Hybrid Deep Reinforcement Learning Agent for Online Scheduling and Control for Chemical Batch Plants

Daniel Rangel-Martinez*, Luis Ricardez-Sandoval**

*Department of Chemical Engineering, University of Waterloo, Canada (e-mail: drangelm@uwaterloo.ca)
**Department of Chemical Engineering, University of Waterloo, Canada (e-mail: laricard@uwaterloo.ca)

Abstract: This study presents a framework for the implementation of a Deep Reinforcement Learning (DRL) agent for optimal scheduling and control integration on flow-shop batch plants with input variability. The agent is designed to take multiple decisions at every time interval which allows for the integration of scheduling and control. A hybrid agent with multiple decision outputs is used to perform online scheduling and control. To account for the short-term history of the process, the agent approaches the optimization problem as a Partially Observable Markov Decision Process (POMDP). The agent makes use of a set of Long Short-Term Memory cells (LSTM) to correlate sequential states from the environment to be aware of its evolution when taking decisions. To demonstrate the advantages and limitations of the hybrid agent, the method is implemented on a batch plant under variability in the inputs. Results showed that the agent's policy reacted to the fluctuations in concentration from raw materials. To validate the proposed method, a comparison with an agent trained on an environment with fixed inputs was performed to demonstrate the adaptive behavior of the agent developed with the presented framework.

Keywords: Reinforcement learning, Integration of scheduling and control

1. INTRODUCTION

Reinforcement Learning (RL) is a learning method that assumes that the decision-making process in an environment follows the Markovian property. This implies the existence and availability of a transition model that defines the probabilities of moving from one state s to another s'. The Markovian property implies that the only information required to know the next state s' of the environment after an action is executed is that provided by the current state s. The application of DRL methods for scheduling and control applications has been widely approached as Markov Decision Processes (MDP). Nevertheless, real world scenarios present multiple situations where the information is spread out in a sequence of states rather than in the immediate present state. Then, it is assumed that a Partially Observable Markov Decision Process (POMDP) results in a more adequate approach for the modelling of these kind of environments.

DRL has found wide acceptance as an approach for solving scheduling problems since in most of the cases the action space consists of a set of discrete actions. For instance, the job to be initialized, the dispatching rule to be adopted, or the machine(s) to activate. In control applications, DRL has also found utility as policies can be trained to respond to changes in the state of the process. Nevertheless, the usual DRL approach is limited to one decision at every time interval and does not allow the use of compound actions that involve multiple decisions at the same time. Then, DRL cannot be applied in problems involving simultaneous scheduling and control decisions. To the authors' knowledge, this integration has not been attempted in the literature.

DRL has been previously applied to solve problems involving optimal process integration. (Mendiola-Rodriguez and Ricardez-Sandoval, 2023) proposed an integration scheme for design and control with DRL. The agent trained with a Deep Deterministic Policy Gradient method considers the first decision as the process design and the subsequent decisions as the control actions. (Sachio et al., 2022) made an integration of design and control using DRL. In that work, a controller is built using DRL and then is integrated into a bi-level optimization problem where the other level corresponds to the design task. Single-task applications involving for example design, scheduling, and control decisions are common. DRL agents have also been applied as reactive controllers that adapts to changes in the process. (Mowbray et al., 2021) presented a framework where a controller is generated using inverse RL and then trained with DRL, showing potential for optimal process control. (Bloor et al., 2024) combined components from Proportional-Integral-Derivative control with DRL, resulting in a controller capable to generalize trajectories outside of the distribution set during the training. Scheduling optimization problems have also been approached with DRL, especially in job and flow shops for multiple fields. (Hubbs et al., 2020) proposed a scheduler agent for a multi-product reactor, capable of handling uncertainty in the process. (Waschneck et al., 2018) used a multi-agent system for reducing the waiting times in a job-shop. (Rangel-Martinez and Ricardez-Sandoval, 2024) proposed an agent that approached the scheduling problem as a POMDP to handle partial information in the states sent from the environment.

