An Industrial Use-Case for Reinforcement Learning: Optimizing a Production Planning in Stochastic Conditions

Paul Berhaut*, Ikhlass Yaya-Oyé, and Axelle Albot

Air Liquide SA, 75 Quai d'Orsay, 75007 Paris, France

paul.berhaut@airliquide.com (P.B.), ikhlass.yaya-oye@airliquide.com (I.Y.O.), axelle.albot@airliquide.com(A.A.)

*Corresponding author

Manuscript received March 31, 2023; revised April 25, 2023; accepted May 15, 2023; published February 7, 2024

***Abstract*—In this paper, we investigate deep Q-learning algorithms to optimize gas production planning in stochastic conditions. To demonstrate the value of reinforcement learning for gas production planning, we model the physical behavior of an industrial asset - an Air Separation Unit – based on historical data, electricity prices and customers' consumption patterns.**

We use the well-established reinforcement learning framework with non-episodic tasks and discounted rewards designed to minimize production costs and discourage insufficient production. We compare reinforcement learning agents to agents based on MILP (Mixed Integer Linear Programming) solvers. MILP solvers are currently used by energy-intensive industries to plan production based on imperfectly forecasted states. With these solvers, taking forecast uncertainty into account leads to high computational complexity (stochastic methods) or potentially conservative results (robust optimization). While demonstrating similar results in low-uncertainty scenarios, the DQN agents have shown better resilience to high amplitude uncertainties. They have demonstrated an efficient risk-averse strategy that outperforms the MILP baseline. DQN algorithms also gain advantage with their ability to be trained on infinite horizons, compared to MILP solvers where the state at the end of a finite horizon is set manually.

***Keywords*——reinforcement learning, stochastic environment, production scheduling, air separation**

I. Introduction

A. Background, Related Work

Industrial gas companies (IGC) are electricity-intensive businesses that require large electricity consumption [1]. Their competitiveness depends on their ability to adapt to the fluctuations in the cost of electricity. Storage tanks are commonly used to decouple production from consumption, which enables adaptation to short-term cost variations.

Academic and industrial works have shown the ability of deterministic algorithms to reduce energy costs of air separation unit (ASU) processes through stock management [2, 3]. The rise of intermittent renewable energy sources and worldwide events disturbing the stability of energy supply push forward the need for more resilient asset management that would take into account these uncertainties in the cost.

Among various stochastic and robust optimization algorithms [4], reinforcement learning and deep-reinforcement learning have been foreseen to solve combinatorial, partly unknown, and non-linear decision-making problems. Reinforcement learning has already been successful in optimizing real-life complex problems in stochastic environments [5, 6]. Within the realm of industrial gas companies, deep reinforcement learning algorithms have recently been shown to outperform classical linear model controllers when used for the purpose of optimal process control of an ASU [7].

In spite of all promising experiments, RL algorithms remain rare in industrial operational decision-making processes. This paper introduces deep Q-learning methods to optimize an ASU production planning under uncertainty. The experiment methods are then compared with currently used deterministic tools, MILP solvers.

B. The ASU Production Planning Problem

An Air Separation Unit (ASU) is an industrial plant built to separate air into its main components (nitrogen, oxygen and sometimes argon), usually via fractional distillation. These products are considered commodities that are sold to industrial clients. Customers expect their supplier to be able to provide the required quantities of product on an on demand basic, without having first announced their consumption.

Air Separation Units are typically electricity-intensive plants where a large portion of operating costs come from electricity consumption. It is commercially important to minimize these costs, both during design and operation. These plants can be designed to produce molecules either in gaseous or liquid form. When in liquid form, the product can be stored for long periods, creating the opportunity to decouple production from consumption.

In this paper, we consider the production planning problem to answer the following question: *"Given an expected consumption profile by customers and expected market price of electricity. When should an ASU produce, and what product should it prioritize?*". A formal description of this problem is given in (1).

