Received: 1 March 2024 | Revised: 12 June 2024 | Accepted: 7 August 2024

DOI: 10.1002/aic.18584

AIChE JOURNAL

RESEARCH ARTICLE

Process Systems Engineering

A reinforcement learning approach with masked agents for chemical process flowsheet design

Simone Reynoso-Donzelli | Luis Alberto Ricardez-Sandoval

Department of Chemical Engineering, University of Waterloo, Waterloo, Ontario, Canada

Correspondence
Luis Alberto Ricardez-Sandoval, Department of Chemical Engineering, University of Waterloo, Waterloo ON, Canada N2L 3G1.
Email: laricard@uwaterloo.ca

Funding information
Natural Sciences and Engineering Research Council of Canada

Abstract

This study introduces two novel Reinforcement Learning (RL) agents for the design and optimization of chemical process flowsheets (CPFs): a discrete masked Proximal Policy Optimization (mPPO) and a hybrid masked Proximal Policy Optimization (mHPPO). The novelty of this work lies in the use of masking within the hybrid framework, i.e., the incorporation of expert input or design rules that allows the exclusion of actions from the agent's decision spectrum. This work distinguishes from others by seamlessly integrating masked agents with rigorous unit operations (UOs) models, that is, advanced thermodynamic and conservation balance equations, in its simulation environment to design and optimize CPF. The efficacy of these agents, along with performance comparisons, is evaluated through case studies, including one that employs a chemical engineering simulator such as ASPEN Plus®. The results of these case studies reveal learning on the part of the agents, that is, the agent is able to find viable flowsheet designs that meet the stipulated process flowsheet design requirements, for example, achieve a user-defined product quality.

KEYWORDS
chemical process flowsheet design, masking, proximal policy optimization, reinforcement learning, unit operations rigorous models

1 | INTRODUCTION

Chemical process flowsheets (CPFs) are an essential tool in chemical engineering, providing a visual representation of a chemical process by illustrating the arrangement of unit operations (UOs), their underlying stream connections, and the materials involved. Flowsheet optimization is thought of as a two-step procedure that involves the optimal synthesis problem, followed by the optimal design problem. On one hand, the synthesis optimization problem seeks to determine the arrangement of UOs to achieve a specific product starting from known reactants and products. On the other hand, the design optimization problem is focused on the determination of the optimal design and operating conditions (e.g., discrete and continuous variables) with a predetermined flowsheet.[1] Usually, these steps are solved separately and follow a set of rules derived from a combination of process data, and the insights and experiences of experts; this approach is referred to as the heuristic methodology. Typically, it follows a structured design approach; starting with the core units of the process, for example, reactors; followed by the inclusion of purification and separation processes, for example, distillation columns, absorbers; and ending with units for mass and energy integration, for example, heaters, mixers, splitters and recycles. While the designs generated through this method may be reliable, in many instances, the information available may be inaccurate or outdated concerning standards and quality parameters.[2]

With the development of advanced optimization and computational tools, CPF design underwent a significant improvement, enabling the integration of process synthesis and process design decisions.[3] With this integration, it is possible to define the CPF design problem as a single optimization problem, giving way to model-based optimization methods. The complexity of this method arises in the definition of the superstructure, which is a key aspect of the problem's definition in which all the possible UOs arrangements are represented. Consequently, the attainment of the optimal CPF depends significantly on the initial formulation of the superstructure, a factor frequently influenced by user experience.[4] The superstructure encompasses a comprehensive network of interconnections among various UOs, along with decision variables and structural design parameters. This comprehensive formulation frequently results in a mixed-integer nonlinear programming (MINLP) problem.[5] Moreover, the complex interaction between variables within the superstructure gives rise to generalized disjunctive programming (GDP) problems, augmenting the difficulty in finding a solution due to their inherent complexity and nonlinearity of the problem. Numerous studies and efforts have been reported on this subject to improve the efficiency and effectiveness in the simultaneous design and optimization of CPF, for example, Ma et al.,[6] Liñán et al.,[7] Bernal et al.,[8] and Mencarelli et al.[9] However, specialized algorithms and significant computational resources are typically required, making these methodologies less appealing to address the synthesis and design of large-scale CPFs.

As an alternative to those methodologies, a third method has recently emerged which aims to combine both approaches by incorporating the rules presented by the heuristic method into its formulation and using mathematical methods to synthesize and design CPFs. The versatility of this hybrid method—or model-free optimization method—enables to handle both equation-driven and data-driven problems. d'Anterroches et al.[2] were among the first authors to explore this hybrid methodology, developing a framework for synthesizing, designing, and optimizing CPFs by employing group contribution—a strategy widely applied in molecular design. Other notable techniques such as deterministic stochastic-algorithms,[1,10] genetic algorithms,[11,12] and Bayesian optimization[13] have been proposed to synthesize and design CPF. However, the greatest interest in hybrid methods emerged with the increased studies and advancements in Machine Learning and Artificial Intelligence techniques. Reinforcement Learning (RL)—a Machine Learning framework specialized in training agents to make sequential decisions through their interaction with an environment[14]—stands out as an attractive framework for CPF design, particularly for its ability to take sequential decisions that are interrelated, while optimizing complex processes.

One of the earliest studies that employed RL for CPF synthesis and design was proposed by Midgely.[15] That study proposed an agent capable of designing distillation trains to separate non-azeotropic mixtures. The RL-agent used was a hybrid soft actor-critic that interacted with an environment coupled with the COCO[16] simulation suite. The outcomes of that study aligned with baseline results, highlighting the effectiveness of CPF design with RL. Khan and Lapkin[17] introduced the use of hierarchical RL-agents, enabling enhanced interaction between the agent and the process, yielding superior results in comparison to baseline methods. Stops et al.[18] further advanced the application of RL-agents for CPF design by representing flowsheets as graphs, enabling the incorporation of graph neural networks into their agent's architecture. In their work, alongside this innovative feature, a hybrid agent that makes use of a Proximal Policy Optimizer (PPO) was considered. This hybrid agent facilitated simultaneous decision-making for both continuous and discrete actions. Gao et al.[19] addressed a similar problem, leveraging transfer learning and linking the environment to a more rigorous simulation suite such as DWSIM.[20] Göttl et al.[21] have introduced a series of compelling studies in which they utilize a turn-based two-player game environment called "SynGameZero" to design CPFs. Through their two-player approach, those authors were able to leverage a pre-existing tree search RL algorithm from DeepMind, enhancing the exploration capabilities of the agents. Subsequently, Göttl et al.[22] further improved their approach by introducing support for recycles and employing convolutional neural networks (CNNs) to solve more intricate problems. From their most recent studies, Göttl et al.[23] enhance their methodology by employing a hierarchical structure of RL agents, enabling them to address problems involving both discrete and continuous variables. The proposed frameworks have been tested using short-cut models to represent the UOs. Furthermore, van Kalmthout et al.[24] extended Midgely's[15] work by modifying the framework, allowing the interaction with the widely used ASPEN Plus®[25] simulation suite. Their study explored challenges encountered by the agent in terms of convergence—mostly attributed to the complexity in the design of distillation columns compared with short-cut methods—and the extensive training required for the agent that resulted in large computational costs. Beyond RL, but still within the area of Artificial Intelligence, a few works involving the design, complete or correct CPF have been proposed using generative transformers[26] and large language models.[27] For those type of works, a considerable number of flowsheets—complete or incomplete—, as well as a tokenization method—similar to the one presented by d'Anterroches[2]—is required. To this regard, Mann et al.[28] have further advanced d'Anterroches[2] work by incorporating Machine Learning and Artificial Intelligence techniques.

This work addresses an emerging area in CPF using RL: the use of masking, a strategy aimed at enhancing the decision-making process of the agent by excluding actions that should not be considered in the agent's decision spectrum. Huang and Ontañón[29] were the first to introduce the concept of masking in a RL context; they utilized a discrete PPO agent and, through various case studies, demonstrated the effectiveness of this technique compared with its standard version. Göttl et al.[21–23] were among the first to utilize the masking technique for the design of discrete CPFs; a detailed explanation of its functionality was not described in those works. The integration of a masking technique with a hybrid PPO approach remains unexplored in the literature. This innovative technique could be easily translated into a CPF problem, allowing to enforce logically constrained operations, for example, rule out the use of a pump to compress a gas. The application of masking in this context can be regarded as incorporating an expert input or design rules, similar to the heuristic method.

This work presents a CPF design and optimization framework using RL, where the employed agents are equipped with the masking technique, enabling them to effectively discard infeasible actions during the decision-making process. Through a reward-shaping process, the masked agents interact with the environment (comprised of a chemical engineering simulation suite) and learn to design attractive and economically viable CPFs. The integration of masking with an environment that utilizes advanced process simulators, ensures a closer alignment with a more realistic CPF design practices, as it allows the consideration of interdependence between UOs or a specific hierarchical order in their arrangement, while modeling UOs with rigorous thermodynamic and conservation balance equations. The agents are trained to design CPF at steady state, adhering to general process design, operational, environmental, and safety constraints. The main contributions of this work are as follows: (1) The deployment of a masked hybrid PPO (mHPPO) agent for CPF design and optimization. Despite undergoing the masking process within the discrete component, in the proposed hybrid agent discrete and continuous actions are taken simultaneously, as aspect that has not considered in the literature. Consequently, the effect of masking is not solely reliant on the discrete aspect but also encompasses the continuous part, thereby augmenting its complexity. (2) A rigorous simulation environment. This study proposed an advanced RL-environment that can be engaged with a chemical engineering simulator such as ASPEN Plus®, which makes use of rigorous UOs models and is flexible enough to operate with a wide variety of UOs, for example, plug-flow reactors, rigorous distillation columns with and without recycle, and so on. (3) A thorough explanation of the masking functionality. This study provides insights on the way RL generates flowsheets and learns complex structural relationships between linked UOs such as recycle streams utilizing the masking technique. A comparative study between the discrete and hybrid agent is performed by testing the agents in two case studies, shedding light in their benefits and limitations. This study is structured as follows: the next section presents the mathematical formulation of a CPF design problem. Section 3 presents the methodology, the adaptation of the mathematical problem into the domain of RL, and the description of the two used agents considered in this work. In Section 4, two case studies are presented and solved using both agents. The first case study is a synthetic case that demonstrates and compares the effectiveness of these agents in designing CPF, whereas the second case involves a more comprehensive process flowsheet, incorporating ASPEN Plus® as part of the simulation environment. Concluding remarks and future research directions are provided at the end.

