tools used in complex process control industries, including chemical plants, refineries, and power generation facilities (Kadam et al., 2003; Adetola and Guay, 2010; De Souza et al., 2010). RTO relies on static mathematical models to optimize efficiency, productivity, and profitability in response to dynamic factors like market fluctuations, energy costs, and demand shifts. On the other hand, APC employs advanced methods, such as model predictive control, to improve the performance and stability of the process while aligning it with the set-points computed by RTO. Although both RTO and APC have solid theoretical foundations that provide closed-loop guarantees, their applicability for any given process is limited by challenges associated with adapting to various operating conditions, handling model uncertainties, and addressing issues related to operator acceptance and training (Campos et al., 2009).

Artificial Intelligence (AI), particularly Reinforcement Learning (RL), offers a promising alternative to address these limitations. RL algorithms can learn control policies through interaction with a process, allowing them to adapt to the system's behavior using training data. This adaptability enhances their performance when dealing with complex or poorly understood processes (Lewis et al., 2012; Recht, 2019; Bertsekas, 2019). However, when it comes to learning policies for real-world applications involving processes characterized by uncertainties and unmodeled dynamics, RL algorithms often face significant challenges related to sample complexity and safety (Garcıa and Fernández, 2015). In such cases, RL methods struggle with taking a long time to learn and the need to avoid many potentially dangerous tasks automatically. Uncertainties and unmodeled dynamics represent aspects of the process that remain partially or entirely unknown during the training phase.

To guarantee robustness and stability, specific RL algorithms prioritize worst-case scenarios (Tamar et al., 2014; Derman et al., 2020; Wang and Zou, 2021). Their main goal is to find control policies that excel in the most adverse operating conditions. However, this strong focus on extreme situations can lead to unnecessary caution, causing the system to miss opportunities for improved performance under normal or less severe conditions. This excessive caution can also result in system slowdowns and reduced adaptability to changes. Conversely, some RL algorithms attempt to balance between aggressiveness and robustness without being overly conservative for worst-case scenarios, see for example (Morimoto and Doya, 2005; Tessler et al., 2019; Panaganti et al., 2022). This balance is achieved through using interpolation techniques or a probabilistic interpretation of model uncertainties. While these methods perform effectively on average, they may fall short in scenarios requiring an exceptionally robust controller. Moreover, since these algorithms are frequently calibrated based on predefined uncertainty distributions, their performance often becomes sub-optimal in practice. To address these drawbacks, (Rajeswaran et al., 2016) developed an ensemble policy optimization algorithm for learning control policies that are robust to model mismatches. Although this method exhibits good generalization performance, its computational time can become impractical, especially for processes with a large number of parameters.

This paper presents a novel algorithm for training an RL agent, using a set of low-fidelity models. The main objective is to ensure process stability in the presence of an-

ticipated events, disturbances, and nonlinearities, thereby reducing the need for manual intervention and promoting increased autonomy. To achieve this goal, the proposed algorithm considers a wide range of potential scenarios that can arise across various operating conditions, including normal and worst-case situations. It also accounts for varying bounds on uncertainties and disturbances. In each scenario, a model serves as an approximation of the underlying process, allowing the algorithm to learn effective policies that can adapt to different operating modes and a range of parameter uncertainties. Throughout its operation, the algorithm assesses these approximated models and ranks them based on their likelihood of accurately characterizing the actual process, prioritizing the most suitable models for consideration.

The proposed approach draws inspiration from ensemble robust control methods, often referred to as Multiple-Model Adaptive Control (MMAC) (Murray-Smith and Johansen, 2020). Specifically, MMAC with RL is combined to leverage actual data and engineering knowledge of a given process. In contrast to traditional methods that either focus only on learning from worst-case scenarios or rely on simple averaging of scenarios, our algorithm selects the most suitable model or combination of models. It assigns weights based on the models' relevance to the current system behavior, ensuring that control solutions remain within process boundary limits.

2. PROBLEM FORMULATION

Consider nonlinear dynamic systems on the form:

$$x_{k+1} = f(x_k) + g(u_k). \tag{1}$$

Here, $x_k \in \mathcal{X}$ is the state vector measured at discrete time $k \in \mathbb{N}$, $u_k \in \mathcal{U}$ is the input vector (referred to as manipulated variables), and f and g represent the state transition and control input functions respectively. The sets $\mathcal{X}$ and $\mathcal{U}$ define feasible states and control inputs. Note that linear systems are a special case of (1) when $f(x) = Ax$ and $g(u) = Bu$.