C. Industrial Use-Case

The use-case considered in this study is an Air Liquide ASU located in France. A simplified representation of the system is shown in Fig. 1. The process uses electricity to power a main compressor and two liquefiers. Three final products are sold to consumers: liquefied oxygen (LOX), gaseous nitrogen (GAN) and liquefied nitrogen (LIN). Liquefied molecules may be stored in tanks, whereas gaseous molecules cannot be stored. The oxygen liquefier can only operate if liquid nitrogen is simultaneously vaporized to assist in generating cold. The plant is operated in a way such that all incoming customer requests are satisfied at all times. The ASU storage capacity represents typically a few days' worth of production.

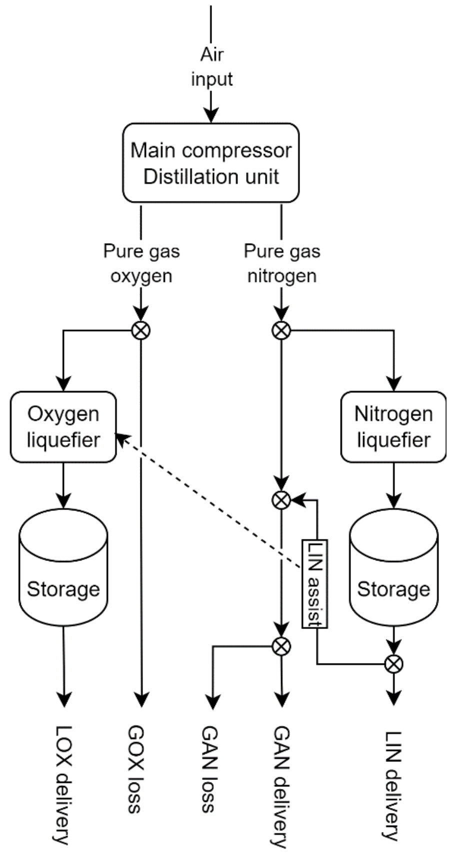


Fig. 1. Simplified description of the Air Separation Unit use-case *(LOX: liquid oxygen, GOX: gaseous oxygen, GAN: gaseous nitrogen, LIN: liquid nitrogen).*

D. Linear Programming Solutions

A standard approach to plan production is to formulate the problem as a Mixed Integer Linear Program (MILP). It is not the objective of this paper to fully describe existing MILP methods. We will therefore only give a brief overview of the approach. The problem is discretized at an hourly scale over a one week period, with the following objective and constraints:

Objective:

$$\underset{\substack{subject\ to\ constraints \\ (2)\ to\ (12)}}{\text{minimize}} \sum_{t}^{\infty} \gamma^t P_t C_t \tag{1}$$

with:
P_t: The electricity price forecasted for time t.
C_t: The overall plant electricity consumption for time t.

Subject to constraints:

Tank storage:

$$L_{t+1,LIN} \leq L_{t,LIN} + Q_{t,LIN}^{liquefier} + Q_{t,LIN}^{external} - Q_{t,LIN}^{lin_{assist}} - Q_{t,LIN}^{client} \tag{2}$$

$$L_{t+1,LOX} \leq L_{t,LOX} + Q_{t,LOX}^{liquefier} + Q_{t,LOX}^{external} - Q_{t,LOX}^{client} \quad \forall\, t \geq 0 \tag{3}$$

$$0 \leq L_{t,p} \leq L_p^{max} \qquad \forall t \geq 0, \forall p \in \{LIN, LOX\} \tag{4}$$

Final tank level:

$$L_{T,LIN} \geq L_{final,LIN} \tag{5}$$

$$L_{T,LOX} \geq L_{final,LOX} \tag{6}$$

with $L_{final,LIN}$ and $L_{final,LOX}$ parameters chosen by the operator.

