Data-Efficient Reinforcement Learning from Controller Guidance with Integrated Self-Supervision for Process Control

Nicolas Bougie [*] **Takashi Onishi** [*,**] **Yoshimasa Tsuruoka** [*,***]

[*] *NEC-AIST AI Cooperative Research Laboratory, National Institute of Advanced Industrial Science and Technology, Tokyo, Japan (e-mail: nicolas-bougie,takashi.onishi,yoshimasa.tsuruoka@aist.go.jp).*
[**] *Data Science Research Laboratories, NEC Corporation, Kanagawa, Japan*
[***] *Department of Information and Communication Engineering, The University of Tokyo, Tokyo, Japan*

Abstract: Model-free reinforcement learning methods have achieved significant success in a variety of decision-making problems. In fact, they traditionally rely on large amounts of data generated by sample-efficient simulators. However, many process control industries involve complex and costly computations, which limits the applicability of model-free reinforcement learning. In addition, extrinsic rewards are naturally sparse in the real world, further increasing the amount of necessary interactions with the environment. This paper presents a sample-efficient model-free algorithm for process control, which massively accelerates the learning process even when rewards are extremely sparse. To achieve this, we leverage existing controllers to guide the agent's learning — controller guidance is used to drive exploration towards key regions of the state space. To further mitigate the above-mentioned challenges, we propose a strategy for self-supervision learning that lets us improve the agent's policy via its own successful experience. Notably, the method we develop is able to leverage guidance that does not include the actions and remains effective when the existing controllers are suboptimal. We present an empirical evaluation on a vinyl acetate monomer (VAM) chemical plant under disturbances. The proposed method exhibits better performance than baselines approaches and higher sample efficiency. Besides, empirical results show that our method outperforms the existing controllers for controlling the plant and canceling disturbances, mitigating the drop in the production load.

Keywords: Reinforcement learning control; Process control; Chemical plant control; Co-Learning and self-learning; Artificial intelligence

1. INTRODUCTION

Novel artificial intelligence (AI) strategies for process control have the potential to significantly improve safety and efficiency in a wide range of industrial domains such as chemical plants (Machida et al., 2016). However, the application of these algorithms has been fairly modest, and they have not yet established a significant position in the process industry. The majority of current systems involve skilled operators and/or are operated by simple feedback controllers such as proportional-integral-derivative (PID) controllers. Nevertheless, these reactive control strategies are unable to cope with disturbances such as weather changes and lack flexibility. In detail, PID controllers are suitable to maintain the system in a steady state but fall short when encountering unexperienced steady states or disturbances. Moreover, they require a high degree of familiarity with the task in order to select appropriate parameters.

Given the limitations of existing industrial controllers, a key challenge for enhancing safety and efficiency in process control industries is dealing with unexperienced states or situations (e.g. disturbances). For instance, chemical plants often experience external disturbances such as day/night or heavy rain, as well as internal disturbances such as sudden changes in feed composition or pressure of a component. In such situations, since PID controllers are likely to be ineffective, it is common practice to ask a skilled operator to manually adjust and control the chemical plant to restore the stability of the plant. These procedures are typically complex and resource-intensive, and they often cause costly interruptions to normal operations.

Model-free reinforcement learning (RL), on the other hand, is a learning-based strategy that is able to adapt its behavior to changes in the environment and novel situations. A few prior studies have explored the use of RL for controlling a chemical plant (Zhu et al., 2020; Cui et al., 2018; Kubosawa et al., 2021) and dealing with disturbances (Kubosawa et al., 2019). However, the flexibility and adaptability of RL comes with the need of experiencing a huge number of interactions to converge upon a satisfactory policy, also known as the "sample-efficiency" issue. Although training an agent for millions or billions of steps is acceptable in sample-efficient simulators such as Atari games (Bellemare et al., 2013), this often becomes intractable in the process control industry. Process control tasks typically involve complex and changing dynamics, huge state-action spaces, and costly computations, which entails a low sample efficiency. In addition, in real-world tasks, rewards are naturally sparse - zero for most of the time steps, further increasing the number of necessary training steps.