We differentiate between the target domain, characterized by unknown system functions (f, g), and the source domain, which consists of M models

$$\{(f_1, g_1), (f_2, g_2), \ldots, (f_M, g_M)\}.$$

These models are derived from first principles. Ideally, we aim for the target domain to align with one of the models from the source domain and use the control policy $\pi_\theta : \mathcal{X} \to \mathcal{U}$ specific to that model for the process. However, in practical scenarios, unmodeled dynamics and uncertainties are likely to exist. To address this, we employ a weighted combination of the models in an adaptive manner.

RL centers around solving an optimal control problem to find the control policy $\pi_\theta : \mathcal{X} \to \mathcal{U}$. The objective is to maximize the value function V_∞, representing the discounted sum of the reward function over an infinite time horizon:

$$V_\infty(k) = \sum_{i=0}^{\infty} \gamma^i r(x_{k+i}, u_{k+i}) \tag{2}$$

In this paper, we select the actor-critic structure to find the optimal policy. This choice is made since it combines aspects of both policy-based (actor) and value-based (critic) methods. The actor is responsible for making decisions, determining the optimal actions based on the current state. It learns and refines the policy through exploration and by assessing the outcomes of its actions. The critic, on the other hand, plays the role of an evaluator. It provides feedback on the actions taken by the actor, assessing their effectiveness and guiding the actor towards more rewarding choices. Figure 1 visualizes the overall learning process in the actor-critic structure.

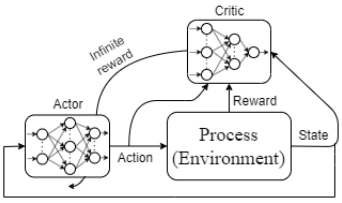

Fig. 1. Structure of the actor-critic RL algorithm.

The network configuration of the actor-critic algorithm is well studied in literature, see for example (Konda and Tsitsiklis, 1999; Peters and Schaal, 2008; Grondman et al., 2012). In the following section, we will explore a new method for robustifying an RL agent trained on a set of inaccurate models of the actual process. We will show how to parameterize the RL training process in order to train an agent on feasible observation-action sets. This is a necessary condition to guarantee that the final trained agent is not only optimal for the weighted combination of the ensemble models, but also has high performance across all models within the source domain in terms of feasibility and optimality.

3. ROBUST RL ALGORITHM

Up to this point, we assume that the actual process equations and parameters are not available. Therefore, a first-principle model of the process, hereafter named as base-model, is serving as the environment on which the RL agent will be trained. This enables us to address the challenges associated with real-world applications where such detailed knowledge may not be readily accessible.

Starting from the base-model and incorporating the domain knowledge captured from the process documents, one can construct and tune several models by introducing various formulations of uncertain equations and parameters. For example, these parameters might pertain to reaction kinetics in a chemical reactor, material properties like thermal conductivity or mechanical strength, or performance models for renewable energy systems, among others. These models are then assessed based on their importance and the probability of their occurrence within the actual process. Once the models are scored, the RL agent undergoes training by interpolating between the scenarios introduced by each model, with weights assigned according to their corresponding scoring rates.

There are two different approaches to interpolate these scenarios, each with its own set of pros and cons:

(1) **Aggregation on observations.** In this approach, the RL agent is trained on the interpolation of the collected observations from the ensemble of models (see Figure 2). This is computationally efficient as

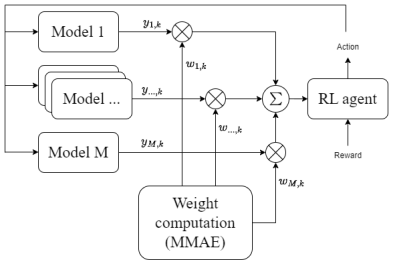

Fig. 2. Robust RL architecture with the MMAC algorithm: Aggregation on observations.