Power consumption:

$$C_t = a_0 * Q_{t,air}^{compressor} + a_1 * Q_{t,LIN}^{liquefier} + a_2 * Q_{t,LIN}^{external} + a_3 * Q_{t,LOX}^{liquefier} + a_4 * Q_{t,LOX}^{external} \tag{7}$$

LIN assist:

$$a_5 * Q_{t,LOX}^{liquefier} \leq Q_{t,LIN}^{assist} \tag{8}$$

Production:

$$Q_{t,LIN}^{liquefier} \leq a_6 * Q_{t,air}^{compressor} \tag{9}$$

$$Q_{t,LOX}^{liquefier} \leq a_7 * Q_{t,air}^{compressor} \tag{10}$$

$$0 \leq Q_{t,air}^{compressor} \leq Q_{max,air}^{compressor} \tag{11}$$

Satisfy customer orders:

$$Q_{t,p}^{client} = Q_{t,p}^{client_{requested}} \quad \forall t \geq 0, \forall p \in \{LIN, LOX, GAN\} \tag{12}$$

with: $a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7$ values fitted to represent the real-world plant behavior.

Variables named Q represent flows in Nm^3/h and variables named L represent tank levels in Liters.

The decision variables $Q_{t,LIN}^{external}$ and $Q_{t,LOX}^{external}$ represent the possibility - not shown on Fig. 1 - to buy an unlimited quantity of liquefied nitrogen and oxygen from an external supplier if the plant's production capacity is insufficient. With these decision variables, the MILP problem is guaranteed to have a feasible solution.

The problem is repeatedly solved every time new inputs (electricity price or customer delivery forecasts) are received. The production schedule is based on the hourly values taken by decision variables $Q_{t,air}^{compressor}$, $Q_{t,LIN}^{liquefier}$, $Q_{t,LOX}^{liquefier}$ and $Q_{t,LIN}^{assist}$.

E. Limitations of MILP Solvers

Limited horizon effect:

Calculation is done at a fixed horizon, which requires operators to explicitly force storage levels at the end of a one-week period. These decisions are taken by ASU operators and are not optimized by the solver; hence, these decisions may be suboptimal.

Forecast error effect:

MILP solvers do not account for forecast error in their inputs, and may return solutions that are not robust to small differences between forecasts and reality.

Problem linearization:

MILP solvers rely on describing the dynamic behavior of an ASU with linear expressions for the problem resolution to remain computationally feasible. Many physical laws are not linear; hence, the problem solved is not fully representative of the actual plant behavior.

II. A Reinforcement Learning Approach for ASU Production Planning

A. Problem Statement

We reformulate the ASU Production Planning problem within the classical reinforcement learning framework [8] with discounted rewards and non-episodic tasks. We design the reward to minimize production costs and discourage insufficient production. RL agents interact hourly with the environment. Their actions define the plant's operating parameters $Q^{compressor}_{t,air}$, $Q^{liquefier}_{t,LIN}$, $Q^{liquefier}_{t,LOX}$ and $Q^{assist}_{t,LIN}$ for the next hour. Stochasticity is built into the environment as the agents observe imperfect forecasts of future prices and consumptions. More details regarding the learning environment design are given below.

State:

The agent observes the current status of the plant *(LIN and LOX storage levels, time of day, day of week)* and hourly forecasts for electricity price, GAN consumption, LIN consumption and LOX consumption over a 1-week horizon.

Actions:

Actions define the ASU setpoint for $Q^{compressor}_{t,air}$, $Q^{liquefier}_{t,LIN}$, $Q^{liquefier}_{t,LOX}$ and $Q^{assist}_{t,LIN}$ within the next hour. These actions must meet the following rules representing the fact that the plant does not have infinite production capabilities.

$$0 \leq Q^{compressor}_{t,air} \leq a_8 \qquad (13)$$

$$0 \leq Q^{liquefier}_{t,LIN} \leq a_9 * Q^{compressor}_{t,air} \qquad (14)$$

$$0 \leq Q^{liquefier}_{t,LOX} \leq a_{10} * Q^{compressor}_{t,air} \qquad (15)$$

with a_8, a_9 and a_{10} fitted to represent the real-world plant behavior.

Reward:

Recall that the overall objective of the ASU Scheduling Problem is to minimize a sum of electricity costs subject to a constraint that all customer orders must be satisfied, as shown in formula (1).