This motivates the need to build sample-efficient RL algorithms for process control. A prior study has attempted to employ a multi-agent framework to deal with large action spaces and reduce the problem complexity (Cui et al., 2018). This idea was then extended to alleviate the curse of dimensionality by introducing Fastfood kernel approximation (Zhu et al., 2020). In a different spirit, it is possible to learn probabilistic transition models using Gaussian processes and incorporate model uncertainties into long-term predictions (i.e. MPC) (Kamthe and Deisenroth, 2018). RL has also been used to tune the initial values of PID controllers (Qin et al., 2018). Another approach models the long-term value of a state via a Markov decision process and feeds it back into the controller (Cheng et al., 2004). However, learning a task from scratch can still require a prohibitively time-consuming amount of exploration of the state-action space in order to learn a good policy. To overcome these difficulties, some work in the field of industrial control has been devoted to reduce sample complexity by initializing their agent from historical data (Li et al., 2020) or cloning behaviors of existing algorithms (Jia et al., 2019). Nevertheless, these approaches require access to the actions of an optimal expert and are limited to specific applications.

In this paper, we aim to build a sample-efficient intelligent system for process control. Our key insight is to leverage existing controllers (e.g. PIDs) in order to encourage a RL agent to visit task-relevant regions of the state space. Namely, our agent observes the states visited by the existing controllers, and uses the information gleaned from these observations to identify potentially meaningful regions, drastically improving its performance at the onset of learning. However, we also account for the possibility that the existing controllers will produce suboptimal guidance when facing unexperienced states (for example caused by disturbances). While suboptimal trajectories can still encode domain knowledge on system dynamics and control, we argue that it is necessary to enrich controller guidance with the agent's own successful experience. Thus, we further propose a self-supervision method that can be used along with controller guidance, greatly accelerating the agent's training. Self-supervision is achieved by employing an episodic memory and keeping meaningful trajectories in terms of return. Together, they provide a flexible and data-efficient approach for process control, allowing learning even when rewards are extremely sparse. Contrary to most imitation learning approaches, the present work does not necessitate access to the expert actions — it solely relies on states, and can extract guidance from suboptimal controllers. Across a range of disturbance scenarios on a VAM plant (Machida et al., 2016), the proposed method learns significantly faster than standard RL baselines and produces higher average return.

2. PROPOSED METHOD

We propose a sample-efficient agent capable of solving process control problems, including when extrinsic rewards are sparse. Our approach explores the idea of combining controller guidance and self-supervision with the agent's own experience. The method solely relies on states as guidance, which broadens its applicability to practical tasks where guidance can be generated by observing existing controllers (e.g. PIDs) or operators. The key insight is to guide exploration towards states/regions that are potentially taking the agent in the appropriate direction (e.g. correct disturbances and/or maintain the system in a steady state). Note that we refer to the existing controllers as *experts* although they are likely to be suboptimal when facing changes in the environment such as disturbances.

To this end, the proposed method learns an ensemble of N_d discriminators. The discriminators are trained to differentiate between the provided guidance — relevant states, and the agent's growing experience. We propose to predict if a transition $s^* \rightarrow s$ is taking the agent in the correct direction, where s^* is a batch of the last states and s is the current state. The intuition behind this formulation is that predicting the relevance of the current state is challenging without accessing past trajectories. On the other hand, we can accurately predict, on the basis of previously visited states s^*, whether the transition looks similar to the existing controllers' intentions and is relevant for reaching the goal being pursued.

In detail, given a batch of previously experienced states s^* and the current state s, the discriminators are trained to differentiate between expert state trajectories (labeled as 1) from a replay memory M_E and the agent's growing dataset R (labeled as 0). In this work, the agent is trained to "fool" the discriminators into thinking itself is the expert. Throughout learning, the discriminators in the ensemble are randomly initialized, and each discriminator D_ϕ is trained according to the loss function defined below, with respect to its parameters ϕ:

$$\begin{aligned}\mathcal{L}(D_\phi, M_E, R) = \ & \mathbb{E}_{(s^*,s)\sim M_E}[\log(D_\phi(s^*,s))] \\ & + \mathbb{E}_{(s^*,s)\sim R}[\log(1 - D_\phi(s^*,s))] \quad (1)\end{aligned}$$