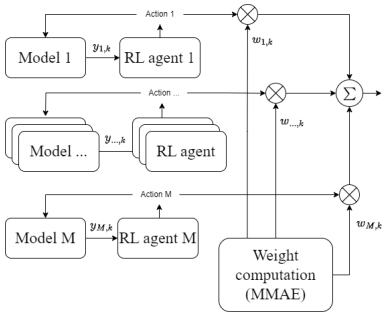

Fig. 3. Robust RL architecture with the MMAC algorithm: Aggregation on actions.

we only learn from one environment hosting M models. However, performance of the obtained agent is extremely dependent on the quality of the historical data. As long as the actual process falls within the domain of historical data, meaning that the actual process operates in the vicinity of the interpolated observation set, the RL agent performs well.

(2) **Aggregation on actions.** In this method, it is assumed that an optimal RL agent exists for each individual model, but this does not necessarily guarantee optimal performance for the actual process. The ultimate action will be determined by interpolating the proposed actions from each model's RL agent, enabling us to compute the optimal combination of these policies for deployment in the actual process (see Figure 3). While this method may be computationally more expensive, it can ensure performance and provide the most accessible sub-optimal solution under given circumstances (see Lemma 1).

Lemma 1. Suppose that there are M models described as

$$x_{i,k+1} = f_i(x_{i,k}) + g(u_{i,k}) \ , \ i \in \{1, .., M\}. \tag{3}$$

Assume that there is a separate controller for each model such that for $u_{i,k} \in \mathcal{U}$ and $x_{i,k} \in \mathcal{X}$, we have $x_{i,k+1} \in \mathcal{X}$. Assume also that g is a convex function and the sets $\mathcal{X}$ and $\mathcal{U}$ are convex. If $x_{i,k} = x_k$ for each i and there exists a vector w_k with $\sum_{i=1}^{M} w_{i,k} = 1$ such that

$$f(x_k) = \sum_{i=1}^{M} w_{i,k} f_i(x_k),$$

then the control input

$$u_k = \sum_{i=1}^{M} w_{i,k} u_{i,k}$$

guarantees the feasibility of the actual process (1).

Proof. The control vector u_k is a convex combination of $u_{1,k}, \ldots, u_{M,k}$. Since $\mathcal{U}$ is a convex set and each $u_{i,k} \in \mathcal{U}$, it follows that $u_k \in \mathcal{U}$. From (1), we have

$$\begin{aligned} x_{k+1} &= \sum_{i=1}^{M} w_{i,k} f_i(x_k) + g\left(\sum_{i=1}^{M} w_{i,k} u_{i,k}\right) \\ &\leq \sum_{i=1}^{M} w_{i,k} f_i(x_k) + w_{i,k} g(u_{i,k}) \\ &= \sum_{i=1}^{M} w_{i,k} \left(f_i(x_k) + g\left(u_{i,k}\right)\right) \\ &= \sum_{i=1}^{M} w_{i,k} x_{i,k+1}, \end{aligned}$$

where the inequality follows from the convexity of g. Each $x_{i,k+1} \in \mathcal{X}$ and $\mathcal{X}$ is a convex set. Thus, x_{k+1} as a convex combination of $x_{1,k+1}, \ldots x_{M,k+1}$ belongs to $\mathcal{X}$. This completes the proof.

According to Lemma 1, when a process is approximated as a weighted average of M models, using the weighted average of control actions from each model ensures the feasibility of both state and control vectors of the process. In other words, the controllers of the these models collaborate effectively to control the process.

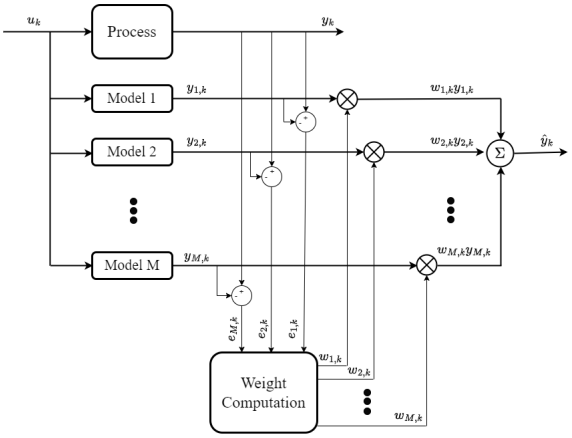

Fig. 4. The MMAC architecture.

3.1 Multiple-model Adaptive Control

Multiple-model Adaptive Control (MMAC) is a method applied in the fields of control theory, signal processing, and filtering (Magill, 1965; Baram and Sandell, 1978; Aguiar et al., 2008; Murray-Smith and Johansen, 2020). It proves especially valuable in scenarios where a single model falls short in accurately capturing the complex dynamics of a process. Such limitations often arise due to factors like uncertainties, or transitions between different operating modes. The core concept behind MMAC involves employing a set of M distinct mathematical models, as shown in Figure 4. Each of these models is tailored to capture the dynamics and behaviors of the process within specific operating modes or predefined ranges of parameter uncertainties.