We approximate this problem with the relaxed formula below, where the new term G_t represents a penalty when the customer orders constraint is not satisfied.

$$\text{minimize} \sum_{t}^{\infty} \gamma^t (P_t C_t + G_t) \qquad (16)$$

We define the penalty term $G_t = L_t * 500\ €/MWh$, where L_t is the energy consumption that would have been required to produce the customer order that cannot be fulfilled at timestep t.

Recognizing in the formula above the formulation for discounted future rewards, we define R_t the reward received by an agent at each timestep as:

$$R_t = P_t C_t + L_t * 500\ €/MWh$$

Note that unlike usual, agents are required to *minimize* the sum of rewards received.

Discount factor:

We use a factor $\gamma = 0.9995$ to discount future rewards. This value is chosen such that the cumulative discount rate over one week γ^{168} is approximately 0.9. Sensitivity tests conducted during the study have not shown a significant impact when varying γ.

B. Discrete Actions

To allow agents to sample from a finite set of actions, a conversion layer was implemented to transform nine discrete actions into continuous parameters $Q^{compressor}_{t,air}$, $Q^{liquefier}_{t,LIN}$, $Q^{liquefier}_{t,LOX}$ and $Q^{assist}_{t,LIN}$.

This conversion layer enforces the rules:

1) The main compressor may only run at 0%, 65% or 100% of its maximum capacity.
2) The oxygen liquefier is either turned off, or liquefies all the available gaseous oxygen.
3) The nitrogen liquefier is either turned off, or liquefies all the available gaseous nitrogen.
4) The $Q^{assist}_{t,LIN}$ flow is set to the minimum amount required for LOX production.

We summarize the list of discrete actions available to the agent and the corresponding continuous values taken by flowrates in the ASU in Table 1 below.

Table 1. Discrete to continuous action transformation

Action	$Q^{compressor}_{t,air}$	$Q^{liquefier}_{t,LOX}$	$Q^{linassist}_{t,LIN}$	$Q^{liquefier}_{t,LIN}$
1	0	0	0	0
2	$0.65 * a_8$	0	0	0
3	$0.65 * a_8$	0	0	all GAN available
4	$0.65 * a_8$	$0.65 * a_8 a_{10}$	$Q^{liquefier}_{t,LOX} a_5$	0
5	$0.65 * a_8$	$0.65 * a_8 a_{10}$	$Q^{liquefier}_{t,LOX} a_5$	all GAN available
6	a_8	0	0	0
7	a_8	0	0	all GAN available
8	a_8	$a_8 a_{10}$	$Q^{liquefier}_{t,LOX} a_5$	0
9	a_8	$a_8 a_{10}$	$Q^{liquefier}_{t,LOX} a_5$	all GAN available

Using discrete actions implies that the agent only has access to a subset of the permissible action space. Discrete agents are therefore handicapped when compared to continuous agents.

This conversion layer makes the environment compatible both with agents using discrete action spaces *(such as DDQN [9] for example)* or continuous control schemes *(such as Soft Actor Critic [10] for example).*

C. Baseline

We define a baseline agent encapsulating the MILP solvers described in section *Linear Programming Solutions*. An optimization is run every hour using one-week horizon forecasts observed from the environment. Storage levels at the end of the horizon are constrained to be strictly above that tank's average level at that time of the week in historical plant data. This baseline agent serves as a reference for what can be achieved with MILP solvers. The agent is implemented in Python, using the OR-Tools library.

It can be noted that the baseline is placed in an advantageous position. Indeed, MILP problems are seldom solved every hour, they would typically be solved every 2-3 days at a 2 week horizon. Moreover we recall that this MILP problem's variables are continuous whereas RL agents are limited to discrete actions.

III. Learning Environment

A. Historical Data

Four years of hourly historical data for electricity prices and customer consumption were obtained, as well as 18 months of historical electricity price forecasts: 500 different forecasts made daily, at the hourly time scale, over a 2 weeks horizon.