The previous works demonstrate that DRL policies have been applied effectively for process integration, and for single-task

applications. In most of these works, the approach consists in handling the optimization problem as a Markov Decision Process (MDP). Moreover, those studies showed as a common feature that the action space only involves a single action. This limitation has been overcome with the use of multi-agents. This work explores the integration of the decisions of multiple agents into one that gathers all the information needed for the entire system of decisions, aiming for an efficient use of the information. Note that this approach assumes that tasks taking place at multiple scales are correlated.

In this study, a framework to address the integration of scheduling and control with a DRL agent is presented. A hybrid agent that can output the scheduling and control decisions is used for the integration. The sequential decision problem is assumed to be a POMDP, which allows the inclusion of input information into the agent from past events that took place in the system. The POMDP assumes a lack of information given to the agent, which is compensated with historical information. This study applies an approach for POMDPs in the scheduling-and-control problem, which to the author's knowledge, has not been reported in the literature. The LSTMs layers are used because they serve as a correlation module for sequences of events in the process and is a key feature in this work. To illustrate the characteristics of this approach, an agent is trained with this scheme to take online decisions of the scheduling and control of a chemical flow-shop batch plant subject to variability in the process inputs, i.e., inlet material flows. The concentrations of these flows are assumed to follow a stochastic behaviour. The scheduling decision defines the task that should be initialized while the control actions aim to accommodate the stochastic perturbations in the inputs while aiming to meet the task's production targets. This study is organized as follows: section 2 presents the problem statement; section 3 shows the proposed methodology; section 4 shows a case study and results; conclusions and future work are presented at the end.

2. PROBLEM STATEMENT

This section presents the scheduling and control problem that can be addressed with the present approach. The problem definition was adapted from a previous work that addressed the integration of scheduling and control for chemical batch plants under stochastic uncertainty (Rodríguez Vera and Ricardez-Sandoval, 2022). Our work does not approach the problem under uncertainty per se, but instead assumes that there are stochastic parameters (ψ_{var}) defined by known Probability Density Functions (PDF). The value of these parameters is only known when a unit related to these parameters is turned on, i.e., a task is assigned to this unit. Their values remain constant during the operation of that particular task. In the case of units involving process dynamics, values for these parameters are known at the beginning of the operation and its value remain constant until the end of that specific operation. Hence, appropriate control actions must be determined to accommodate for these changes and be able to meet the unit operation targets. This situation demands an online decision-making process for scheduling and unit operation control that needs to adapt to the outcomes of such stochastic parameters. For the systems that can be considered with the method presented in this work, consider the following:

- A flow-shop plant that is composed by N_k set of tasks, with N_j set of equipment.
- A set of chemical processes described by the mechanistic dynamic model f for N_p states of the system, and expressions h that contain the set N_q of constraints of the system.
- A set Ψ of fixed model parameters ψ_{fix} and of stochastic parameters ψ_{var} described by a PDF known *a priori*.
- A set C that considers the cost information of utility services, raw materials, products, and by-products.
- A finite time horizon H which starts at t_s and ends at t_f.
- A finite number of equal-length time intervals t that belong to the set N_t and are used for the discretization of the scheduling horizon H. At every time interval t, scheduling and control decisions are taken.
- A set T of processing times indicating the length of time $\tau_{k,j}$ that the units in N_j takes to complete the tasks in N_k.
- An economic function G that considers product profits, operational times, costs related to utility services, penalties incurred during the operation, and other related costs.