At each time step $k \in \mathbb{N}$, the predicted outputs of all M models, denoted as $y_{1,k}, \ldots, y_{M,k}$, are computed using the same control input u_k as the process. The difference

between the output of each model and the actual process output, represented by y_k, allows for the derivation of model errors as follows:

$$e_{i,k} = y_k - y_{i,k}, \quad i = 1, \ldots, M.$$

These errors are then used to calculate the likelihood of each model accurately characterizing the process. The resulting likelihood values provide guidance for the assignment of suitable weights to individual models, with models having lower probabilities receiving proportionally lighter weights. This weighting strategy enables the selection of a model or a combination of models that most effectively represents the input-output behavior of the process at time step $k \in \mathbb{N}$.

MMAC performs two primary tasks at each time step: firstly, it computes probabilities through a Bayesian approach, and secondly, it assigns weights based on these probabilities. The Bayes theorem is used in the iterative computation of the current probability associated with each model being the true representation of the process. Specifically, at the kth iteration, the probability for the ith model is calculated as

$$p_{i,k} = \frac{\exp\left(-e_{i,k}^\top \Lambda e_{i,k}\right) p_{i,k-1}}{\sum_{j=1}^{M} \exp\left(-e_{j,k}^\top \Lambda e_{j,k}\right) p_{j,k-1}}. \tag{4}$$

Here, the matrix Λ is a tuning parameter for adjusting the convergence rate of the probabilities. Larger values of Λ magnify the impact of model errors, thereby accelerating convergence toward a single model. Note that the recursion in (4) is initialized with equal probabilities $1/N$ assigned to all the models. One of the key advantages of this algorithm is its computational efficiency. Moreover, it has the capability to systematically reject poor models. This feature allows for a diverse range of models without significantly compromising performance, even during the initial stages (Yu et al., 1992).

Using the probability calculated for individual models at each recursive step, suitable weights are assigned to each model. A model with a higher probability receives a correspondingly higher weight, while a model with a lower probability is assigned a reduced weight. Considering the inherent uncertainties in real-world system, it is unlikely that any single model will perfectly match the process. Hence, it becomes essential to effectively combine models.

Since (4) operates recursively, once the probability of any model reaches zero, there is no way for that probability to become nonzero in future time steps. To address this and ensure the continued participation of every model in the calculation, an artificial cutoff threshold, denoted as $p_{\min}$, is introduced. For models with probabilities falling below the threshold, i.e., $p_{i,k} < p_{\min}$, their probabilities are reset to $p_{i,k} = p_{\min}$, and are consequently excluded from the weighting process (Athans et al., 1977). Thus, at the kth iteration, the ith model is assigned a weight $w_{i,k}$ using the following scheme:

$$w_{i,k} = \begin{cases} \dfrac{p_{i,k}}{\sum_{j=1}^{M} p_{j,k}}, & p_{i,k} \geq p_{\min}, \\ 0, & p_{i,k} < p_{\min}. \end{cases}$$

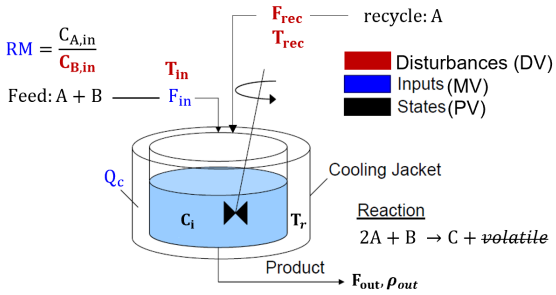

Fig. 5. Schematic diagram of the CSTR and the surrounding cooling jacket.

4. EXPERIMENT SETUP

The proposed algorithm has been tested on a benchmark CSTR (Continuous Stirred-Tank Reactor) model, which simulates a nonlinear complex reaction between components A and B to produce C. The problem is purposefully made up to address the real-world challenges for evaluation of the algorithms proposed in this paper. It is able to convey sufficient complexities, while also providing a reasonable number of input and output signals, including manipulated variables (MVs), process variables (PVs), and disturbance variables (DVs).