Electricity prices were retrieved from the website of ENTSO-E, the European Network of Transmission System Operators. The customer consumption database was extracted internally. Price forecasts were obtained from a commercial vendor.

Based on this historical data, we generate synthetic scenarios which will be implemented and played out within our Gym environment [11].

B. Synthetic Scenario Generation

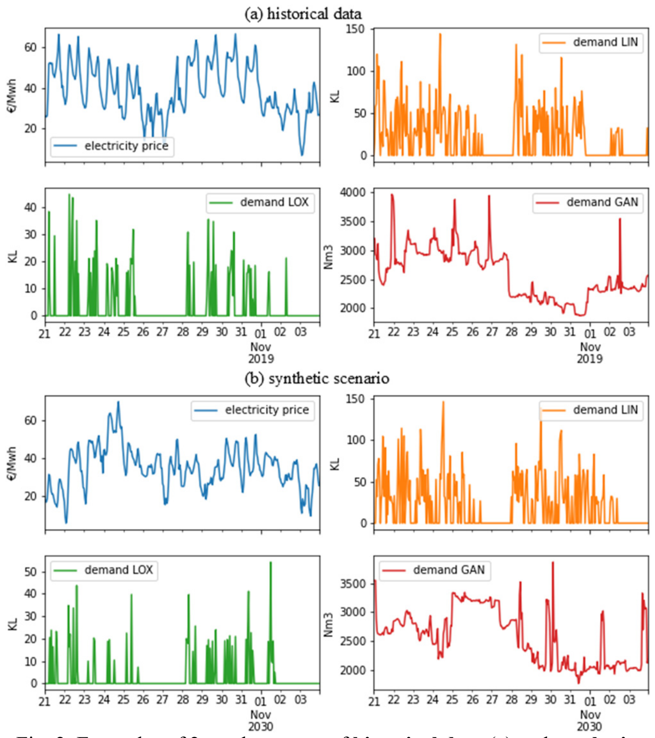


Fig. 2. Examples of 2-week extracts of **historical data** (a) and **synthetic scenario** (b) for the time-series electricity price, LIN demand, LOX demand and GAN demand.

We implemented a custom bootstrap resampling algorithm to generate synthetic time series from historical data. This algorithm splits historical data into 24-hour windows, shuffles and then groups the windows to create a synthetic new series. The shuffle and group procedure is constrained to:

1) preserve daily and weekly seasonalities from historical data *(prices are low during nights and week-ends)*
2) ensure smooth junction between two consecutive sampled windows *(prices do not jump suddenly at midnight)*
3) preserve the underlying structure of historical data by prohibiting transitions between two windows unlikely to follow each other *(prices rarely go from very high to very low within a day)*

In using this method, we make several assumptions:

1) Yearly seasonalities have limited impact on the optimal policy. We expect this to be true as the typical storage time frame on an ASU is measured in weeks.
2) The time series representing electricity price and customer demand are not conditionally dependent on each other.
3) The times series used as historical data are stationary, or can be transformed to be stationary

This algorithm allows us to generate an infinite number of scenarios representative of historical data, without the risk that a reinforcement learning agent overfits on repeated observations from the environment.

We show in Fig. 2 an example of historical and synthetic time series. These figures illustrate that we were able to capture the historical series' behavior and reproduce it in the synthetic scenarios.

C. Imperfect Forecasts

The environment is partially observable as agents are only allowed to plan production based on imperfect forecasts.

For customer gas consumption, the forecast is based on the average consumption over the past three days.

For customer liquid consumption, this imperfection is modeled as a random multiplicative error sampled weekly from a Gaussian distribution centered on 1 with 0.15 standard deviation. This model is consistent with Air Liquide logistics experience.

For electricity prices, this imperfection is modeled as an additive forecast error. Let us denote $\delta(e_i, f_j)$ the forecast error made by the forecast emitted at time e_i for time f_j. Analysis of historical data shows that a correlation exists between $\delta(e_i, f_j)$ and $\delta(e_i, f_{j+k})$ with a Pearson coefficient above 50% for k up to approximately 5 hours. Strong correlation also exists between between $\delta(e_i, f_j)$ and $\delta(e_{i+k}, f_j)$ with a Pearson coefficient above 50% for k up to 14 days.