Each discriminator is trained independently by drawing examples at random with replacement from M_E and R. Since our approach operates in the low data regime — where not many expert data are available, we embrace dropout and mixup training (Zhang et al., 2018) as regularization techniques. Therefore, we introduce a dropout layer before every weight layer of our discriminator networks. We also utilize mix-up regularization that creates new examples as convex combinations of training points and labels. While this technique has been primarily used on images, we found this technique extremely effective on sensor data. Namely, mixup constructs virtual training examples $(\overline{s^*}, \overline{s}, \overline{y})$ as follows:

$$\overline{s^*} = \lambda s_1^* + (1-\lambda)s_2^* \ ; \ \overline{s} = \lambda s_1 + (1-\lambda)s_2 \quad (2)$$

$$\overline{y} = \lambda y_1 + (1-\lambda)y_2 \quad (3)$$

where $\lambda \in [0,1]$ controls the extent of mixup $\lambda \sim Beta(\alpha,\alpha)$, (s_1^*, s_1, y_1) and (s_2^*, s_2, y_2) are two examples drawn at random from our training datasets M_E and R, and y_1 and y_2 are the labels (i.e. expert or policy) of the examples.

We can now employ the fitted discriminators to generate a reward that will encourage the agent to explore states around potentially meaningful regions of the state space. The exploration bonus b_t, which is further summed with the task reward r_t, is computed by using the outputs of the discriminators:

$$b_t(s^*, s) = F(w(s^*, s, D_{\phi_1})D_{\phi_1}(s^*, s), ..., w(s^*, s, D_{\phi_{N_d}})D_{\phi_{N_d}}(s^*, s)) \tag{4}$$

where $w(s^*, s, D_\phi)$ denotes the weight of tuple (s^*, s, D_ϕ) and the aggregation function F is a hyperparameter of our method. Theoretically, $F = mean$ would be a good choice; however, in practice it is prone to learning inaccurate behaviors due to the small amount of available expert data. Empirically, we found that $F = max$ works well as a robust substitute to mean, rewarding the agent for exploring regions that either are considered as relevant by the discriminators or the ones for which the ensemble has high spread. In other words, it captures both the expert's intentions and uncertainty. We assign a weight $w(s^*, s, D_\phi)$ to the output of the discriminator $D_\phi(s^*, s)$ in order to take into account its confidence. It was shown that the use of dropout can be interpreted as a Bayesian approximation of Gaussian process (Gal and Ghahramani, 2016). Therefore, to estimate predictive confidence, we collect the results of stochastic forward passes through the discriminator network:

$$w(s^*, s, D_\phi) = \text{clip}(\left[\frac{\beta}{\mathbb{E}_{d_j \sim D}[D_\phi^{d_j}(s^*, s) - p]^2}\right], 0.1, 1) \tag{5}$$

where $D_\phi^{d_j}(s^*, s)$ represents the discriminator with dropout mask d_j, D is a set of dropout masks, β is a hyperparameter of our method, and p is the predictive posterior mean, $p = \mathbb{E}_{d_j \sim D} D_\phi^{d_j}(s^*, s)$. Since the forward passes can be done concurrently, the method results in a running time identical to that of standard dropout.

2.1 Leveraging Controller Guidance

As discussed above, we maintain a replay buffer M_E where we store *expert* data. Expert state trajectories are used to determine if a transition $s^* \rightarrow s$ is taking the agent in the correct direction. Rather than relying on standard human demonstrations that are cost and time prohibitive to collect in process control tasks, we take advantage of existing controllers — *controller guidance*.

The most straightforward way to leverage controller guidance is to run the plant under the control of the existing controllers (e.g. PIDs) and collect *steady states*. In this setting, the existing controllers strive to maintain the system in a steady state, where the production of the plant is optimal and conditions are stable. The collected tuples (s^*, s) are stored inside the replay buffer, $M_E \leftarrow M_E \cup \{(s_0^*, s_0), (s_1^*, s_1), ...\}$. This type of guidance allows the learner to identify the regions that must be reached in order to maintain the system in steady/optimal conditions.