4.1 Process description: CSTR example

Figure 5 illustrates a schematic diagram of the vessel and the surrounding cooling jacket. A nonlinear exothermic and irreversible reaction, $2A + B \rightarrow C + volatiles$, takes place in the vessel, which is assumed to be always perfectly mixed.

The feed stream of mixture of reagents A and B enters the tank at the mole ratio of $RM = \frac{concentration(A)}{concentration(B)}$ and volumetric rate F_{in}. The product stream C exits continuously at the volumetric rate F_{out}, which may include some impurities of not reacted components A and B. Perfect removal of volatile components are assumed resulting in a variable outlet density (ρ_{out}). Outlet volumetric rate is calculated under assumption of constant volume of the reacting liquid. There is also a recycled line of component A from a downstream unit in the process, which is considered an external bounded disturbance.

The process base-model is constructed by mass and energy balance equations, formulated in the compact form of (1). The concentration of all components C_A, C_B, C_C together with the reactor temperature T_r are assumed as the four state variables of the system. Alternatively, the process variables (PVs) within the system consist of the four state variables, as well as the product flow rate F_{out} and density ρ_{out}, which are either directly measurable or can be estimated with high accuracy.

The feed mole ratio RM, feed flow rate F_{in}, and set point of reactor temperature $T_{r,SP}$, are considered as process manipulated variables (MVs). There is an inner PID controller on the cooling jacket loop, which receives the desired reactor temperature $T_{r,SP}$ and manipulates the cooling power Q_c.

The process is also subject to various external disturbances (DVs), which are outlined below:

- feed B concentration $C_{B,in}$ (measurable)
- feed fluid temperature T_{in} (measurable)
- recycled fluid temperature T_{rec} (unmeasurable)
- recycled fluid flow rate F_{rec} (unmeasurable)

Detailed formulation of the state transition, control input and output functions are provided in the following GitHub link: https://github.com/Soroush-Git/ADCHEM2024---CSTR-example.

4.2 Submodels definitions

The main source of uncertainty in the proposed CSTR process can be defined in the kinetic formulation of chemical reaction. For the reaction of $2A + B \rightarrow C$, the rate equation can be written as $k_r f_r(C_A, C_B)$, where f_r is a nonlinear function of the concentration of the inlet components A and B, and k_r is the corresponding rate constant representing the specific reaction rate at a given temperature (see the above-mentioned GitHub link). Hence, the tuning of k_r, which is a function of activation energy E_r and exponential factor k_0 will account for the parametric uncertainty, whereas the definition of the kinetic function f_r will be employed to address significant structural uncertainty.
We introduce four different submodels according to different formulations of these parametric and structural uncertainties:

	f_r	k_0	E_r
Actual Process	$C_A C_B$	k_0^*	E_r^*
submodel1	$C_A^2 C_B$	$\omega(0.1k_0^*, 0.5)$	$\omega(1.05E_r^*, 0.5)$
submodel2	C_B^2	$\omega(2k_0^*, 0.5)$	$\omega(1.05E_r^*, 0.5)$
submodel3	$C_A C_B^2$	$\omega(2k_0^*, 0.5)$	$\omega(0.65E_r^*, 0.5)$
submodel4	C_A^2	$\omega(2k_0^*, 0.5)$	$\omega(1.05E_r^*, 0.5)$

where $\omega(\mu, \sigma^2)$ represents a normally distributed random variable with mean value of μ and variance σ^2. In fact, in each epoch, a value is randomized from this normal distribution.

The submodels are tuned and adjusted using experimental data from the actual process. However, they may not respond adequately to all operating conditions. The actual process can always be situated within the boundaries of this ensemble of models, which satisfy the conditions of Lemma 1.