We implemented a custom algorithm to generate synthetic forecasts errors replicating the characteristics observed in historical data.

IV. Results

We define three environments (E_0, E_1 and E_3) with varying levels of stochasticity. Stochasticity is minimal if agents observe future electricity prices and consumptions without any forecast error. Stochasticity levels can be controlled with a multiplicative coefficient applied to synthetic forecast errors. Environments E_0, E_1 and E_3 respectively have forecast errors with 0, 1 or 3 times the amplitude of real-life errors.

We implement two agents A_0 and A_1 in Python using the Jax, Haiku, and Coax libraries. These agents are trained in the corresponding E_0 and E_1 environments using the DDQN algorithm with Prioritized Experience Replay and N-Step transitions. Training runs required 18 million interaction steps with the environment over the course of 48 hours to converge, representing approximately 2000 years of simulated operation.

The agents (Baseline, A_0 and A_1) are tested in all environments with fixed seeds such that all agents are placed in comparable scenarios. The agents' performance is measured by their total undiscounted reward over a hundred thousand timesteps (approximately twelve years of simulated operation). Recall that the reward represents the cost of production and must therefore be minimized. Test scores are homogeneous to M€ spent for production and summarized in Table 2.

Table 2. Agents test scores within each environment

Environment	Baseline	Agent A_0	Agent A_1
E_0	105	**102**	110
E_1	**106**	**106**	111
E_3	131	162	**116**

The overall best performance is achieved by a reinforcement learning agent. We hypothesize that the Baseline MILP agent can be hindered by minor uncertainties remaining in customer delivery profiles and suboptimal constraints that are placed on storage levels at the end of the forecast horizon Reinforcement learning agents can learn policies that are not limited to the observable horizon of the environment.

Within the highly stochastic E_3 environment, agent A_1 has learned the most effective risk-averse strategy. This is expected as A_1 is the only agent to have encountered forecast errors during training. In fact, the stability of that agent's score across all environments suggests that it has learned a policy not reliant on the quality of forecasts received as observations.

It is highly surprising that agent A_1 is the worst performer in environment E_1 where it has been trained. We are still investigating this observation.

Table 3. Agent behavior in environment E1

Criteria	Baseline	Agent A_0	Agent A_1
% of reward due to insufficient storage	**22%**	6%	1%
%likely suboptimal action	0%	**7%**	-

In Table 3, we analyze the agents' behavior in environment E_1 according to the criteria below.

Remembering that the reward is designed to carry two signals *(minimize the cost of production, discourage insufficient storage)*, we measure the proportion of both signals within the total reward received by each agent. The baseline agent is significantly more penalized for insufficient production (22%) than reinforcement learning agents (6%). This is likely due to the ability of MILP solvers to target very precisely zero storage level, combined with the randomness of customer consumption.

Some actions were marked as likely suboptimal based on expert rules such as: *"If a tank is already full, it is unnecessary to produce more liquid as it cannot be stored"*. While the baseline agent makes no such mistakes, agent A0 has a very high proportion (7%) of likely suboptimal actions. Such information may be used in future training to heuristically guide trained agents towards better policies.

We also tested the agents within other environments where forecast errors were only applied to electricity prices or customer consumption, but not both. The agents' test scores were minimally impacted by electricity price forecast errors whereas they were strongly influenced by customer consumption forecast errors.

V. Discussion

We note that any difference between real-world scenarios and our synthetic scenarios would be detrimental to the performance of RL agents while the baseline agent would be unaffected. Therefore, the performance shown in this article should be considered as an upper bound of what could be observed in the real world. While more effort can be made to improve the quality of synthetic scenarios (e.g. handle yearly seasonalities or consider the impact of electricity price on customer demand). In the future, we will prioritize developments in the agents instead of more sophisticated scenario generation.