In some experiments we attempt to leverage the expert trajectories under disturbances to enrich the set of initial examples. Thus, we collect imperfect behaviors by observing the controllers attempting to correct disturbances. To do so, we run the simulator under disturbances and collect state trajectories. Note that most disturbances cannot be corrected; nevertheless, even imperfect trajectories can still encode domain knowledge on system dynamics and control. This necessity to learn from suboptimal controllers further highlights the importance of building a novel type of guidance that does not include the expert actions. We found that distinguishing *expert* states from agent states is less prone to overfitting suboptimal behaviors than relying on state-action pairs from an expert (see Sect 3.6.2).

2.2 Self-Supervision

In addition to controller guidance, we propose to incorporate the agent's past good trajectories, a form of self-supervision. The intuition behind this approach is that controller guidance drastically improves performance at the onsent of the training phase. Once the agent acquires knowledge about the task, we aim to prioritize the agent's past good experience over suboptimal controller guidance. To this end, we propose to store past episodes with large returns in an episodic memory: $M_R = \{(\tau, R_t, done), ...\}$, where τ is a state trajectory, $R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k$ is the discounted sum of rewards, and *done* indicates if the trajectory was successful.

The episodic memory has a limited capacity K. At every step, a new trajectory might be added to the memory. What to do when the capacity is exceeded? We substitute the element with the lowest return R_t in memory with the current element. Until the entire memory is filled with successful trajectories, this strategy is solely applied among unsuccessful trajectories. This way, there are still more fresh elements in memory than older ones, but the older meaningful elements are not totally neglected.

To train the discriminators, we extract transitions $s^* \rightarrow s$ from M_R, and then employ a mixture of examples from M_E and M_R as expert data. At the beginning of the training, we only provide the agent controller guidance so $|M_R| = 0$. The agent samples from the trajectories in M_E with probability δ, M_R with probability ρ, and from steady data in M_E with probability $1 - \rho - \delta$. As the agent acquires more knowledge about the task — controller guidance is mostly needed to collect the first few successful trajectories, we decrease the probability of replaying controller guidance and increase the probability ρ of replaying experienced states. δ and ρ are annealed according to the following rules each time a new successful trajectory is added to the episodic memory:

$$\delta = \delta - (\frac{\rho_0}{K}); \rho = \rho + (\frac{\rho_0}{K}) \tag{6}$$

where ρ_0 is the initial value of ρ, and K is the maximum size of the episodic memory. Note that in terms of implementation, examples from M_R are merged into M_E when training the discriminators — M_R and M_E are treated as a single memory $M_E \leftarrow M_E \cup M_R$ in Eq. 1 and transitions are sampled according to Eq 6.

3. EXPERIMENTS

We evaluate the proposed method on a chemical plant simulator, which replicates a real vinyl acetate monomer (VAM) plant. Our goal is to evaluate the agent's capability to control the chemical plant under disturbances in terms of return and data efficiency.

3.1 Chemical Plant Simulator

Experiments are conducted on a VAM plant environment under disturbances, which reflects the characteristics and practical problems of real plants. The simulator is comprised of eight components for materials feeding, reacting, and recycling. The process is observed via 109 sensors that measure the volume, flux, temperature, concentration, and pressure of the chemical substances. In order to complete the task, the agent has to: 1) avoid failures in equipment that can be triggered by disturbances, 2) stabilise and correct internal or external disturbances — recover from disturbances, and 3) maintain the process in a steady state. The environment includes 19 disturbance scenarios that can be used to evaluate these properties.

Concretely, the **state** space consists of the sensor readings. To facilitate the learning of agents, we use an observation normalization scheme. That is, we whiten each dimension by subtracting the running mean and then dividing by the running standard deviation. The **action** space consists of a set of PIDs to control — these PIDs are selected based on their relevance with regards to the scenario. The ranges of actions are defined as $[-x_1\%, +x_2\%]$ from the initial values. In our experiments, we set $x_1 = 0.60$ and $x_2 = 1.35$. The agent interacts with the environment once a minute for 60 virtual minutes, which corresponds to one episode. In the absence of domain knowledge and to replicate real-world problems where rewards are naturally sparse, a general-purpose choice is to set the **reward** function as:

$$r(x, x_t) = \begin{cases} 1.0 & \text{if } (\frac{|x-x_t|}{x_s} < \epsilon) \\ 0.0 & \text{otherwise} \end{cases} \tag{7}$$

where x is the current state, x_t is the target state, x_s is a steady state value, and ϵ is a threshold value. In practice, we set $x_t = x_s$ and $\epsilon = 0.01$.