4.3 RL *agent objectives*

The primary control objective for the RL agent is to ensure safe and efficient operation while ultimately maximizing the reactor's profitability.
In this context, the process adheres to a barrier function that enforces constraints on both PVs:

$$1155[kg/m^3] \leq \rho_{out} \leq 1165[kg/m^3]$$
$$10[m^3/h] \leq F_{out} \leq 20[m^3/h]$$
$$520[K] \leq T_r \leq 570[K]$$

and MVs:

$$1 \leq RM \leq 2$$
$$10[m^3/h] \leq F_{in} \leq 20[m^3/h]$$
$$520[K] \leq T_{r,SP} \leq 570[K]$$

The process profit is calculated as the total yield, which is obtained by multiplying the product flow by the concentration of the desired product, while deducting the costs associated with feed and energy, as shown below:

$$Profit = -\frac{5}{3600} F_{in} C_A M_A - 0.045 T_r + \frac{10}{3600} F_{out} C_C M_C \tag{5}$$

where $M_A = 62$ and $M_C = 254$ are the molar mass [g/mol] of components A and C.
The robust RL agent seeks to approximate the optimal policy π_θ through iterative steps, aiming to maximize a reward function. The reward is the actual profit (5), but it is penalized by a negative gain when the barrier function is violated:

$$\begin{aligned} Reward = Profit \\ - 40\{max(0, \rho_{out} - 1165) + max(0, 1155 - \rho_{out})\} \\ - 120\{max(0, F_{out} - 20) + max(0, 10 - F_{out})\} \\ - 120\{max(0, T_r - 570) + max(0, 520 - T_r)\}. \end{aligned} \tag{6}$$

5. RESULTS DISCUSSION

In this section, we will examine the advantages and disadvantages of the proposed algorithms and compare the results with the conventional RTO-APC solutions, which are widely used in the energy industries.
Three different RL agent will be analyzed:

- Regular RL: trained on a model of the process which is tuned on some historical data (the best base-model).
- Robust RL1: trained on the ensembled of the base-models with an aggregation on observation by MMAE.
- Robust RL2: trained on the ensembled of the base-models with an aggregation on actions by MMAE.

All approaches adopt the same architecture of the Actor-Critic networks to ensure a fair comparison. Both the Actor and Critic are represented by feedforward neural networks with multiple fully connected layers, each consisting of 256 neurons in the input direction, which are concatenated towards the output layer(s).
RL agent is trained through 100 epochs, each encompassing 48 hours of environment simulation (corresponding to the process) with random initialization of state variables and disturbances.
For each RL agent, this task is completed within 40 minutes on a standard computer equipped with an Intel(R) Core(TM) i7-1165G7 CPU running at 2.80GHz, without the use of GPU parallelization.
We also introduced two key performance indicators (KPIs) to evaluate each control solution. The first one is the percentage of time the process either complies with the constraints, along with the count of unsatisfied constraints ranked by their level of significance. In the example CSTR process, the outlet volumetric flow rate constraint takes precedence over product density, as it receives a higher penalty in the reward function (6). The second one is the profit for the period during which all constraints are satisfied.
Figure 6 compares the product flow rate over a 10-day simulation with a change in production mode, so-called grade change, at the day 5. All three RL solutions are

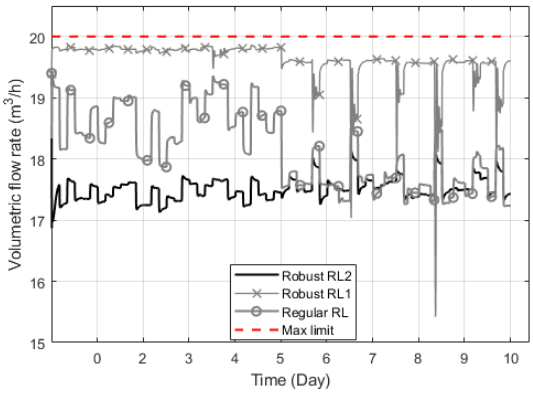

Fig. 6. Product flow rate over 10-day simulation with a grade change at the day 5.

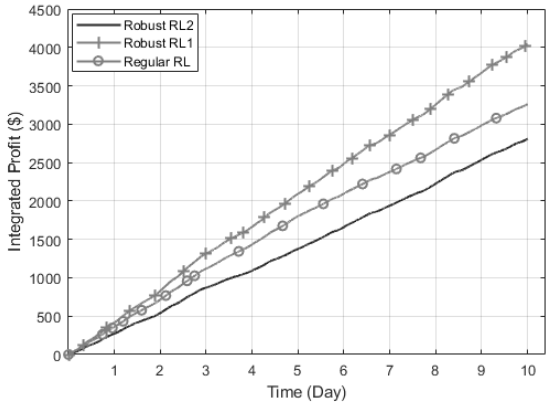

Fig. 7. Profit over 10-day simulation with a grade change at the day 5.