The optimization formulation for the batch plant described above is stated as problem P1. This integrated optimization problem aims for an optimal schedule plan with dynamic control profiles that maximizes the profits of the process.

$$\max_{u_{k,j(t)},\, s_{k,j,t}} G(x_{k,j}(t), u_{k,j}(t), \varphi, \tau_{k,j}, s_{k,j,t}, c) \qquad \text{(P1)}$$

s.t. $f_p\left(x_{k,j}(t), \dot{x}_{k,j}(t), u_{k,j}(t), \varphi, \tau_{k,j}, s_{k,j,t}, t\right) = 0$

$\forall\, t, p \in N_p, t \in N_t, q \in N_q, k \in N_k, j \in N_j$

$h_q(x_{k,j}(t), \dot{x}_{k,j}(t), u_{k,j}(t), \varphi, \tau_{k,j}, s_{k,j,t}, t) \leq 0$

$\forall\, t, t \in N_t, q \in N_q, k \in N_k, j \in N_j$

$\tau_{k,j} \in \tau \qquad \forall\, k \in N_k, j \in N_j$

$s_{k,j,t} \in \{0,1\} \quad \forall\, k \in N_k, j \in N_j, t \in N_t$

$x \in X \subseteq \mathbb{R}^{N_x \times N_j \times N_k}$

$u \in U \subseteq \mathbb{R}^{N_u \times N_j \times N_k}$

$c \in C \subseteq \mathbb{R}^{N_c}$

$\psi_{fix}, \psi_{var} \in \Psi \subseteq \mathbb{R}^{N_\psi}$

$\psi_{var} \sim f_{\psi_{var}}(\psi)$

$\tau \in \mathrm{T} \subseteq \mathbb{R}^{N_\tau}$

$t \in [t_s, t_f], \; H = t_f - t_s$

where f_p is the p^{th} differential-algebraic equation of the model and h_q is the q^{th} model constraint. $x_{k,j}(t)$ is a state variable from the system for task k taking place at unit j and $\dot{x}_{k,j}(t)$ is the derivative of that variable. $u_{k,j}(t)$ is a control decision variable involved in task k at unit j. The control actions $u_{k,j}(t)$ that can be taken at each time interval t by the time-dependent unit operations j to maintain the process on target are limited to a set of discretized actions included in the set U. $f_{\psi_{var}}(\psi)$ is the PDF used to describe the likelihood of a realization of ψ_{var}.

$s_{k,j,e}$ is a binary decision variable that specifies the scheduling process for the batch plant. This variable is defined for every task k at unit j during the set of time intervals t and tells if such task and unit are occupied or not.

Problem P1 can be defined as a stochastic Mixed Integer Dynamic Optimization (MIDO) problem, which may become challenging to solve since the realizations of ψ_{var} are not known *a priori*. Approaches consisting of the decomposition of the problem into MILP or MINLP can be used to reduce the problem's complexity but they present their own challenges for online implementation due to intensive calculations (Andrés-Martínez and Ricardez-Sandoval, 2022). Simplifications can be made with a trade-off between time response and quality of the solution. The DRL method presented in this work is used to train the agent for stochastic realizations of the input parameters and then, once trained, it can be implemented online and provide immediate (online) responses to the integrated process thus providing a fast reliable solution to this problem.

3. METHODOLOGY

In this section we present the methodology to design the DRL agent that can provide online solutions to the flow-shop scheduling and control problem described in P1. A Proximal Policy Optimization (PPO) method is used to train the agent as it is a stable and general-purpose algorithm. The following key components in the DRL framework are explained in this section: 1) the action space for the scheduling and control tasks; 2) the input sequence vector for the agent, which contains relevant information used to take a decision; 3) the reward function, 4) the time-dependent stochastic parameters, and 5) the architecture of the hybrid agent.

3.1. Action space

The number of decisions that an agent will take at each time interval t in the horizon H include both scheduling and control decisions. It is assumed here that both tasks need at least one action to be executed in the process but this can be extended to multiple actions that correspond to each task. The set of action spaces is shown in Eq. (1), where all the needed actions for scheduling and control tasks are included. Note that all the action spaces in this set are discretized; it is also possible to set continuous spaces but this is beyond the scope of this work.

$$Actions = \{a_1, a_2, \dots, a_n\} \tag{1}$$

The scheduling action space can specify the task and unit to be initialized at each time interval. Since an action is delivered at every time interval t, the action space also includes an idle action that do not start any task. Other discrete actions related to the scheduling task can be added to the agent, for instance, the capacity at which a task should be started (i.e., 100%, 20%, etc.). Similarly, the number of control actions is given by the number of control decision variables included in the set U defined above. Since the control actions are discretized, the control profiles generated by the agent for each control variable u_{kj} will be similar to step-like functions applied to each time interval t.