2 | PROBLEM STATEMENT

Flowsheet design problems are typically expressed as MINLPs, with the main objective being the minimization of an objective function (f), which may be a combination of process economics, sustainability incentives, environmental requirements, and other factors. In this work, the vector of variables $\boldsymbol{x_c} \in \mathbb{R}^{n_{x_c}}$ denotes the continuous design variables that define the degrees of freedom of the problem, for example, equipment sizing, required purities, flow rates, among others; vector $\boldsymbol{x_d} \in \mathbb{Z}^{n_{x_d}}$ represents all the discrete design variables in the problem such as the number of stages in a distillation column or the number of tubes in a shell and tube heat exchanger. Moreover, $\boldsymbol{D} \in \{True, False\}^{n_D}$ includes the Boolean variables that appear in disjunctions to determine if a particular constraint vector is enforced or not. This disjunctive formulation is widely used in CPF design to incorporate structural and/or logical decisions, for example, the presence or absence of a UO, the placement or exclusion of a flow in a specific location, or the implementation or omission of a recycle at a particular point in the flowsheet. Vector $\boldsymbol{y} \in \mathbb{R}^{n_y}$ denotes the intermediate continuous variables calculated internally by the chemical process simulator, for example, material flows, compositions, and operating conditions. Equation (1), which serves as a CPF conceptual formulation for this work, can be posed as follows:

$$\begin{aligned} &\min_{\boldsymbol{x_d},\boldsymbol{x_c},\boldsymbol{y}} f(\boldsymbol{x_d},\boldsymbol{x_c},\boldsymbol{y}) \\ &s.t. \boldsymbol{g}(\boldsymbol{x_d},\boldsymbol{x_c},\boldsymbol{y}) \le 0 \\ &\Psi(\boldsymbol{D}) = True \\ &\mathrm{V}_{i \in P_q} \begin{bmatrix} D_{i,q} \\ \boldsymbol{h}_{i,q}(\boldsymbol{x_d},\boldsymbol{x_c},\boldsymbol{y}) \le 0 \end{bmatrix}, \forall q \in Q \\ &[\boldsymbol{x_d},\boldsymbol{x_c},\boldsymbol{y}] \in \Omega \\ &\boldsymbol{x_d^L} \le \boldsymbol{x_d} \le \boldsymbol{x_d^U}, \boldsymbol{x_c^L} \le \boldsymbol{x_c} \le \boldsymbol{x_c^U} \end{aligned} \tag{1}$$

Equation (1) presents various types of equations, among which the following can be highlighted. Constraints with a specific mathematical expression ($\boldsymbol{g}(\boldsymbol{x_d},\boldsymbol{x_c},\boldsymbol{y}) \le 0$): these inequalities frequently contain particular restrictions relating to UO design and operational limitations, such as the allowed operating pressure or temperature of a UO, as well as process design target criteria, such as reactant conversion or product purity. Moreover, Boolean variables $D_{i,q}$ indicates if constraint $\boldsymbol{h}_{i,q}$ is enforced or not. These constraints are related to the logical proposition $\Psi(\boldsymbol{D}) = True$ through the logical operators (e.g., AND, OR, XOR, negation, implication, equivalence), which are defined based on the specific disjunctive decisions considered in the problem. Note that Boolean variables are represented as binary variables in the MINLP and the upper limit for n_D is given by $n_D = \sum_{q \in Q} |P_q|$. Following Equation (1), $[x_d, x_c, y] \in \Omega$ represents the constraints handled by the chemical simulation suite; these restrictions enclose all the mass and energy balances, thermodynamic models, chemical equilibria and reaction rates, and more restrictions considered internally by the chemical simulator. Hence, the variables within the set Ω represent all the possible variable combinations that ensure a convergence of the model in the simulator.[1] Lower bounds (superscript L) and upper bounds (superscript U) for the discrete and continuous variables are also included in Equation (1). Addressing a problem like that depicted in Equation (1) through model-based optimization methods can be performed; however, it requires the specification of a comprehensive superstructure and constructing their corresponding process models. Additionally, one must reformulate the problem effectively or use solvers suitable to handle general disjunctive constraints. Furthermore, the limited availability of open-source software, the substantial

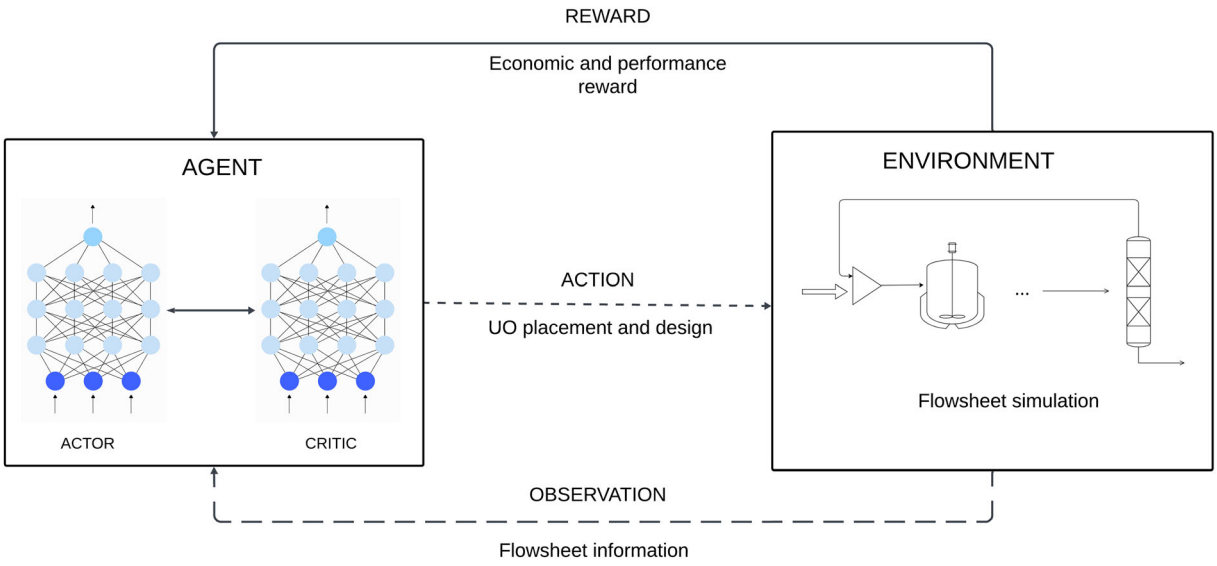

FIGURE 1 Agent-environment interaction.

computational expenses—particularly for large-scale problems—, coupled with their inflexibility to seamlessly integrate with simulation suites, exacerbate the problem's complexity. These challenges motivate the need to explore hybrid approaches. In the following section, we introduce an approach based on a RL strategy.

3 | SOLUTION FRAMEWORK

In this section, the RL approach used to solve the CPF design problem shown in Equation (1) is presented. The environment in RL is the external system or context with states, actions, and rewards, where an agent interacts to learn a policy that maximizes cumulative rewards over time. In RL, a step (i) denotes an individual interaction between the actor and the environment, that is, an instance of a simulation of the environment; likewise, an episode encloses a complete sequence of interactions from the initial state to a terminal state.[14] In parallel to traditional optimization, the environment corresponds to the mathematical problem formulation, encompassing all the variables, process model equations, constraints, and the objective function, while the solver functions as the agent interacting within this environment to acquire (i.e., learn) a strategy, that is, a policy, that aims to maximize cumulative rewards.

The interaction between the agent and the environment for CPF design is schematically represented in Figure 1. Section 3.1 provides details on the two agents presented in this work and their structures. Section 3.2 presents the action space, the observation, and the reward function. The addition of the novel masking technique proposed in this work is described within these sections.

3.1 | Agents

In this study, the foundational structure of the agents is based on the typical PPO architecture proposed by Schulman et al.[30] The agent's structure varies based on the nature of the addressed variables, adopting either a hybrid structure (i.e., a combination of discrete and continuous variables) or a conventional discrete PPO structure involving only discrete variables. Regardless of the agent, as a novel feature, both architectures incorporate the use of the masking methodology adapted from Huang and Ontañón.[29]

3.1.1 | Masked discrete proximal policy optimizer

The architecture of this agent presents a conventional structure, featuring a fully connected or dense neural network with one hidden layer having the same number of neurons. This was determined after several experiments where it was observed that increasing the number of hidden layers was not improving the training and instead leading to overfitting toward the training buffer and reducing the agent's exploration capabilities. Hyperbolic tangent (tan$h$) activation functions are applied to all layers except for the last one; this choice is made to avoid disparities in the raw output scores (logits). By doing so, prevents the creation of a strong bias toward one action over another during the network training, ultimately empowering the agent with enhanced exploration capabilities. Before reaching the activation function in the final layer, the masking filter is applied. The way this technique operates is as follows: as shown in Figure 2, at each step i the agent receives an observation vector o_i containing information about the current state of the environment. In addition to the observation vector, the agent also receives a masking vector (mv_i) from the environment, which indicates with Boolean variables the position of the actions to be masked. An illustrative demonstration of the functioning of the masking function in the environment is presented in Section S1. The integrated masking function in the agent works by setting the logit values (l) from the last layer of the neural network, with the same position as the actions to be masked, to a large negative number $(l \rightarrow -\infty)$. These modified values are then propagated through the activation function—*SoftMax*—of the final