3.2 Disturbance Scenarios

We now describe the three disturbance scenarios that we use to evaluate the presented method:

- Change Feed Pressure AcOH ("Pressure AcOH"): raw material acetic acid feed composition is changed due to condition changes of the acetic acid plant. The intensity level varies randomly between $[1, 50]$. The agent controls the PIDs: PC130, LC130, FC130, FC170, and PC210.
- Change Feed Pressure C2H4: raw ethylene feed pressure is changed due to condition changes of the ethylene plant. The intensity level varies randomly between $[70, 140]$. The agent controls the PIDs: PC130 and LC130 ("Pressure C2H4-2"), or, PC130, LC130, FC130, and TC150 ("Pressure C2H4-4").
- Day and Night ("Day/Night"): a day and night cycle leads to atmosphere changes, resulting in non-steady conditions and fluctuations in internal temperatures. The intensity level varies randomly between $[1, 50]$. The agent controls the PIDs: PC130, LC130, FC310, TC150, and FC170.

3.3 Experimental Details

As our policy learning method, we rely on proximal policy optimization (PPO) (Schulman et al., 2017). We refer to our algorithm as CGS — Data-Efficient RL from **C**ontroller **G**uidance with Integrated **S**elf-Supervision. The actor and critic networks consist in 3 fully-connected layers with 128 hidden units. Tanh is used as the activation function. Training is carried out with a fixed learning rate of 7^{-4} using the Adam optimizer (Kingma and Ba, 2014), with a batch size of 128. The policy is trained for 4 epochs after each episode. In all our experiments, the discriminators' confidence is estimated using 100 dropout masks with $p = 0.2$. In general cases, $N_d = 3$ is sufficient to yield satisfactory performance. The discriminator networks are updated 5 times after each episode using batches of size 128, except for the first 10 episodes where the networks are updated for 100 times. To create s^*, we stack the four most recently experienced states. We set $\beta = 0.7$, $\rho_0 = 0.30$, and $\gamma = 1$. The exploration reward is normalized by dividing it by a running estimate of the standard deviations of the exploration returns. The exploration bonus is scaled by a factor 0.1 and the extrinsic reward by a factor 0.9. For mixup training, we set $\alpha = 1$. Our method uses 100 steady states as training examples (referred to as *ste*), which can be enriched with examples extracted from 15 expert trajectories under disturbance (referred to as *trj*). We refer to the self-supervision algorithm as *sel*, and we set the size of the episodic memory $K = 20$.

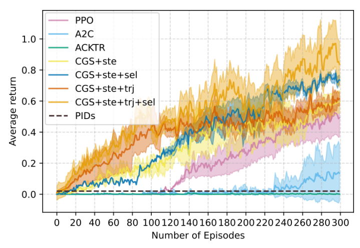
3.4 Feed Pressure Disturbance Scenarios

We first perform experiments on three internal disturbance tasks: pressure AcOH, pressure C2H4-2, and pressure C2H4-4. We evaluate our method with different settings: CGS+*ste*, CGS+*ste*+*trj*, CGS+*ste*+*sel*, and CGS+*ste*+*trj*+*sel*. Moreover, we compare our method against several baselines including PPO (Schulman et al., 2017), A2C (Mnih et al., 2016), ACKTR (Wu et al., 2017), and the existing PIDs. We show learning curves in Figure 1. As can be observed, our method learns faster than other approaches in all the tasks. In addition, CGS achieves higher average return than baseline methods including the existing controllers. The results show that controller guidance (*ste* and *trj*) drastically improves the agent's performance at the onset of its training. On the other hand, self-supervision allows the agent to constantly receive supervision during its training, which leads to a continuous improvement of the agent's performance. Notably, Figure 1b and Figure 1c highlight that the gap between our approach and the others is increasing with the degree of complexity of the task — the number of PIDs to control.