able to fully respect the constraint on product flow rate. Regular RL is able to push more toward the upper limits which can results in a higher profit as illustrated in Figure 7. However, the constraint on product density is not well guaranteed by Regular RL. As it is shown in Figure 8, although Regular RL can effectively push the process operating point closer to the upper limits, there is no guarantee that limit violations will be completely avoided, even during the initial 5 days when the process operating conditions appear to align closely with the model used by Regular RL. On the contrary, during the initial 5 days (first mode of operation), Robust RL1 successfully maintains the constraints within their limits and surpasses Regular RL in terms of the first KPI. This is primarily due to Robust RL1's approach, which considers a weighted average of observation sets derived from submodels for agent training process, prioritizing more critical scenarios. While this may slightly impact performance from a profit perspective (Figure 7), it ensures the constraint limits are maintained. The challenge becomes more evident on day 5 when the process undergoes a change in production mode. This shift significantly pushes the operating point beyond the validity of the models employed in both Regular RL and Robust RL1, resulting in a severe constraint violation (Figure 8).

In such conditions, the control actions generated by the Robust RL2 approach demonstrate a more reliable and robust performance compared to Robust RL1 and Regular RL. This outcome aligns with expectations, as it employs the robust control actions generated from an ensemble of agents optimally weighted based on the actual process operating conditions using the MMAE approach. This approach can yield substantial profit when the actual process closely aligns with the agent with the highest score. However, it also ensures performance even if there is a slight deviation, as long as it remains within the boundaries of the submodels.

In this analysis, all three RL approaches were trained using identical numbers of epochs and simulation lengths. However, they exhibited varying sample complexities, leading to larger memory requirements and computational costs. Specifically, the results indicate that training RL necessitates less complexity than RL1, which in turn requires less complexity than RL2.

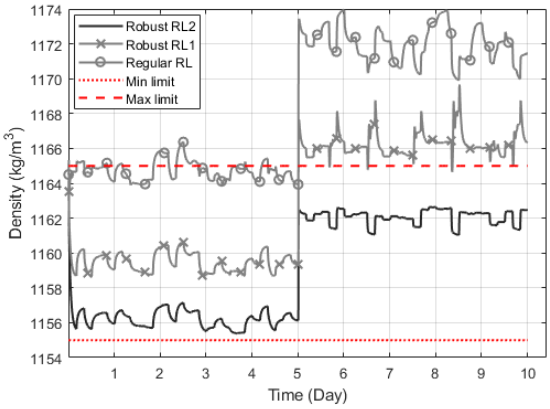

Fig. 8. Product density over 10-day simulation with a grade change at the day 5.

The proposed Robust RL algorithms are compared with a conventional RTO-APC approach. Three different RTO-APC approaches are tested. RTO-APC1 and RTO-APC2 are based on the first two most probable submodels selected by the MMAE algorithm. The third one is based on the perfect model of the system. RTO-APC approaches consistently yield higher profits by operating closer to boundary conditions. However, they struggle to consistently satisfy constraints. For instance, RTO-APC1 can poorly ensure constraints in first five days (mode1 of operation), while RTO-APC2 fails during grade changes. The optimal RTO-APC, of course, ensures performance in about 98% of operating conditions.

6. CONCLUSIONS

The proposed robust RL algorithm enhances the robustness of the RL control agent against model uncertainties and mode changes in the actual process. It can minimize the number of constraint violations for those conditions where Regular RL approaches or the current conventional control algorithms like APC-RTO are not capable of effectively handling the complexities of the process.

REFERENCES

Adetola, V. and Guay, M. (2010). Integration of real-time optimization and model predictive control. *Journal of*

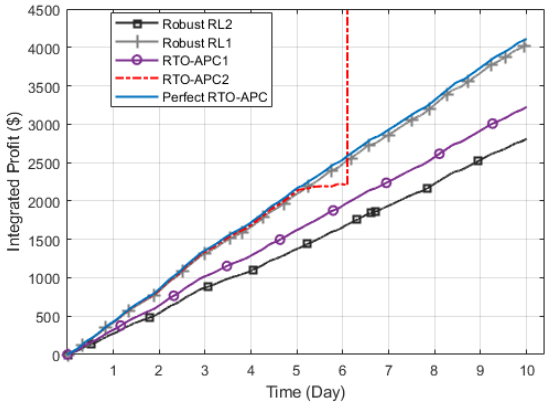

Fig. 9. Profit over 10-day simulation with a grade change at the day 5: .