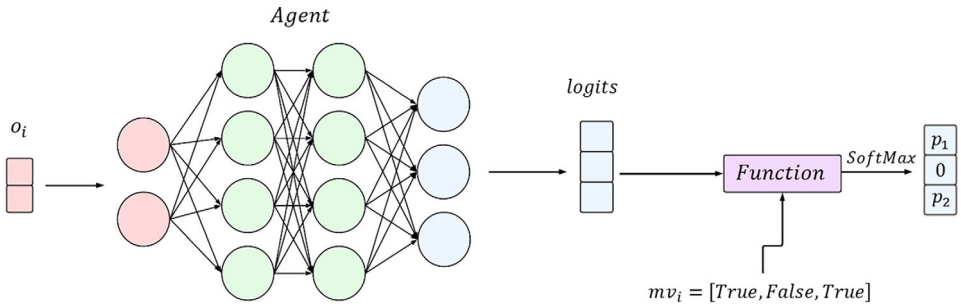

FIGURE 2 Illustrative example of the masking functionality.

layer of the actor-network, transforming large negative values into the probability domain, effectively setting them to zero, that is,

$$Softmax(\boldsymbol{l}) = \frac{e^{l_{mask}}}{\sum_{j=1}^{n_{x_d}} e^{l_j}} \forall mask \in n_{x_d} \text{ and } \boldsymbol{l} \in \mathbb{R}^{n_{x_d}}, \tag{2.1}$$

$$\lim_{l \to -\infty} e^{l_{mask}} = 0. \tag{2.2}$$

During training, the agent needs to randomly sample discrete actions, which is not directly feasible from this output. Therefore, once the probability vector is obtained, it is transformed into a categorical distribution, from which the agent can sample discrete actions at each step *i*.

3.1.2 | Masked hybrid proximal policy optimizer

The agent's architecture resembles the hybrid proximal policy optimizer (HPPO) proposed by Fan et al.,[31] with the minor modification of including the masking technique in the discrete component. Note that masking is applicable exclusively to discrete actions for this agent. Consequently, continuous actions cannot undergo masking, potentially leading the agent to consistently choose certain continuous actions, even if their corresponding discrete actions have been masked. The agent's architecture can be divided into two parts. The first part resembles an encoder structure, where the neural network exhibits a decreasing number of neurons as the layers become progressively deeper until reaching a two-neuron layer. In the second part, stemming from these two neurons, two separated fully connected neural networks emerge. The first network, which accounts for discrete decision-taking, shares a similar architecture as that described in Section 3.1.1, incorporating the masking mechanism explained above. The second network handles continuous decision actions; like most of the hidden layers in all the neural network, the activation functions in its hidden layers are hyperbolic tangent. In contrast to previous networks, this network outputs two parameters for each of the continuous decisions; these parameters could be referred to as α and β, which are parameters needed to shape a Beta distribution: $B(\alpha,\beta)$. The activation function used to change the logits into these parameters is a *SoftPlus* function. Similar to the masked discrete proximal policy optimizer (mPPO) agent, the mHPPO agent employs a categorical distribution for the discrete action sampling and multiple Beta distributions to sample the continuous actions. Note that during training, the shape of these distributions changes and adapts to a certain type of distribution, causing the agent to focus more on a specific region in the final training steps rather than the entire domain of the distribution.

3.2 | Environment

The environment is the segment of the RL framework where the optimization problem (Equation (1)) is translated into RL terms. This subsection delineates its distinct components: the action space, the observation vector, and the reward function. The action space is the sole component whose structure varies depending on the agent, which has been described above.

3.2.1 | Action space

The action space $(\mathcal{A})$ of a hybrid agent (mHPPO) is composed of the set of all discrete $(\boldsymbol{x_d} \in \mathbb{N})$ and continuous $(\boldsymbol{x_c} \in \mathbb{R})$ variables, that is,

$$\mathcal{A} = [\boldsymbol{x_d}, \boldsymbol{x_c}]. \tag{3}$$

Note that $\mathcal{A}$ does not encompass the combination of all discrete and continuous variables, but rather probability distributions. As the RL method requires the selection of an action at each step *i*, actions are sampled from these distributions, resulting in the step-action a_i. For the discrete variables, there is a categorical distribution from which only one value is stored in a_i. In the case of the continuous variables, each has its own distribution, that is, a Beta distribution. Thus, a_i is composed of n_{x_c} values, even if some of them are not strictly used in that particular step *i*.

For a fully discrete agent (mPPO) the continuous variables can be discretized and unlike the previous action space, this discretized version $(\mathcal{A}_{FD})$ contains all the combinations of the former discrete variables with their corresponding realizations of the discretized continuous variables. The primary motivation to discretize continuous variables within the action space is to streamline the agent's exploration domain, thus enhancing its interaction with specific actions. Note that this may not be feasible in a direct manner with large infinite search spaces produced by $\mathcal{A}$. This simplification comes with drawbacks,

including limited flexibility—discrete action spaces may not adequately represent the continuous nature of the variables—, loss of precision—discretizing actions can lead to a loss of precision in decision-making, as the granularity of actions is limited by the discretization—and the requirement for prior knowledge of the criterion used to perform the discretization. The discretization criterion depends on the case study under consideration, the computational budget, and the expected accuracy in the solution. Typically, a refined discretization will likely return accurate results at the expense of increasing the search space domain thus leading to a lengthy training, and consequently larger computational costs. Note that the present RL framework can be adapted to multiple discretization methods in the continuous variables' domain, for example, uniform, normal discretization. $\mathcal{A}_{FD}$ is defined as a database, consisting of keys and values; the keys represent the discrete actions available for the mPPO to take, while the values associated with these keys denote the different combinations of the discretized continuous realizations. Once the key is chosen by the discrete agent, its value is mapped in the database, and the associated value is used. For an illustrative example, refer to Section S1.

3.2.2 | Observation

The step-observation vector (o_i) is the portion of the state that is conveyed to the agent as information about the environment at each step i.[14] This is defined as follows:

$$o_i = [\varpi_i, \kappa_{i,s}, \gamma_i]. \tag{4}$$

The structure of o_i may change depending on the setting of the problem; however common features remain. For instance, at each step we keep track of: (a) process operating variables (ϖ_i), for example, temperature and pressure; (b) chemical components (reactants or products) tracker ($\kappa_{i,s}$), which could be molar, mass, or volumetric flowrates, as well as molar or mass fractions, where $s \in \{reactants, products\}$; (c) the step of the current observation (γ_i), to add temporal context to the vector and differentiate potential identical action responses. The elements in o_i are normalized, thus preventing larger values from having a greater influence on data processing within the agent's neural network.

3.2.3 | Reward

The reward function represents a numerical measure of the immediate benefit or cost derived from the agent's action in a specific state,[14] thus, the reward is assigned at each step (r_i). Designing an effective reward function is crucial since it needs to be explicit enough to facilitate the agent's assertive learning and sufficiently flexible to generalize well across the task at hand. For simple processes, such as the maximization of the agent's interaction with the environment or reaching a specific state, the design of the reward function is relatively straightforward. However, for more complex problems where the objective is not as punctual as the other cases—for example, opposing maximization and minimization objectives—and there are constraints to uphold, designing an effective reward function becomes a more challenging task. To tackle this challenge, a method referred to as reward shaping is used in this work:

TABLE 1 Reward shaping.

Optimization problem equations	Translation to RL sub-reward
Minimization of $f(\boldsymbol{x_d}, \boldsymbol{x_c}, \boldsymbol{y})$	Objective: The penalty increases as the values obtained from evaluating the function become more negative.
Constraints $\boldsymbol{g}(\boldsymbol{x_d}, \boldsymbol{x_c}, \boldsymbol{y})$	Process design goal: The penalty lowers as the constraint violation approaches the desired value.
Constraints $\boldsymbol{g}(\boldsymbol{x_d}, \boldsymbol{x_c}, \boldsymbol{y})$	Specific constraint (UO constraint): Constant value provided for the constraint satisfaction
—	Driving force: Varying reward depending on the magnitude of the driving force.
—	By-product reward: Varying reward depending on the achievement of the specified goal
—	Short flowsheets: Varying reward depending on the remaining steps.

$$r_i = \sum_{b \in B} r_b. \tag{5}$$

The step reward (r_i) used in this problem is decomposed into b-individual sub-rewards ($b \in B$) which promotes different objectives in the policy.[32] This approach is applied to the objective function and constraints included in Equation (1), that is, assign separate sub-rewards to each of these elements, as illustrated in Table 1. Additional to these elements, and to make the agent acquire a better understanding of the problem, extra sub-rewards are added. One of these extra sub-rewards is the driving force, defined as the difference (Δ) in one distinctive parameter, for example, conversion, molar fraction, purity, and so on. This sub-reward empowers the agent to actively progress toward the creation of a CPF rather than opting to remain stationary. In CPF design, the RL agent employs a specific product, as its driving force; however, the process may also produce other by-products of interest. To prevent the agent from losing track of these by-products, an additional sub-reward is introduced, where the agent is rewarded for achieving a specific objective with these by-products, for example, reaching a specified purity, produces a certain quantity of by-product, etc. Another sub-reward is introduced to motivate the agent to design short and effective CPF, so in the fewest steps possible. This sub-reward adds the unused steps ($I - i$) to r_i, where I represents the maximum number of steps allowed in an episode.