Indeed, the agents trained in this article use a relatively basic flavor of the deep Q-Learning algorithm, without recent extensions such as Distributional Reinforcement Learning. Given the stochastic nature of our environment, we expect a distributional approach would have been beneficial for the agents' performance.

We observe that the parameterization of insufficient production penalization has a strong effect on optimal policies: further work is required to ensure fair balance between storage level optimization and risk of insufficient production.

In this article, we used a simplified representation of an ASU's dynamic behavior. Future work will incorporate a more representative ASU model within the same training framework.

VI. Conclusion

Our work lays the foundation for future research in applying reinforcement learning for ASU production planning by establishing representative environment dynamics, realistic scenarios and a strong baseline against which RL agents may be compared.

We applied a deep Q-learning approach to plan industrial air gasses production in stochastic conditions. The RL trained agents show ability to minimize the energy costs while ensuring sufficient levels of production. Their performance remains stable despite varying degrees of data uncertainty: matching the MILP baseline in low-uncertainty scenarios and outperforming it in realistic and high-uncertainty cases. The developed agents are resilient to customers' consumption forecast errors.

VII. Acceptability

An important step to bring RL agents in operational use for ASU production planning is to assert their reliability on the field. This entails pilot phases during which operational teams can gauge the trust they place in trained deep reinforcement learning agents, ensuring actions taken are both explainable and safe to guide industrial operations.

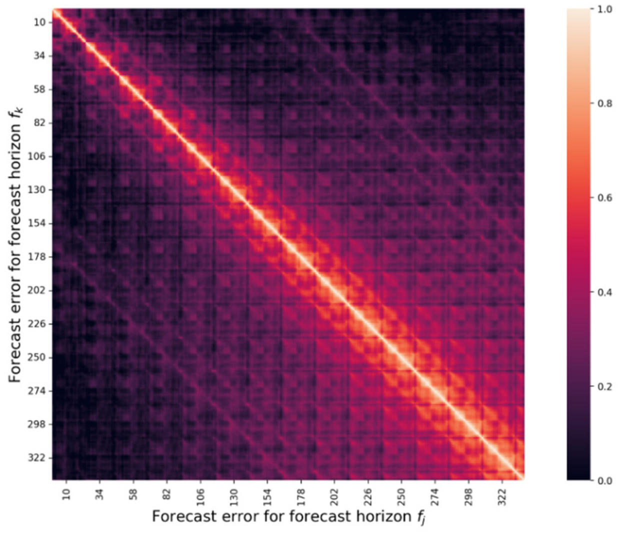
Appendix

A. Electricity Price Forecast Error

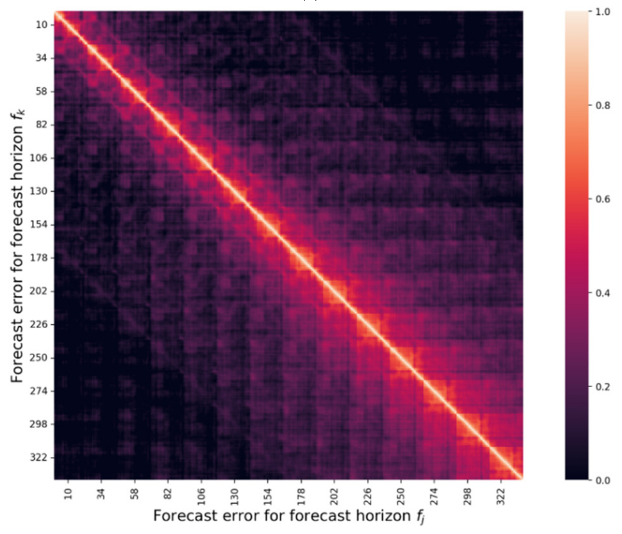
We plot in Fig. 3 (a) the correlation between forecast errors

$\delta(e_i, f_j)$ and $\delta(e_i, f_k)$ within **historical** forecasts emitted. e_i represents the time at which the forecast was made. f_i and f_j represent two different timesteps at which this forecast made a prediction. With this plot we try to answer the question: *If Monday's forecast overestimates the electricity price for Wednesday at 1PM, is that same forecast also likely to overestimate the price for Wednesday at 2PM?* Light colored values indicate strong correlations. Patterns emerge where forecast errors are strongly correlated when up to 5 hours apart or 24 hours apart.