During implementation, at a specific time interval, the environment sends the sequence of observations from the process to the agent. Then, for every action space in $Actions$, a categorical distribution p_{a_n} is provided (Eq. 2). The largest probability at each action space distribution defines the action executed in the environment. During training, the distributions are used to sample actions and learn from their execution. Note that at all time intervals, the agent generates categorical distributions for all the action spaces described in the set $Actions$. Naturally, not all of them are used at all time intervals, for instance a control action will be only used when the corresponding unit is active. Actions that are not required at a certain time interval are ignored by the environment.

$$p_{a_n} = (p_1, p_2, \dots, p_m) \text{ where } \sum_{m \in M} p_m = 1 \tag{2}$$

3.2. Observation Vector

A vector with the information from the environment called observation vector (o_t) is generated at every time interval and sent to the agent to produce the next set of actions. The vector o_t defined in Eq. (3) contains the features that have a measurable effect in the scheduling and control tasks. In this work, o_t gathers the following information: i) the occupation of the available units j; ii) processing times of the units that are in use; iii) information relevant to the control problem, e.g., concentrations, flow rates, and temperatures; and iv) the current time interval. This information is retrieved at every time interval from the environment and is also normalized to stabilize the learning process. Note that the representation of these variables in o_t depends on the user's preferences. For instance, the use of probabilities, constant values, one-hot encodings, or normalized values can be used to pre-process these features and then add them to the observation vector o_t.

$$o_t = \left[x_{k,j}(t), s_{k,j,e}, \tau_{k,j}, t\right] \tag{3}$$

Although this state representation provides full observation to the agent, the use of the POMDP approach is justified to provide the agent with additional information from past events. The LSTMs correlate the information in a temporal context which helps are key to consider interactions between scheduling and control.

3.3. Reward function

A reward shaping method was used in this work to decompose the total reward into m sub rewards r, i.e.,

$$Reward = \sum_m r_m \tag{4}$$

The subset of rewards r_m for the scheduling actions is based primarily in allocation constraints and the correct order of task initialization in the flow-shop. This ensures that the conditions for initiating a task are met, for instance, availability of material or availability of units. Moreover, penalties for machines staying idle during a process should be set to ensure the reduction of the makespan. Furthermore, the reward should be increased as the process reaches the end, i.e., to provide larger rewards to the agent for initializing tasks that are closer to the end.

Regarding the control decisions, the agent should account for the economic incentives associated with these tasks. For instance, providing larger rewards to utility services that are less expensive. Penalties should be provided for constraint violations that may occur during transient operation, e.g., surpassing a safety limit or not reaching the expected product specifications at the end of the task. It is assumed that any task k that comprises a dynamic system will need at least one time interval to be completed. Hence, a sequence of control decisions, each implemented at each time interval t, are required for the duration of that task. Then, the reward for that sequence is provided at the end of task k. Here, the POMDP approach becomes useful as the sequence of actions (i.e., the control profile) is receiving a reward, rather than only the present action. Thus, the use of LSTMs results convenient for handling the delayed rewards in the environment. After being sent to the environment, the actions from the set $Actions$ that were implemented (recall that not all of them might be used) are evaluated with the system of rewards. When all the rewards are assigned to the scheduling and control decisions, they are added up and sent as a unique (scalar) reward to the agent.

3.4. Stochastic Parameters

The environment is a representation of the process where the agent can learn by trial and error on how to perform a task. The limitations of the real process (i.e., constraints) and the description of fixed and stochastic parameters (i.e., ψ_{fix}, ψ_{var}) should be added into this simulation. During each iteration of the training, the stochastic parameters are set to present the agent a specific realization of this parameter. The agent gathers experience from these realizations and generalizes a policy that can choose the best action for a given realization of ψ_{var}.