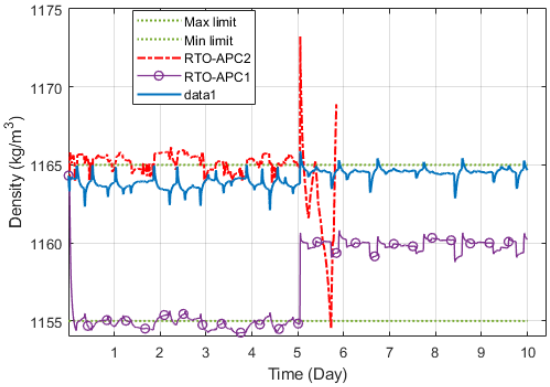

Fig. 10. Product density over 10-day simulation with a grade change at the day 5.

Process Control, 20(2), 125–133.

Aguiar, A.P., Hassani, V., Pascoal, A.M., and Athans, M. (2008). Identification and convergence analysis of a class of continuous-time multiple-model adaptive estimators. *IFAC Proceedings Volumes*, 41(2), 8605–8610.

Athans, M., Castanon, D., Dunn, K.P., Greene, C., Lee, W., Sandell, N., and Willsky, A. (1977). The stochastic control of the F-8C aircraft using a multiple model adaptive control (MMAC) method–Part I: Equilibrium flight. *IEEE Transactions on Automatic Control*, 22(5), 768–780.

Baram, Y. and Sandell, N. (1978). An information theoretic approach to dynamical systems modeling and identification. *IEEE Transactions on Automatic Control*, 23(1), 61–66.

Bertsekas, D. (2019). *Reinforcement Learning and Optimal Control*. Athena Scientific.

Campos, M., Teixeira, H., Liporace, F., and Gomes, M. (2009). Challenges and problems with advanced control and optimization technologies. *IFAC Proceedings Volumes*, 42(11), 1–8.

De Souza, G., Odloak, D., and Zanin, A.C. (2010). Real time optimization (RTO) with model predictive control (MPC). *Computers & Chemical Engineering*, 34(12), 1999–2006.

Derman, E., Mankowitz, D., Mann, T., and Mannor, S. (2020). A bayesian approach to robust reinforcement learning. *Uncertainty in Artificial Intelligence*, 648–658.

Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1), 1437–1480.

Grondman, I., Busoniu, L., Lopes, G., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems nd Cybernetics*, 42(6), 1291–1307.

Kadam, J., Marquardt, W., Schlegel, M., Backx, T., Bosgra, O., Brouwer, P., Dünnebier, G., Van Hessem, D., Tiagounov, A., and De Wolf, S. (2003). Towards integrated dynamic real-time optimization and control of industrial processes. *Proceedings Foundations of Computer-aided Process Operations*, 593–596.

Konda, V. and Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in neural information processing systems*, 12.

Lewis, F.L., Vrabie, D., and Vamvoudakis, K.G. (2012). Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems Magazine*, 32(6), 76–105.

Magill, D. (1965). Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control*, 10(4), 434–439.

Morimoto, J. and Doya, K. (2005). Robust reinforcement learning. *Neural computation*, 17(2), 335–359.

Murray-Smith, R. and Johansen, T. (2020). *Multiple Model Approaches to Nonlinear Modelling and Control*. CRC press.

Panaganti, K., Xu, Z., Kalathil, D., and Ghavamzadeh, M. (2022). Robust reinforcement learning using offline data. *Advances in neural information processing systems*, 35, 32211–32224.

Peters, J. and Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71(7-9), 1180–1190.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. (2016). Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*.

Recht, B. (2019). A tour of reinforcement learning: The view from continuous control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2, 253–279.

Tamar, A., Mannor, S., and Xu, H. (2014). Scaling up robust MDPs using function approximation. *International conference on machine learning*, 181–189.

Tessler, C., Efroni, Y., and Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. *International Conference on Machine Learning*, 6215–6224.

Wang, Y. and Zou, S. (2021). Online robust reinforcement learning with model uncertainty. *Advances in Neural Information Processing Systems*, 34, 7193–7206.

Yu, C., Roy, R.J., Kaufman, H., and Bequette, B.W. (1992). Multiple-model adaptive predictive control of mean arterial pressure and cardiac output. *IEEE Transactions on Biomedical Engineering*, 39(8), 765–778.