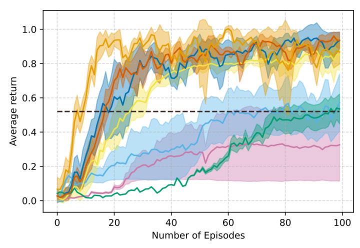
We further report some examples of learned policies in Figure 2. The solid lines indicate the actual measured distance to the target state. Note that to ensure the readability of the figure, we show examples of policies learned by the following RL methods: CGS+*ste* + *trj* + *sel*, PPO, A2C, and ACKTR. This experiment highlights that quickly after the onset of a disturbance, our agent can correct it and return to a steady state (dashed line), mitigating the drop in the production load.

3.5 Day and Night Disturbance Scenario

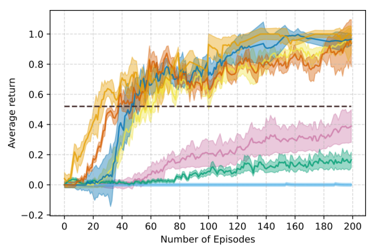
In addition to the first set of experiments, we evaluate our methodology on a different type of disturbance: day/night.



(a) Pressure AcOH

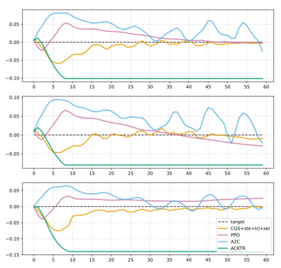


(b) Pressure C2H4-2

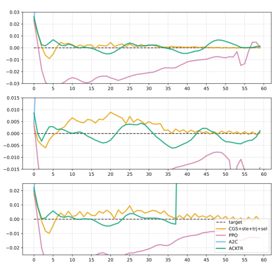


(c) Pressure C2H4-4

Fig. 1. Performance for different disturbance scenarios on the VAM plant. Results are averaged over 5 runs (±std).



(a) Pressure AcOH



(b) Pressure C2H4-4

Fig. 2. Examples of learned policies for several disturbances with different levels of intensity. A straight line usually indicates a failure in equipment, meaning that the policy is no longer able to control the plant.

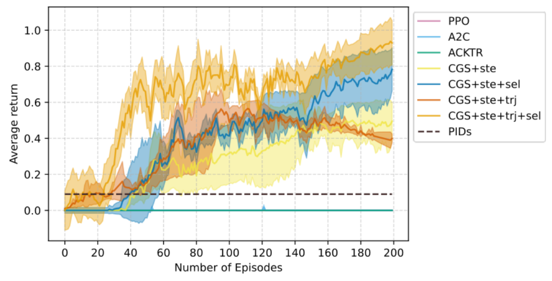


Fig. 3. Average return on the VAM plant under day and night disturbance. Results are averaged over 5 runs.

In this scenario, day and night cycles may result in non-steady conditions caused by a drop in internal temperatures. Figure 3 plots the learning curves of all the models. We can observe that our strategy helps to greatly improve convergence speed and performance. We can further observe that the existing PIDs cannot cope with disturbances. On the other hand, leveraging controller guidance allows our agent to quickly experience first few successful trajectories. After a few training episodes, self-supervision provides enough supervision for the learner to discover alternative strategies to further correct disturbances.

3.6 Ablation Study

Size of the Episodic Memory We now report evaluations showing the effect of increased episodic memory size. Table 1 demonstrates that agents trained with memory size larger than 10 obtain higher mean returns after similar numbers of updates. However, we can also observe that a too large value (i.e. K >50) tends to hurt the performance. One reason is that a large memory induces a higher chance of keeping in memory outdated suboptimal trajectories.

Table 1. Return (± std) for different values of the memory size (CGS+$ste + sel$). Results are averaged over 5 random seeds.