In the environment, the mathematical description of the stochastic parameters should be incorporated through a PDF, e.g., a Gaussian or Uniform distribution. These values should also be incorporated in the observation that are sent to the agent as they are relevant for the next decision. Note that the increase in time-varying parameters has a direct impact on the length of the training. This is because the agent needs to explore multiple realizations to learn its policy. Thus, the exploration space grows as more stochastic parameters are considered.

3.5. Hybrid Agent

The architecture considered in this study for the hybrid agent was adapted from (Fan et al., 2019). This architecture allows the agent to generate n outputs for the scheduling and control decisions defined in the set $Actions$. The n outputs from the agent are categorical distributions. Each distribution gives the probability to each possible outcome of that particular action (see Eq. (2)). The hybrid agent uses a correlational module built with W layers of LSTMs to gain insights of the sequence of observations o_t. The sequence is also called observation window O_t as shown in Eq. (5). It starts in the time interval t-λ+1, and ends in the time interval t, where λ denotes the number of observations. This allows to register all the steps that a task takes to complete. That is, the agent can access the sequences that contain the evolution of the tasks from start to end, which is not possible under the MDP approach.

$$O_t = [o_{t-\lambda+1}, \dots, o_{t-2}, o_{t-1}, o_t] \quad (5)$$

The general architecture of the neural network is depicted in Fig. 1. The observation window O_t is an input to the correlational module with W layers. Then, the outputs are passed through a set of Y linear layers. The final output of this section is input into n separated sets with Z linear layers. The output of each set is passed through a SoftMax activation function to output each categorical distribution.

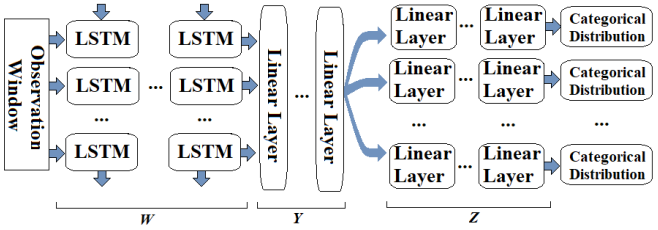

Fig. 1. Hybrid agent with correlational module

With this architecture, an agent that can coordinate both scheduling and control tasks can be trained. The incorporation of LSTMs into the agent allows to correlate the observations. This feature provides the agent with a wider visualization of the development of the processes. Limitations of networks with recurrence include the computational effort that they require as they evaluate the sequence element by element, i.e., not in a parallel fashion.

4. CASE STUDY AND DISCUSISON OF RESUTLS

The methodology is applied to perform the simultaneous scheduling and control of a flow-shop batch plant from the literature (Rodríguez Vera and Ricardez-Sandoval, 2022). Fig. 2 shows the process composed by four tasks: 1) a set of chemical reactions (RI), 2) a filtration process (FI), 3) a set of reactions (RII), and 4) a separation process (SI), i.e., $N_k = \{RI, FI, RII, SI\}$. The time horizon H is set to 15 hours and is divided in even time intervals of $t = 0.5$ h; hence, there are 30 time intervals at which the agent can take decisions.

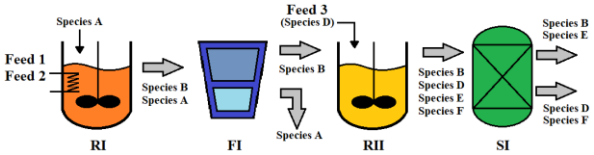

Fig. 2. Bach Plant model

The process begins in RI, where a non-isothermal batch reactor is used to transform substance A into an intermediate product B. The temperature of the reactor is controlled through the flow rates of Feed 1 and Feed 2, which correspond to a hot and a cold stream of water, respectively. The resulting mixture of products A and B from RI is then filtered in task FI. The intermediate species B is then passed to a semi-batch reactor where species D is added through Feed 3. The flow-rate of Feed 3 is controlled during the process to regulate the production of products E and F. The mixture of species B, D, E, and F (where E is the desired product) is separated in task SI, as shown in Fig. 2. Processes FI and SI are assumed to