As presented in Equation (1), the main goal is to minimize the objective function. To avoid the agent from choosing the actions with the lowest penalty, the previously defined driving force is used to

balance out the rewards, thus avoiding trivial solutions. As mentioned in the problem statement section, the constraint $\boldsymbol{g}(\boldsymbol{x_d},\boldsymbol{x_c},\boldsymbol{y})$ can be divided into two categories, the process design goal requirements, and the specific constraints. The former only becomes relevant in the final steps of the episode. Failing to meet those constraints compels the agent to continue constructing the CPF, potentially leading to incorrect designs—resulting in highly negative rewards—or failing to satisfy the specified constraint within *I*, resulting also in a negative reward. Specific design constraints, which are only activated for certain actions, offer an additional constant sub-reward to the agent when the constraint is not violated. Treating these constraints as penalties, where the agent is penalized for any violation (similar to a constraint programming approach) may lead to poor learning rates. This formulation hinders the agent's exploration, resulting in convergence toward suboptimal results or, in some cases, it may lead to divergence. Therefore, a constant value reward is chosen; as it does not limit the agent's exploration freedom and rewards it when operating within the established criteria. Note that disjunctive constraints and constraints managed by the chemical simulator do not directly translate into the RL sub-reward shaping system. The latter is managed within the simulator, representing a black box for the agent, thereby denying direct access, while the former is addressed through the masking technique explained before and expanded in Section S2.

4 | CASE STUDIES AND DISCUSSION

In this section the methodology presented in the previous section is tested on two case studies adapted from the literature, each emphasizing distinct features. The two cases use both, the discrete and hybrid approaches, to design the process flowsheet and allow for a comparison of the results obtained from each methodology. More specifically, the first case study seeks to provide insights on how the proposed hybrid and discrete agents operate when designing CPFs by addressing the synthesis of a relatively simple process. On the other hand, the second case study aims to design a more advanced process, exploring the potential of the proposed RL framework as a possible commercial tool for CPF design. The two case studies were implemented on a PC with Intel® Core™ i7-3770 CPU@3.40 GHz and 32 GB of RAM, using Python as the programming language. The main libraries used were torch for the development of the neural networks and gym for creating the RL environment.

4.1 | Case study 1

The objective of this case was to design a CPF that enables the conversion of reactant A into product B up to a 97.5% conversion rate, while minimizing the process economics. For this case, three UOs (discrete actions) with their associated design variables (continuous actions) were considered. Note that the equations modeling each of these operations stem from simple mass and energy balances. This simplification enables a deeper focus on comprehending the design and optimization capabilities of the agents introduced in the previous section. The UOs involved in this case study include a mixer (*M*); a reactor (*R*),[33] with design variables being the diameter (*D*) and height (*H*) of the reactor; and a flash tank (*FT*), with the design variable being the vapor/feed fraction (*q*). A detailed description of the UOs, their models, and cost functions are provided in Section S3.

For the present case study, the action space ($\boldsymbol{\mathcal{A}}$) is composed by discrete actions $\boldsymbol{x_d} \in [M,R,FT]$ and the continuous actions $\boldsymbol{x_c} \in [\mathbb{R}_D,\mathbb{R}_H,\mathbb{R}_q]$. For the mPPO, $\boldsymbol{\mathcal{A}_{FD}}$ was discretized as follows. *D* and *H* in *R* were each discretized into three uniform realizations between their upper and lower limits. The resulting combinations between each of these realizations in *D* and *H* resulted in a total of 9 discrete actions. Regarding *FT*, the domain of *q* was uniformly divided into 6 discrete values. Taking into consideration the discrete action associated with the choice of a *M*, combined with the discrete actions of *R* and *FT*, a total of 16 discrete actions were considered for this problem. The observation vector at each step *i* contains information about the outlet stream of the current UO: concentration of the reactant, temperature, total molar flowrate, current step, and the achieved conversion, that is, $o_i = [C_A,T,F,i,x_A]$. The reward function considered in this case study follows the method described in Section 3.2.3 and is represented by four sub-rewards shown in Table 2.

Note that all sub-rewards listed in Table 2 were normalized to prevent an individual term from having a disproportionate impact, for example, the operating cost of a UO. Regarding the masking, to prevent the agent from selecting a trivial sequence like a *M* followed by a *FT*—where no conversion occurs—, the condition to unmask the *FT* requires the existence of at least one reactor between these operations. Additionally, to avoid confusion for the agent concerning the recycling, once *M* is chosen, that operation is masked and unmasked again only after *FT* is selected by the agent.

All the hidden layers in both agents consisted of 64 neurons. Both agents were trained 10 times for 75,000 steps. The networks were updated every 2048 steps with a discount factor of 0.99 and optimized using Adam[34] optimizer with a starting learning rate of 2.5e−4, which was adjusted throughout the training and from preliminary simulation tests.

Figure 3 illustrates the learning curves of both agents for a single training run. The solid line, representing the running mean, depicts the trend of cumulative rewards throughout the learning process, while the faded region indicates the cumulative rewards obtained at the end of each episode. In both cases, the running mean exhibits a similar behavior—an exponential growth in the initial episodes followed by a plateau when reaching a certain value. On the other hand, the behavior of cumulative rewards is notably different. In mPPO, the variability is almost reduced at the end of training, whereas mHPPO shows a slight reduction compared with the initial episodes. Also, the plateau reached by mHPPO indicates a potential room for improvement, as it does not reach the maximum cumulative rewards achieved by the agent, whereas mPPO does achieve this condition. The differences in behavior between agents can be attributed to the types of variables considered and their corresponding relationships.

TABLE 2 Sub-rewards for case study 1.

Sub-reward	Reward or penalty	Assigned	Description
Driving force	Reward	Every step	Quantified as the conversion difference between consecutive steps, it signals the UO's efficacy in transforming reactant A into product B.
UO cost	Penalty	Every step	Quantifies the cost associated with each UO.
Process design requirement	Penalty	Final step	Examines whether the desired conversion was attained. Depends on the gap between the achieved conversion and the desired conversion.
Short flowsheets	Reward	Final step	Denoted as the difference between the maximum number of steps *l* (which for this case was 10) and *i*.

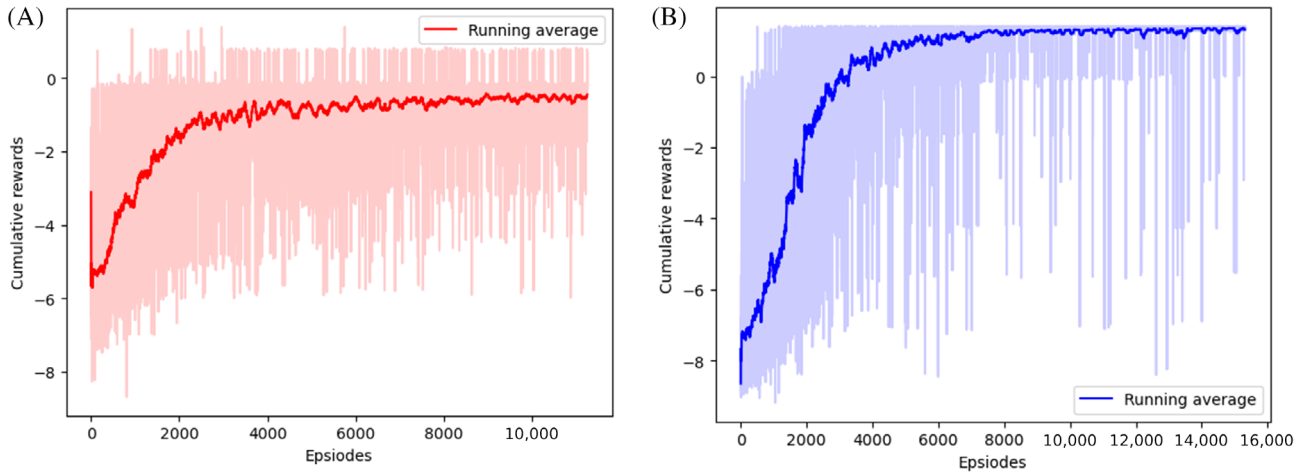

FIGURE 3 Learning curve for a single run: (A) mHPPO; (B) mPPO.

The mPPO operates with a single type of variable—discrete variables. Discrete variables generate a finite action space; hence, the agent has a finite number of actions to choose from. In addition to this, and boosted by the masking technique, the agent is capable of finding the optimal combination of actions to achieve maximum rewards. A convergence test on the number of discretization points is presented in Section S3.

On the other hand, mHPPO faces a more complicated task as it handles both discrete and continuous variables. In addition to creating an infinite search space, the combination of discrete and continuous variables has a more substantial impact on the agent's learning process. In the neural network, the agent's loss function consists of two components—one discrete and one continuous. Figure 4 presents the cumulative reward and losses profiles with respect to the agent's updates—optimization via backpropagation of the neural network. The losses often have opposing objectives, potentially leading to a trade-off. Figure 4 clearly illustrates this conflict, wherein as the agent learns—manifested by the increase in cumulative rewards—the discrete loss decreases while the continuous loss fluctuates. Throughout the training process, it was observed that at the initial updates of the network, as the continuous loss decreased, there was a corresponding increase in the discrete loss. This prompted a modification in the agent's strategy for the subsequent update, resulting in a reversal of roles. While both losses show a decreasing pattern over the updates, it suggests that the behavior of the continuous loss is influenced by the discrete loss, preventing it from exhibiting a consistently decreasing pattern. Moreover, the cumulative reward ceases to grow once the discrete loss reaches a plateau, also hindering the decrease in the continuous loss. This is reflected in the decision-making process of actions, where the agent engages in a trade-off between both discrete and continuous decisions, ultimately arriving at a response that could be deemed as the best trade-off between the two types of variables. Note that the behaviors exhibited in Figure 4 are consistent for the rest of all mHPPO runs.