	Average return ± std			
Memory Size	Pressure AcOH	Pressure C2H4-2	Pressure C2H4-4	Day/Night
10	0.77 ± 0.11	0.93 ± 0.04	0.89 ± 0.02	0.75 ± 0.03
20	0.79 ± 0.10	0.95 ± 0.04	0.92 ± 0.03	0.77 ± 0.02
30	0.80 ± 0.15	0.95 ± 0.05	0.91 ± 0.03	0.76 ± 0.04
50	0.77 ± 0.13	0.92 ± 0.04	0.88 ± 0.05	0.76 ± 0.06
100	0.71 ± 0.11	0.89 ± 0.06	0.86 ± 0.04	0.69 ± 0.06

Table 2. Return (± std) for different types of imperfect guidance in the pressure C2H4-4 scenario, where ϱ is the probability of adding noise. Results are averaged across 5 seeds.

	Average return ± std			
Method	$\varrho = 0.0$	$\varrho = 0.05$	$\varrho = 0.1$	$\varrho = 0.2$
CGS+ste (act)	0.82 ± 0.05	0.68 ± 0.11	0.55 ± 0.06	0.48 ± 0.12
CGS+ste (no-act)	0.91 ± 0.03	0.87 ± 0.07	0.84 ± 0.10	0.71 ± 0.09
CGS+$ste + trj$ (act)	0.65 ± 0.08	0.57 ± 0.08	0.38 ± 0.18	0.25 ± 0.21
CGS+$ste + trj$ (no act)	0.90 ± 0.11	0.84 ± 0.09	0.73 ± 0.12	0.70 ± 0.10
CGS+$ste + sel$	0.92 ± 0.03	–	–	–
CGS+$ste + trj + sel$	0.98 ± 0.02	–	–	–
PPO	0.40 ± 0.07	–	–	–
A2C	0.01 ± 0.01	–	–	–
ACKTR	0.17 ± 0.03	–	–	–
PIDs	0.52	–	–	–

Generally, the size does not require to be fine-tuned for each task (i.e. $20 \leq K \leq 50$) since the agent maintains acceptable performance.

Effect of Imperfect Guidance In this section, we aim to investigate the effect of imperfect guidance on the agent's performance. We compare our method (CGS no-act) to CGS where we replace s^* by the expert actions a (CGS-act). In other words, given tuples (a, s), the discriminators distinguish between the provided guidance and the agent's growing experience, where s is the current state and a is the next action. The goal is to evaluate whether leveraging state trajectories is more robust to imperfect/noisy guidance than learning from state-action trajectories. Although trj guidance is by nature suboptimal, we sought to increase the difficulty of the task. Thus, we simulate imperfect guidance by adding normal noise $\mathcal{N}(0, \sigma^2)$ to the current state (CGS no-act) s or the expert actions (CGS act), with a probability $\varrho \in \{0.0, 0.05, 0.1, 0.2\}$ and $\sigma = 0.03$. For this experiment, we use PPO+ste and PPO+$ste + trj$ since self-supervision is not directly affected by noisy guidance. We report in Table 2 the performance of our framework. We observe that CGS can still achieve acceptable performance. Even though the proposed method performs slightly worse in the imperfect

Table 3. Return (± std) for different amounts of controller guidance. Results are averaged over 5 random seeds.

	Average return ± std		
# Steady States / Trajectories	Pressure AcOH	Pressure C2H4-4	Day/Night
50 / 5	0.89 ± 0.06	0.87 ± 0.07	0.53 ± 0.05
100 / 5	0.89 ± 0.05	0.88 ± 0.06	0.52 ± 0.06
500 / 5	0.90 ± 0.05	0.87 ± 0.08	0.52 ± 0.05
50 / 15	0.91 ± 0.05	0.88 ± 0.08	0.55 ± 0.04
100 / 15	0.93 ± 0.04	0.90 ± 0.11	0.61 ± 0.04
500 / 15	0.92 ± 0.05	0.90 ± 0.09	0.62 ± 0.03
50 / 50	0.92 ± 0.07	0.90 ± 0.08	0.56 ± 0.02
100 / 50	0.92 ± 0.06	0.93 ± 0.09	0.62 ± 0.04
500 / 50	0.93 ± 0.05	0.92 ± 0.08	0.64 ± 0.03

setting, it still notably improves performance compared to the agents that learn from actions. This trend continues when noise is artificially added to the expert actions or recently observed states (i.e. $\varrho > 0$). Overall, empirical results suggest that: 1) our method is reasonably robust to suboptimal guidance, 2) an agent trained from state guidance is less prone to overfitting suboptimal behaviors than an agent relying on state-action guidance.