achieve perfect separation and are stationary processes. The four tasks have a fixed operation time of 2h. RI and RII are assumed to be dynamic processes and are expected to finish at their corresponding processing time. Each task has one unit, i.e., $N_j = \{Reactor1, Filter, Reactor2, Separator\}$. After completion, each unit will store the outgoing material until the following task is initialized. While the material is stored in that unit, a subsequent task for this unit cannot be initialized. The mechanistic models of the dynamics systems, the process constraints, model parameters and initial conditions for this case study can be found in (Rodríguez Vera and Ricardez-Sandoval, 2022). For the present case study, the initial concentrations of species A entering RI and D in Feed 3 vary according to a uniform distribution. For the former, the distribution is bounded between 1.01 and 1.15 $kmol/m^3$, while the latter is bounded between 0.80 and 0.89 $kmol/m^3$.

The economic function is described in Eq. (6) and aims to maximize the revenue from the production of product E while considering the minimization of costs related to the utility services from the three controlled flow rates corresponding to Feeds 1, 2, and 3 as described above.

$$G = cost_E * Storage_{SI} + cost_{utility\ services} \qquad (6)$$

where $cost_E * Storage_{SI}$ denotes the revenue obtained from the product that is output from task SI. The costs of the product E and utility services are specified in (Rodríguez Vera and Ricardez-Sandoval, 2022). The objective of the DRL agent is to design a policy that can produce an online schedule and a control strategy considering the stochastic variability in the concentrations in the raw materials, i.e., species A and D.

At every time interval t, the agent will provide four ($n = 4$) decisions: 1 for the scheduling task, and 3 for the control task. For the scheduling task, the agent will decide which task to activate; hence, this decision has 5 possible outcomes (i.e., 4 tasks and 1 idle action). The control task manipulates the flows of Feeds 1, 2, and 3 (see Fig. 2) while the task RI and RII are in operation. For every time interval cloistered in the operation time of these tasks, the agent sends a control decision to the environment. Then, the states of tasks RI or RII are sent back to the agent where another control decision is made for the next time interval. The control actions for each flowrate are discretized as shown in Table 1.

Table 1. Flow-rates for the control tasks.

	Low m^3/h	Medium m^3/h	High m^3/h
Feed 1	3.75	7.5	16.0
Feed 2	3.75	7.5	16.0
Feed 3	0.08	0.9	1.7

Since the processing times ($\tau_{k,j} = 2.0h$) are the same for every task and since a time interval is of length 0.5h, the control profile for each process has four stages. If the reaction task do not reach the desired concentration at the end of the processing time, or if one of the constraints is violated during operation, then the product will be wasted away from the process. For the subsequent actions, the agent should consider this interruption and schedule tasks accordingly. The system of rewards shown in Table 2 was used to guide the learning of the agent to aim for the maximization of profit.

Table 2. Rewards assigned to the actions.

	Description of the rewards
Scheduling actions	Reward +5 if the unit for that task is not busy when initializing a task and -10 if the unit is busy. If the task is completed to the end, then give a reward of +10, if the process is interrupted due to constraint violation, give a penalty of -5. For every time interval that tasks RI, FI and RII keep the product stored, a penalty of -0.5 is addressed.
Control actions	Feeds 1 and 2: If Task 1 (RI) is running, provide a reward of +3 if the high profile is chosen, +6 for the medium profile, and +9 for the low profile. This is done for each hot and cold flows. The total reward is the addition of both.
	Feed 3: If Task 3 (RII) is running, provide a reward of +6 if the high profile is chosen and applied, +12 if it is medium profile, and +18 if it is the low profile.