When evaluating the best-performing agents, the resulting CPFs (Figure 5 and Table 3) are capable to reach the desired conversion while learning the advantage of adding recycling to the process flowsheet. Achieving shorter and superior flowsheets than the naïve assumption of merely placing reactors in series, which would have also achieved the desired conversion. The significant difference between the two flowsheets shown in Figure 5 lies in the sizing of the equipment, thereby affecting the length of the CPF. In the discrete case (mPPO), the agent selects the first reactor with the largest dimensions, the flash tank with the smallest liquid-vapor fraction, and the second reactor with the largest diameter and smallest height. Due to the previously explained behavior, the mHPPO agent seeks to find for a balance (trade-off) in its decisions by opting for reactors of medium size, compelling the process to have more units.

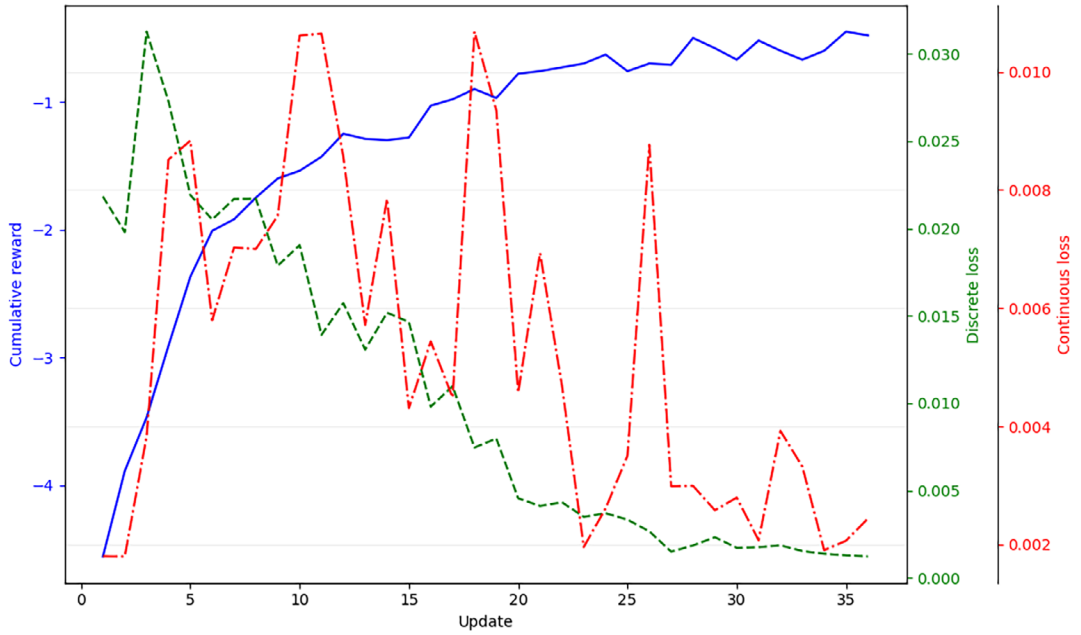

FIGURE 4 Reward and losses profiles for mHPPO.

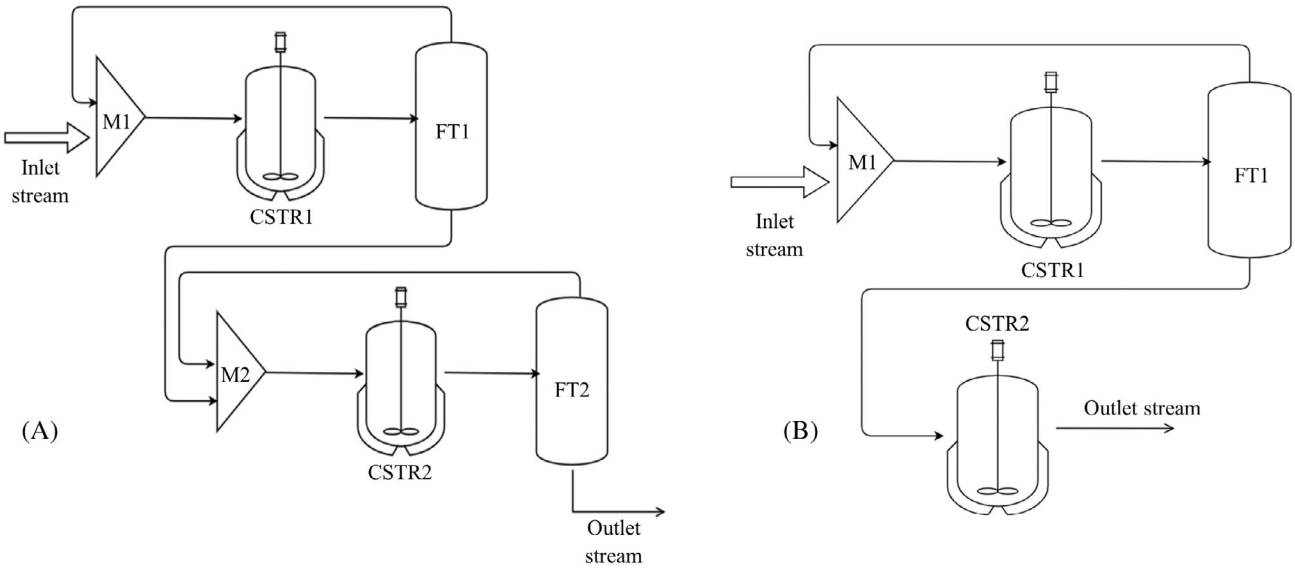

FIGURE 5 CPF obtained for case study 1: (A) mHPPO; (B) mPPO.

TABLE 3 UOs designs and specifications for case study 1.

Unit operation	Design variable	mHPPO	mPPO
CSTR1	Diameter (*D*) (m)	2.2073	2.3622
	Height (*H*) (m)	2.0409	2.3622
CSTR2	Diameter (*D*) (m)	2.0193	2.3622
	Height (*H*) (m)	2.0203	1.6764
FT1	Vapor/feed (*q*)	0.3501	0.2000
FT2	Vapor/feed (*q*)	0.2559	–
Achieved conversion (x_A)		0.9816	0.9768
Cumulative reward		−0.0951	1.3993

Outcomes from this case study provide insights that can aid in understanding the behavior of the PPO agents, and the type of rewards they should consider. When translating the optimization problem into RL terms, it is crucial to design a non-sparse reward system for the agent, that is, the agent should not receive all the rewards at the end of the episode. Instead, the agent must receive a reward at each step that is descriptive enough to allow it to learn the consequence of choosing an action in a specific position. In the case of the mHPPO, the actions taken align with this idea, as each action has the greatest impact on the conversion of the reactant, while seeking to minimize the process economics. Hence, the designed CPF may not be the shortest, but each action seeks to yield the highest reward. The mPPO operates on the

same principle; however, given the simplicity of the case study—a reduced and discretized search space—the agent is capable of identifying an attractive combination within the feasible action space.

4.2 | Case study 2

This case study aims to design the CPF for a dimethyl ether (DME) production plant, based on the work proposed by the authors Caballero et al.[35] and Luyben,[36] while minimizing the process economics. In contrast to case study 1, this case study features the use of a chemical simulation software (i.e., ASPEN Plus® v8.8) as part of the simulation environment. The key motivation behind the development of this hybrid platform that combines a computer programming language (i.e., Python) with a chemical simulation software such as Aspen Plus lays in harnessing the thermodynamic packages and equations governing all declared UOs within the latter. Notably, this approach facilitated the incorporation of a broader spectrum of UOs, resulting

TABLE 4 UOs for study case 2.

Unit operation	ASPEN Plus® block	Design variable	Bounds
Mixer (*M*)	*MIXER*	—	—
Pump (*P*)	*PUMP*	Pressure	Fixed to 10 bar
Heat exchanger (*HEX*)	*HEATER*	Operating temperature (T_{HEX})	150–400 (°C)
Cooler (*C*)	*HEATER*	Operating temperature (T_C)	5–50 (°C)
Plug flow reactor with constant cooling (*PFR*)	*RPLUG*	Length (*L*) Diameter (*D*)	0.5–3.5 (m) 6.5–12 (m)
Adiabatic plug flow reactor (*APFR*)	*RPLUG*	Length (*L*) Diameter (*D*)	0.5–3.5 (m) 6.5–12 (m)
Binary distillation column (*DC*)	*RADFRAC*	Number of stages (n_{stages}) Distillation rate (*dist*)	5–25 20–110 (kmol/h)
Binary distillation column with recycle (*DCR*)	*RADFRAC*	Number of stages (n_{stages}) Distillation rate (*dist*) Recycle ratio (*rr*)	5–25 20–110 (kmol/h) 0.5–0.95
Ternary distillation column (*TC*)	*RADFRAC*	Number of stages (n_{stages}) Distillation rate (*dist*) Intermediate distillation rate ($inter_{dist}$)	5–25 70–110 (kmol/h) 20–60 (kmol/h)
Ternary distillation column with recycle (*TCR*)	*RADFRAC*	Number of stages (n_{stages}) Distillation rate (*dist*) Intermediate distillation rate ($inter_{dist}$) Recycle ratio (*rr*)	5–25 70–110 (kmol/h) 20–60 (kmol/h) 0.5–0.95

TABLE 5 Sub-rewards for case study 2.

Sub-reward	Reward or penalty	Assigned	Description
Driving force	Reward	Every step	Quantified by the molar fraction difference between the current and previous steps, its magnitude reflects the UO's effectiveness in reducing the current stream's MeOH content.
UO total costs	Penalty	Every step	Composed of the capital and operational costs of the UO. The capital costs of each unit operation were calculated using functions derived from Douglas.[37]
Process design requirement	Penalty	Final step	Ensures that the desired purities of DME and water are met. Magnitude depends on the gap of the achieved purities and desired purities.
Specific design constraint	Reward	Specific step	Constant reward granted to the agent when the UO is a reactor and meets the design specification (operating below 400°C). No penalty is awarded if this constraint is not met.
Short flowsheets	Reward	Final step	Denoted as the difference between the maximum number of steps *I* (which for this case was 15) and *i*.
DME extra	Reward	Specific step	Granted to the agent upon achieving the target DME purity in the current step. The magnitude hinges on the distilled amount of pure DME, with a higher value for larger flows. Notably, this sub-reward is awarded only once, contingent on attaining the desired water purity for a complete flowsheet.

in processes that closely emulate actual design procedures. This study stands out from others employing ASPEN Plus® in the simulation environment by virtue of its incorporation of multiple types of UOs, distinguishing itself from works that solely report the utilization of a single UO.[24] Additionally, the proposed RL framework is capable of designing CPFs from an empty flowsheet, automatically connecting and simulating the selected UOs. To the authors' knowledge, this has not been attempted before; hence, the novelty of this work.