We also plot in Fig. 3 (b) the same correlation for the **synthetic** forecast errors that were generated. As the figures are similar, we are satisfied that we were able to capture the historical forecast errors' behavior and reproduce it in our synthetic scenarios.



(a)



(b)

Fig. 3. Forecast error correlation within forecasts, compared between historical data (a) and synthetic scenarios (b).

Historical and synthetic scenarios were also compared on a variety of different metrics not detailed in this article, we were satisfied that the learning environment is sufficiently representative of real-world dynamics for agents to learn meaningful policies.

B. Training Procedure

Agent A_0 was trained from scratch in environment E_0. Agent A_1 was trained in environment E_1 starting from the weights learned by A_0 in E_0. We summarize in Table IV the hyper parameters used during the final training of agents A_0 and A_1.

Table 4. Training hyper parameters

Learning rate	10^{-6}
Batch size	256
Optimizer	Adam
N-steps	3
Replay buffer size	300000
Epsilon (exploration)	Starts at 0.99, linearly decreases towards 0.1 over 1M iterations, then 0.05 after 500k more iterations
PER α	0.6
PER β	Starts at 0.4 and linearly increases to reach 1 after 1M iterations
Neural network architecture	1 hidden layer with 100 neurons, Leaky ReLU activation

Conflict of Interest

The authors are employed by Air Liquide, an industrial gas company that operates air separation units.

Author Contributions

Paul Berhaut and Axelle Albot conceived the analysis and collected the data; Ikhlass Yaya-Oyé, Paul Berhaut and Axelle Albot contributed analysis tools; Ikhlass Yaya-Oyé performed the analysis; Paul Berhaut, Axelle Albot and Ikhlass Yaya-Oyé wrote the paper.

References

[1] Air Liquide, *2022 Universal Registration Document*, p. 34, 2022.

[2] R. Adamson, M. Hobbs, A. Silcock, and M. J. Willis, "Steady-state optimisation of a multiple cryogenic air separation unit and compressor plant," *Applied Energy*, vol. 189, pp. 221-232, Mar. 2017.

[3] M. T. Kelley, R. C. Pattison, R. Baldick, and M. Baldea, "An MILP framework for optimizing demand response operation of air separation units," *Applied Energy*, vol. 222, pp. 951-966, Jul. 2018.

[4] N. V. Sahinidis, "Optimization under uncertainty: State-of-the-art and opportunities," *Computers & Chemical Engineering*, vol. 28, no. 6-7, pp. 971-983, Jun. 2004.

[5] J.-H. Lee and J. W. Labadie, "Stochastic optimization of multireservoir systems via reinforcement learning," *Water Resources Research*, vol. 43, no. 11. American Geophysical Union (AGU), Nov. 2007. doi: 10.1029/2006wr005627.

[6] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, "Reinforcement learning – overview of recent progress and implications for process control," *Computers & Chemical Engineering*, vol. 127, pp. 282-294, Aug. 2019. doi: 10.1016/j.compchemeng.2019.05.029.

[7] N. Blum, V. Krespach, G. Zapp, C. Oehse, S. Rehfeldt, and H. Klein, "Investigation of a model-based deep reinforcement learning controller applied to an air separation unit in a production environment," *Chemie Ingenieur Technik*, vol. 93, no. 12. Wiley, pp. 1937-1948, Nov. 04, 2021. doi: 10.1002/cite.202100094.

[8] R. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[9] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," arXiv, 2015. doi: 10.48550/ARXIV.1509.06461.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," arXiv, 2018. doi: 10.48550/ARXIV.1801.01290.

[11] G. Brockman *et al.*, "OpenAI Gym," arXiv, 2016. doi: 10.48550/ARXIV.1606.01540.