The architecture of the agent is based on Fig. 1. The length of the observation window is set to 4 (i.e., $\lambda = 4$) since each unit is assumed to require 4 time intervals to complete a task. The relational module has three layers of LSTMs ($W = 3$), followed by two layers, i.e., $Y = 3$. Then, the output of this set is the input of each of the sets of layers that represent each action ($n = 4$); each one with three layers ($Z = 3$). Hidden layers use tanh activation functions while a SoftMax activation function is used to generate categorical distributions. The DRL method was developed in Python 3.11.3 and PyTorch version 2.1.0. The training was performed using Adam's optimizer; with the learning rate gradually decreased through an annealing technique, starting at 1e-4 and finishing at 1e-6. To refine the exploration as the training progressed, the smoothness of the categorical distributions was sharpened using temperature annealing, starting at 1 and ending at 0.001. A test with 1000 multiple realizations in ψ_{var} was performed with the trained policy π_v. Fig. 3 shows the number of times (cycles) that the process in Fig. 2 was executed in the horizon H. In almost 90% of the realizations, the agent could complete two to three cycles. The agent could not set three cycles in all the instances due to time limitations or a task failure, as discussed below. To compare the performance of the policy obtained by this method (π_v), a second policy (π_f) was trained on the same environment but under no variability in the inlet concentrations, i.e., ψ_{var} were fixed to 1.08 $kmol/m^3$ and 0.845 $kmol/m^3$ for species A and D, respectively. The resulting policy π_f was then tested under 1000 stochastic realizations in ψ_{var}. As shown in Fig. 3, the strategy learned by π_f could handle many of the scenarios on which π_v was trained. This was expected because the control profiles that were learnt by π_f may be adequate to accommodate most of the stochastic scenarios; nevertheless, they might not align with the objective function. In half of the processes, π_f could not produce a full cycle, i.e., no production; only in 13.3% of

the instances it was able to complete three cycles as policy π_v. Fig. 4 shows a schedule built with π_v in which the first cycle was interrupted due to the cancellation of Task RII. After this, the agent chooses the initialization of Task RI instead of Task S, which aims to produce more product E. This decision corresponds to the reaction of the agent to the suspension of the first cycle. It was observed that the agent left blank spaces in between tasks, presumably due to a lack of sensitivity to the observation window. Also, some tasks that were already in operation were selected to be initialized by the agent; these actions result in an infeasible operation. This problem can be addressed by increasing the penalties associated with violation of allocation constraints in exchange for a more conservative agent. A heuristic that cancels the initialization of infeasible actions was added to the environment, i.e., new task that must be started on a unit that is already in operation are cancelled.

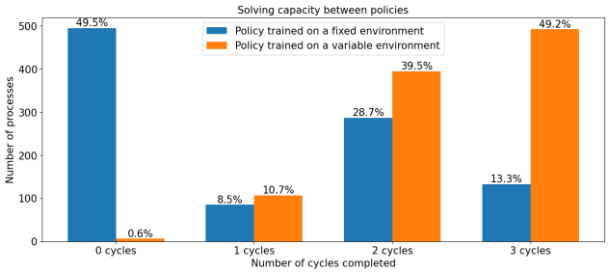

Fig. 3. Comparison between policies

In most of the cases, the control profiles found by π_v completed reactions RI and RII without any constraint violations (not shown for brevity). Results from Fig. 3 confirm the effectiveness of the control actions chosen by the hybrid agent since it completed multiple cycles under stochastic variability in the inputs parameters affecting RI and RII.