The goal of this case study is to formulate a CPF for the production of DME—obtained through the dehydration of methanol—that can achieve outflows with a purity of 99% for both the produced DME and water. The dehydration of methanol (Equation (6)) is a reversible reaction carried out in the presence of a catalyst, considering the catalyst efficiency, the temperature within the catalytic reactors employed in this process must be maintained below 400°C. In this case study, only one type of catalyst is being considered. The exploration for optimal catalyst materials falls outside the scope of this study but could be expanded within the current framework by incorporating reactors with different catalyst materials. This can potentially lead to the specification of a set of reactors, each with a different catalyst material and performance, for example, product yield or conversion.

$$2CH_3OH \rightleftharpoons C_2H_6O + H_2O. \tag{6}$$

For this case study, the selected thermodynamic package used in ASPEN Plus® was the Universal Quasichemical Activity Coefficient (UNIQUAC)/Redlich-Kwong equation of state with Henry's law (see Section S3). The UOs and their corresponding design variables are described in Table 4.

To simplify the analysis and prevent an excessive number of degrees of freedom, a few UO have some variables specified a priori. For instance, specifications for the reactor are the reaction that takes place (Equation (6)), heat transfer coefficients and the catalytic load, which is a function of the reactor's design. For the distillation columns, the feed stage is located at the midpoint of the tower. Additionally, for ternary columns, the intermediate flow outlet is established in the first third of the column.

In the RL domain, the hybrid action space consists of 10 discrete variables and 19 design actions—15 continuous variables and 4 discrete variables. Due to the large number of discrete actions and their corresponding continuous actions, to maintain a reasonably sized action space, each continuous variable was uniformly discretized into three realizations, resulting in a fully discretized action space of 170 actions. For each step i, the observation vector returned information about the outlet stream of the selected UO, that is, temperature (T), pressure ($Press$), the current step, and the molar fractions (χ) of methanol (MeOH), water (H_2O), and dimethyl ether (DME), that is, $o_i = [T, Press, \chi_{MeOH}, \chi_{H_2O}, \chi_{DME}, i]$. The reward function considered in this case study is similar to that used for the previous case study and follows a similar format for the sub-rewards, as shown in Table 5. As in the previous case study, all sub-rewards were normalized to prevent an individual term from having a disproportionate impact.

ALGORITHM 1 Pseudocode for the masking process, case study 2

```
done = False
stage = preparation
avail_units = []

while not done do
    if stage = preparation then
        avail_units = [M,P,HEX]
        if T ≥ 250° C and Press = 10 bar then
    stage = reaction
    end if
    elif stage = reaction then
        avail_units = [PFR,APFR]
        if conv_MeOH ≥ 0.75 then
    stage = cooling
    end if
    elif stage = cooling then
        avail_units = [C]
    stage = separation
    elif stage = separation then
        avail_units = [DC,DCR,TC,TCR]
        if purity_DME ≥ 0.99 and purity_Water ≥ 0.99 then
    done = True
    end if
    else
    Error
    done = True
    end if
end while
```

Unlike case study 1, the masking mechanism in this case study is composed of different process stages, as shown in Algorithm 1. Within each of these stages (see Algorithm 1), specific UOs are temporarily unmasked. Once the stage is completed, the UOs and the stage itself are then masked. All the required values to complete the stage and advance to the subsequent stage are derived from Luyben's[36] work and set to prevent ASPEN Plus® simulation errors. A more detailed description of each stage, and the derivation of intermediate stage constraints, is presented in Section S3.

Note that the masking technique does not fix or pre-establish any ordering or configuration of UOs; in fact, each time an episode starts, an empty flowsheet is initialized in the environment. This allows the agent to learn more effectively and avoid issues when simulating in ASPEN Plus multiple environments, that is, process flowsheets. Also, when an error happens during the simulation, the episode is terminated, and the agent is rewarded with a heavy penalty. Figure 6 presents the individual training curves for an unmasked and a masked hybrid PPO agent. Note that if masking were not applied, the agent would not be

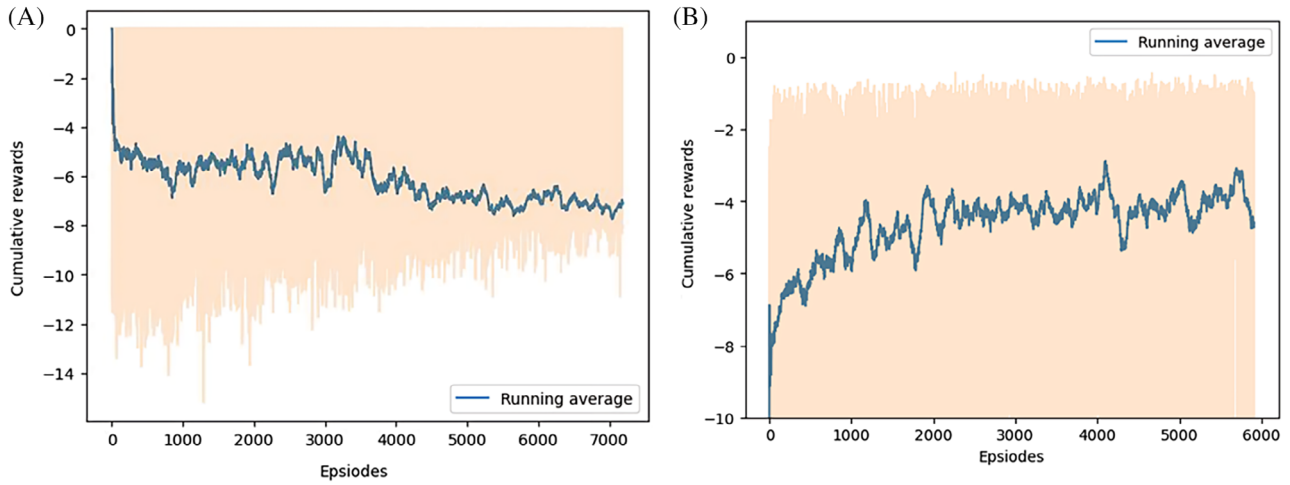

FIGURE 6 Individual training curve for HPPO: (A) unmasked; (B) masked.

able to meet the design specifications for this case study and exhibit a decreasing trend in its learning (Figure 6A).

During the agent's learning process, it was observed that it learned to reduce the penalty as little as possible enforcing the failure, reaching a point where it would choose the reactor without the specified conditions, resulting in single-step episodes and direct failures, thereby avoiding larger penalties.

All the hidden layers in the mPPO and mHPPO agents consisted of 64 neurons. Both agents were trained 10 times for 50,000 steps. The networks were updated every 512 steps with a discount factor of 0.99 and optimized using Adam[34] optimizer with a starting learning rate of 2.5e−4 which was adjusted throughout the training. The agents' exploration was encouraged by the inclusion of an entropy factor of 0.025, which progressively decayed at a rate of 0.9. These hyperparameters were obtained from preliminary testing with both agents.

Figure 7 presents the learning curves of both agents for 10 runs. The solid line represents the mean of the running means of the 10 runs, while the faded region represents the standard deviation around the different running means in those 10 runs. As in the previous case study, the mPPO achieves higher cumulative rewards with lower variability, whereas the mHPPO fluctuates around lower cumulative rewards with higher variability. Despite the mHPPO showing high variability in its learning (Figure 7A), when evaluating the mHPPO agents individually, the CPF obtained was the same. The only difference lie in the sizing of the UOs. The showcased variability arises from significant fluctuations in cumulative rewards during individual training sessions of mHPPO agent (Figure 6B). The source of these fluctuations arises from various factors. First, there is the inherent variability due to the implementation of continuous decision variables. Second, the interaction between continuous and discrete variables, that causes changes in the design and amount of UO present in the flowsheet, thus affecting the rewards. Third, the agent's difficulty in converging the flowsheet, particularly with units that can be considered as challenging. Operations such as distillation columns or reactors are often regarded as challenging units since the rigorous equations describing their behavior are highly non-linear models that are challenging to converge. Despite being well-specified, a discrepancy in the input flow or an imbalance in mass or energy balance can trigger warnings or convergence errors, resulting in simulation failures. These failures are attributed in the environment as significant penalties, thus prematurely ending episodes. On the other hand, the learning curve of the mPPO (Figure 7B) shows small to almost negligible variability in the last steps, suggesting that the 10 trained agents presented similar learning curves and converged to the same CPF. Note that the difference in episode lengths is precisely due to these variations in learning. While mHPPO exhibits extensive exploration, the length of the designed CPFs may not be constant and may vary from one episode to another. On the other hand, once the mPPO finds a flowsheet that meets all the requirements, the agent starts to exploit that arrangement of UOs, keeping the number of units of the CPF constant and only varying the equipment sizing and operating conditions.

When evaluating the best-performing agents, the designed CPFs shown in Figure 8 and Table 6 demonstrate the ability to achieve the desired DME and water purities, maintain the reactor's operating temperature below the specified threshold, and yield an economically attractive process flowsheet.