Amount of Controller Guidance Finally, to quantify the impact of controller guidance on learning and performance, we evaluate our architecture (CGS+ste + trj) with different amounts of controller guidance. Table 3 reports the mean episode-returns obtained on different scenarios. We notice that although the number of steady states significantly differs, the difference in learning effect can be negligible. This can happen because the agent is able to leverage a small amount of steady states in order to identify task-relevant regions. We can further observe that our method is able to operate in the low-data regime — a few imperfect trajectories generated by PIDs provide enough supervision to enhance the performance of agents.

4. CONCLUSION

In this paper, we presented CGS, a sample-efficient RL framework for process control, which is capable of controlling a chemical plant and correcting disturbances. The proposed method relies on controller guidance to improve the agent's performance, particularly at the onset of its training, and self-supervision to guide the agent throughout the training process. We further propose a strategy to prioritize sampling of the agent's successful experience, and adjust the exploration bonus according to the discriminators' confidence. Experimental results on a VAM plant show that CGS can greatly benefit in sample efficiency and could help to expand the possible applications of RL to real-world process control. Remarkably, our approach is able to leverage guidance from suboptimal controllers and alleviates the need to access to the expert actions. We further demonstrate that CGS significantly outperforms baselines in terms of average return and sample efficiency.

REFERENCES

Bellemare, M.G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47, 253–279.

Cheng, L., Subrahmanian, E., and Westerberg, A.W. (2004). A comparison of optimal control and stochastic programming from a formulation and computation perspective. *Computers & Chemical Engineering*, 29(1), 149–164.

Cui, Y., Zhu, L., Fujisaki, M., Kanokogi, H., and Matsubara, T. (2018). Factorial kernel dynamic policy programming for vinyl acetate monomer plant model control. In *IEEE International Conference on Automation Science and Engineering*, 304–309.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 1050–1059.

Jia, R., Jin, M., Sun, K., Hong, T., and Spanos, C. (2019). Advanced building control via deep reinforcement learning. *Energy Procedia*, 158, 6158–6163.

Kamthe, S. and Deisenroth, M. (2018). Data-efficient reinforcement learning with probabilistic model predictive control. In *International conference on artificial intelligence and statistics*, 1701–1710.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kubosawa, S., Onishi, T., and Tsuruoka, Y. (2019). Synthesizing chemical plant operation procedures using knowledge, dynamic simulation and deep reinforcement learning. *ArXiv*, abs/1903.02183.

Kubosawa, S., Onishi, T., and Tsuruoka, Y. (2021). Computing operation procedures for chemical plants using whole-plant simulation models. *Control Engineering Practice*, 114, 104878.

Li, Y., Wen, Y., Tao, D., and Guan, K. (2020). Transforming cooling optimization for green data center via deep reinforcement learning. *IEEE Transactions on Cybernetics*, 50(5), 2002–2013.

Machida, Y., Ootakara, S., Seki, H., Hashimoto, Y., Kano, M., Miyake, Y., Anzai, N., Sawai, M., Katsuno, T., and Omata, T. (2016). Vinyl acetate monomer (vam) plant model: A new benchmark problem for control and operation study. 49(7), 533–538. IFAC Symposium on Dynamics and Control of Process Systems Including Biosystems.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Conference on machine learning*, 1928–1937.

Qin, Y., Zhang, W., Shi, J., and Liu, J. (2018). Improve pid controller through reinforcement learning. In *IEEE Guidance, Navigation and Control Conference*, 1–6.

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

Wu, Y., Mansimov, E., Grosse, R.B., Liao, S., and Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *Proceedings of Advances in neural information processing systems*, 30, 5279–5288.

Zhang, H., Cisse, M., Dauphin, Y.N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *international Conference on Learning Representations*.

Zhu, L., Cui, Y., Takami, G., Kanokogi, H., and Matsubara, T. (2020). Scalable reinforcement learning for plant-wide control of vinyl acetate monomer process. *Control Engineering Practice*, 97, 104331.