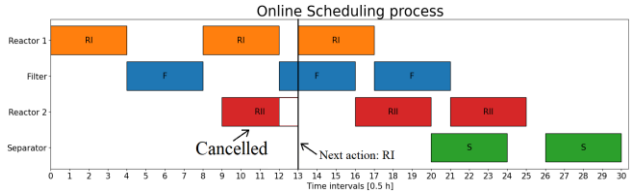

Fig. 4. Schedule with three cycles initialized

5. CONCLUSIONS

In this work a methodology for developing a DRL to address the integration of scheduling and control in flow-shop batch plants is presented. The DRL method allows to build online schedules and control profiles according to the conditions of the plant. A POMDP approach is used to retrieve history from the plant and consider the tasks evolution in the decision-making process. To handle multiple decisions taken during transient operation, a hybrid agent is used to output more than one action at every time interval. To validate the method, a batch plant was considered for optimal scheduling and control. The agent designed an effective online schedule and control policy that can accommodate stochastic realizations in the input parameters. The application of this method for partially observable environments is part of the future work. Consideration of discrete and continuous actions for scheduling decisions has been recently reported (Rangel-Martinez and Ricardez-Sandoval, 2025); extending this approach for integration of scheduling and control is also recommended for future work.

6. ACKNOWLEDGMENT

Support provided by CONAHCYT, Mexico is acknowledged.

7. REFERENCES

Andrés-Martínez, O., Ricardez-Sandoval, L.A., 2022. Integration of planning, scheduling, and control: A review and new perspectives. The Canadian Journal of Chemical Engineering 100, 2057–2070. https://doi.org/10.1002/cjce.24501

Bloor, M., Ahmed, A., Kotecha, N., Mercangöz, M., Tsay, C., Chanona, E.A.D.R., 2024. Control-Informed Reinforcement Learning for Chemical Processes.

Fan, Z., Su, R., Zhang, W., Yu, Y., 2019. Hybrid Actor-Critic Reinforcement Learning in Parameterized Action Space.

Hubbs, C.D., Li, C., Sahinidis, N.V., Grossmann, I.E., Wassick, J.M., 2020. A deep reinforcement learning approach for chemical production scheduling. Computers & Chemical Engineering 141, 106982. https://doi.org/10.1016/j.compchemeng.2020.106982

Mendiola-Rodriguez, T.A., Ricardez-Sandoval, L.A., 2023. Integration of design and control for renewable energy systems with an application to anaerobic digestion: A deep deterministic policy gradient framework. Energy 274. https://doi.org/10.1016/j.energy.2023.127212

Mowbray, M., Smith, R., Del Rio-Chanona, E.A., Zhang, D., 2021. Using process data to generate an optimal control policy via apprenticeship and reinforcement learning. AIChE Journal 67, e17306. https://doi.org/10.1002/aic.17306

Rangel-Martinez, D., Ricardez-Sandoval, L.A., 2024. A recurrent reinforcement learning strategy for optimal scheduling of partially observable job-shop and flow-shop batch chemical plants under uncertainty. Computers & Chemical Engineering 188, 108748. https://doi.org/10.1016/j.compchemeng.2024.108748

Rangel-Martinez, D., Ricardez-Sandoval, L.A., 2025. A Recurrent Reinforcement Learning Strategy with a Parameterized Agent for Online Scheduling of a State Task Network Under Uncertainty. Industrial & Chemical Engineering Research. https://doi.org/10.1021/acs.iecr.4c04900

Rodríguez Vera, H.U., Ricardez-Sandoval, L.A., 2022. Integration of Scheduling and Control for Chemical Batch Plants under Stochastic Uncertainty: A Back-Off Approach. Ind. Eng. Chem. Res. 61, 4363–4378. https://doi.org/10.1021/acs.iecr.1c04386

Sachio, S., Mowbray, M., Papathanasiou, M.M., del Rio-Chanona, E.A., Petsagkourakis, P., 2022. Integrating process design and control using reinforcement learning. Chemical Engineering Research and Design 183, 160–169. https://doi.org/10.1016/j.cherd.2021.10.032

Waschneck, B., Reichstaller, A., Belzner, L., Altenmüller, T., Bauernhansl, T., Knapp, A., Kyek, A., 2018. Optimization of global production scheduling with deep reinforcement learning. Procedia CIRP, 51st CIRP Conference on Manufacturing Systems 72, 1264–1269. https://doi.org/10.1016/j.procir.2018.03.212