From the two CPF presented in Figure 8, it can be seen that the overall structure of both agents is similar. In fact, the first five UOs selected by both agents are the same, only varying in their sizing. The differences arise in the purification section, where the number and type of distillation columns differ. The mPPO opts for a *TCR*, where all the DME is distilled to the desired purity (i.e., 99%). Hence, the only task left for the agent is to distill the water using another column. Conversely, the mHPPO selects a series of columns that simultaneously distill both DME and water to their desired purities. The difference between the two agents lies not so much in the percentage of DME lost during the process, that is, the percentage is quite similar, as

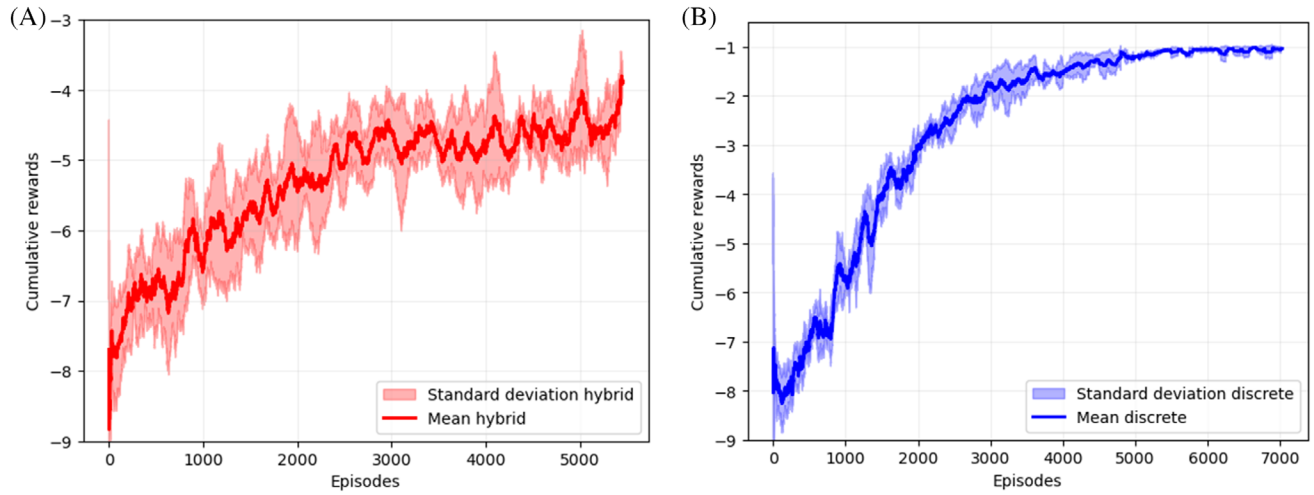

FIGURE 7 Average learning curve for 10 runs: (A) mHPPO; (B) mPPO.

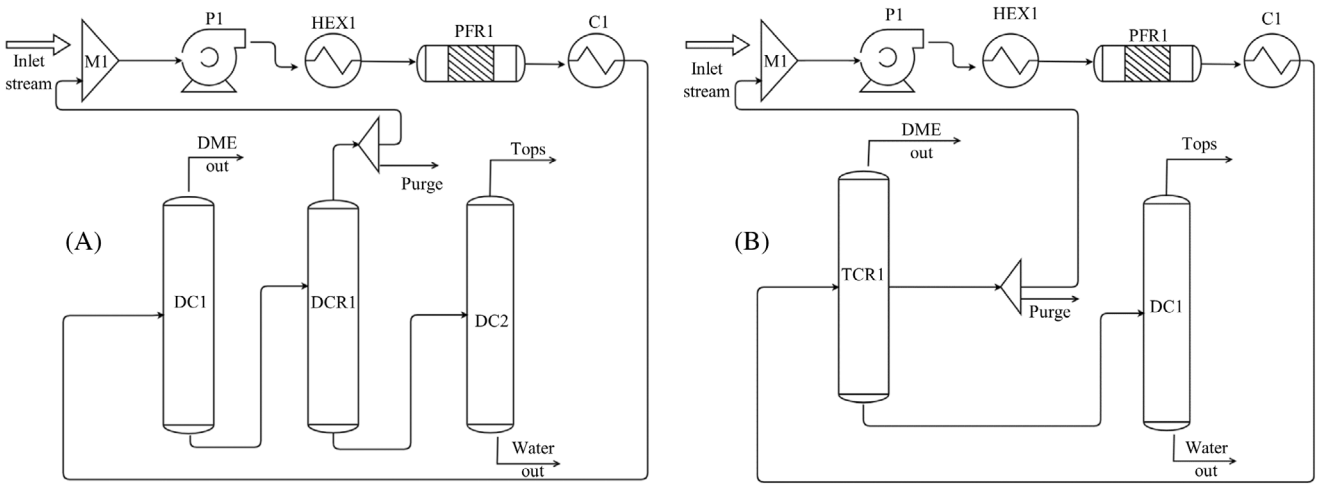

FIGURE 8 CPF obtained for case study 2: (A) mHPPO; (B) mPPO.

shown in Table 6, but in the amount of methanol (MeOH) recycled into the system. Despite achieving a lower cumulative reward, the mHPPO manages to reintegrate MeOH more effectively, allowing the agent to produce more DME than that obtained by the mPPO's CPF. For this case study, MeOH losses were not directly considered in the reward function, as the primary objective was to produce streams of DME and water that meet the desired specifications depicted in Table 6. However, its proper reintegration into the system is positively reflected in the sub-reward associated with the amount of DME distilled.

Note that a key insight achieved by the agents lies in their ability to acknowledge the need to include a recycle stream to improve the CPF. The recycle intricately connects various UOs and manages to yield a better outcome than without their consideration in the flowsheet. For instance, with the design specifications for the first distillation column, the mHPPO agent is unable to achieve the desired DME purity. Faced with this scenario, this agent can either mark the action as erroneous and adjust the specifications in the next episode, or, in the subsequent step, implement recycling to assess if it can meet the expected design conditions. If the recycling option is not chosen, the alternative is to decrease the magnitude of the distillate rate. Given that DME is the most volatile compound, this action ensures that the distillation column only contains this compound in the overheads. However, the downside of taking this action is that the agent loses some of the converted DME to other streams and also forfeits the opportunity to convert more DME from the MeOH that is not recirculated. Accordingly, recycling allows the agent to achieve the desired purity while maintaining the initially proposed distillate rate.

When comparing the flowsheets designed by the proposed framework with those presented in the works of Caballero et al.[35] or Luyben,[36] it is evident that the results are somewhat similar. Note that the overall design, economic, and operational objectives are the same;

TABLE 6 UOs designs and specifications for case study 2.

Unit operation	Design variable	mHPPO	mPPO
HEX1	T_{HEX} (°C)	277.2498	275.0000
PFR1	D (m)	2.0294	3.5000
	L (m)	9.2744	6.5000
C1	T_C (°C)	28.1489	50.0000
DC1	n_{stages}	21	20
	dist (kmol/h)	84.0929	60.0000
DCR1	n_{stages}	21	—
	dist (kmol/h)	39.7867	—
	rr	0.7295	—
DC2	n_{stages}	21	—
	dist (kmol/h)	40.1090	—
TCR1	n_{stages}	—	20
	dist (kmol/h)	—	80.0000
	$inter_{dist}$ (kmol/h)	—	60.0000
	rr	—	0.8500
Operating reactor temperature (°C)		329.1773	314.6445
Output DME purity		0.9998	0.9999
Output water purity		0.9999	0.9999
DME wasted/DME produced (%)		5.0911	4.5352
Lost methanol (kmol/h)		32.1410	39.8877
Cumulative reward		−2.0805	−1.1404

however, the approach used to arrive at the flowsheet design is different. Caballero et al.[35] considered a superstructure approach, while Luyben[36] fixed the arrangement of UOs, thus the goal was to size the units. In contrast, this study begins with an empty flowsheet and its goal is to propose the arrangement and design of UOs without any a priori knowledge of potential super-structures or CPF. Another point to consider is regarding the optimization methods; RL differs from conventional optimization techniques. Replicating the results reported in those studies using RL techniques may be quite challenging. As mentioned above, the agent aims to find a trade-off in its decisions, maximizing its impact on the flowsheet—specifically, obtaining 99%-pure water and DME streams—while minimizing operating costs. Translating results from traditional optimization methods to RL may not always seem the best option for the agent at a particular step and may even lead to their potential rejection. Note that the agent does not have access to the entire flowsheet at once but rather operation by operation, that is, step by step when building the flowsheet. RL offers advantages in CPF design over conventional methods, with similar results validating its effectiveness. Among its various advantages, RL demonstrates flexibility and adaptability in handling complex systems without prior knowledge. It also is capable of identifying new processing routes and the ability to generate CPFs with limited information, serving as a starting point for more advanced studies. Additionally, its flexibility to interact with various simulation environments (process simulators) and advanced dynamic exploration techniques is a key feature that becomes critical in uncertain design contexts where input variables may not be known a priori.

5 | CONCLUSIONS AND FUTURE WORK

This work introduced a new RL strategy for the generation and design of attractive CPFs. Within the RL framework, two novel agents featuring discrete (mPPO), and continuous and discrete (mHPPO) decisions were developed. The novelty of this approach is underscored by the integration of masking, an unexplored technique for CPF design. This method proves adept at discerning correlations among the constituent UOs. Throughout the study, the capability of both agents for the design and optimization of CPFs was assessed. The positive impact of the masking technique was noted in the deeper understanding of the correlation between units in the process and the enhanced decision-making of the proposed agents. Throughout this work, both approaches were compared leading to the conclusion that the superior performance of one over the other depends on the specific case study at hand. When the number of discrete and continuous variables is reduced, and the discretization of continuous variables is straightforward, mPPO demonstrated a significantly better performance than mHPPO. The rewards achieved by mPPO align with the maximum possible rewards, and due to the restricted search (combinatorial) space, it finds the optimal configuration possible. On the other hand, when the number of discrete and continuous variables is extensive, despite having an infinite search space, the mHPPO outperformed the mPPO. Even though yielding lower rewards, the generated CPFs not only meet all specified requirements but also exhibited novel arrangements to attain additional benefits, such as minimizing reactant losses or enhancing flow reintegration more effectively. An inherent limitation of the framework lies in the

extensive training time, particularly when the RL framework is connected to an external chemical simulation software such as ASPEN Plus® to simulate the environment. Training times could extend to hours or days under these conditions, posing a significant practical challenge and a area for future improvement; particularly for large and more complex case studies. To mitigate this issue, one potential solution could involve exploring more efficient methods to connect with chemical engineering software through an interphase communication link, that would potentially streamline the simulation process and enhance the overall computational efficiency of the framework.

As mentioned above, the CPF generated through this methodology provides results that are sufficiently robust to meet primary objectives and serve as a solid starting point when seeking to optimize a process flowsheet. Similarly, if dealing with a process whose flowsheet is unknown, this technique is ideal for initiating the outline of UOs' arrangements and gaining insights on potential (attractive) configurations. Once a RL-agent designs a CPF, this can be used as a starting point for the optimization process and refine its performance. The latter could be considered in future work—the integration of this methodology with conventional optimization methods. Additional avenues for extending this work involve exploring the agent's performance while considering uncertainty in the problem's specifications, accounting for human expert feedback in the RL framework, and consideration of process dynamics in the design of the CPF.

AUTHOR CONTRIBUTIONS

Simone Reynoso-Donzelli: Conceptualization; investigation; methodology; validation; visualization; writing – original draft; formal analysis. **Luis Alberto Ricardez-Sandoval:** Funding acquisition; writing – review and editing; validation; methodology; formal analysis; project administration; supervision; software; conceptualization.

ACKNOWLEDGMENTS

Financial support provided by Natural Sciences and Engineering Research Council of Canada (NSERC) is acknowledged.

DATA AVAILABILITY STATEMENT

All the raw data and the code for reproducing the results and images are distributed through our GitLab repository https://git.uwaterloo.ca/ricardez_group/flowsheet-desing. To replicate Case Study 1 results, navigate to the folder with the same name. For results presented in Figure 3 and Table 3, refer to the code titled "main.ipynb" in the subfolders Discrete and Hybrid, named after the respective agent. For Figure 4, refer to the code "comparison.ipynb." To replicate Figure S1, run the discrete agent from the Discrete subfolder, changing the number of discrete points in the size_dict found in "env.py." After that, run "main.ipynb." To replicate Case Study 2 results, refer to the folder with the same name. To replicate Table 6 results for each agent, run "main.ipynb" and use "test.py" found in the Discrete and Hybrid subfolders. Figure S6B is produced in the same manner with the hybrid agent. To replicate Figure 7, run "main.ipynb" 10 times with different seeds, store the values in an Excel file, and use "plots.ipynb" to obtain the images. For Figure S6A, navigate to the Hybrid subfolder of the Case Study 2 folder and modify "env.py" to allow all actions at all times, then run "main.ipynb."

ORCID

Luis Alberto Ricardez-Sandoval https://orcid.org/0000-0001-9867-6778

REFERENCES

1. Liñán DA, Contreras-Zarazúa G, Sánchez-Ramírez E, Segovia-Hernández JG, Ricardez-Sandoval LA. A hybrid deterministic-stochastic algorithm for the optimal design of process flowsheets with ordered discrete decisions. *Comput Chem Eng*. 2024;180:108501. doi:10.1016/j.compchemeng.2023.108501
2. d'Anterroches L, Gani R. Group contribution based process flowsheet synthesis, design and modelling. *Fluid Phase Equilib*. 2005;228-229:141-146. doi:10.1016/j.fluid.2004.08.018
3. Bertran MO, Frauzem R, Zhang L, Gani R. A generic methodology for superstructure optimization of different processing networks. In: Kravanja Z, Bogataj M, eds. *Computer Aided Chemical Engineering*. Elsevier; 2016:685-690. doi:10.1016/B978-0-444-63428-3.50119-3
4. Brüggemann S, Wolfgang M. Rapid screening of design alternatives for nonideal multiproduct distillation processes. *Comput Chem Eng*. 2004;29(1):165-179. doi:10.1016/j.compchemeng.2004.07.009
5. Grossmann IE, Aguirre PA, Barttfeld M. Optimal synthesis of complex distillation columns using rigorous models. *Comput Chem Eng*. 2005;29(6):1203-1215. doi:10.1016/j.compchemeng.2005.02.030
6. Ma Y, Yang Z, El-Khoruy A, et al. Simultaneous synthesis and design of reaction–separation–recycle processes using rigorous models. *Ind Eng Chem Res*. 2021;60(19):7275-7290. doi:10.1021/acs.iecr.1c00250
7. Liñán DA, Ricardez-Sandoval LA. A benders decomposition framework for the optimization of disjunctive superstructures with ordered discrete decisions. *AIChE J*. 2023;69(5):e18008. doi:10.1002/aic.18008
8. Bernal DE, Ovalle D, Liñán DA, Ricardez-Sandoval LA, Gómez JM, Grossmann IE. Process superstructure optimization through discrete steepest descent optimization: a GDP analysis and applications in process intensification. In: Yamashita Y, Kano M, eds. *Computer Aided Chemical Engineering*. Elsevier; 2022:1279-1284. doi:10.1016/B978-0-323-85159-6.50213-X
9. Mencarelli L, Chen Q, Pagot A, Grossmann IE. A review on superstructure optimization approaches in process system engineering. *Comput Chem Eng*. 2020;136:106808. doi:10.1016/j.compchemeng.2020.106808
10. Herrera Velázquez JJ, Zavala Durán FM, Chávez Díaz LA, Cabrera Ruiz J, Alcántara Avila JR. Hybrid two-step optimization of internally heat-integrated distillation columns. *J Taiwan Inst Chem Eng*. 2022;130:61. doi:10.1016/j.jtice.2021.06.061
11. Contreras-Zarazúa G, Vázquez-Castillo JA, Ramírez-Márquez C, Segovia-Hernández JG, Alcántara-Ávila JR. Multi-objective optimization involving cost and control properties in reactive distillation processes to produce diphenyl carbonate. *Comput Chem Eng*. 2017;105:185-196. doi:10.1016/j.compchemeng.2016.11.022
12. Contreras-Zarazúa G, Sánchez-Ramírez E, Vázquez-Castillo JA, et al. Inherently safer design and optimization of intensified separation processes for furfural production. *Ind Eng Chem Res*. 2019;58(15):6105-6120. doi:10.1021/acs.iecr.8b03646
13. Chanona EA d R, Petsagkourakis P, Bradford E, Graciano JEA, Chachuat B. Real-time optimization meets Bayesian optimization and derivative-free optimization: a tale of modifier adaptation. *Comput Chem Eng*. 2021;147:107249. doi:10.1016/j.compchemeng.2021.107249
14. Sutton RS, Barto A. *Reinforcement Learning: An Introduction*. The MIT Press; 2014.

15. Midgley LI. Deep reinforcement learning for process synthesis. arXiv: 2009.13265. 2020.
16. van Baten J, Baur R, Kooijman H. COCO (CAPE-OPEN to CAPE-OPEN). 2023. https://www.cocosimulator.org/index.html
17. Khan A, Lapkin A. Searching for optimal process routes: a reinforcement learning approach. *Comput Chem Eng*. 2020;141:107027. doi:10.1016/j.compchemeng.2020.107027
18. Stops L, Leenhouts R, Gao Q, Schweidtmann AM. Flowsheet synthesis through hierarchical reinforcement learning and graph neural networks. arXiv:2207.12051. 2022.
19. Gao Q, Yang H, Shanbhag SM, Schweidtmann AM. Transfer learning for process design with reinforcement learning. In: Kravanja Z, Bogataj M, eds. *Computer Aided Chemical Engineering*. Elsevier; 2023: 2005-2010. doi:10.1016/B978-0-443-15274-0.50319-X
20. Medeiros DW. DWSIM – Open Source Process Simulator. https://dwsim.org/
21. Göttl Q, Grimm DG, Burger J. Automated synthesis of steady-state continuous processes using reinforcement learning. *Front Chem Sci Eng*. 2022;16(2):288-302. doi:10.1007/s11705-021-2055-9
22. Göttl Q, Grimm DG, Burger J. Using reinforcement learning in a game-like setup for automated process synthesis without prior process knowledge. In: Yamashita Y, Kano M, eds. *Computer Aided Chemical Engineering. Vol 49. 14 International Symposium on Process Systems Engineering*. Elsevier; 2022:1555-1560. doi:10.1016/B978-0-323-85159-6.50259-1
23. Göttl Q, Pirnay J, Burger J, Grimm DG. Deep reinforcement learning enables conceptual design of processes for separating azeotropic mixtures without prior knowledge. 2024. doi:10.2139/ssrn.4776784
24. van Kalmthout SCPA, Midgley LI, Franke MB. Synthesis of separation processes with reinforcement learning. arXiv:2211.04327. 2022.
25. Aspen Technology Inc. ASPEN Plus. https://www.aspentech.com/en/
26. Vogel G, Schulze Balhorn L, Schweidtmann AM. Learning from flowsheets: a generative transformer model for autocompletion of flowsheets. *Comput Chem Eng*. 2023;171:108162. doi:10.1016/j.compchemeng.2023.108162
27. Balhorn LS, Caballero M, Schweidtmann AM. Toward autocorrection of chemical process flowsheets using large language models. arXiv: 2312.02873.
28. Mann V, Sales-Cruz M, Gani R, Venkatasubramanian V. eSFILES: intelligent process flowsheet synthesis using process knowledge, symbolic AI, and machine learning. *Comput Chem Eng*. 2024;181: 108505. doi:10.1016/j.compchemeng.2023.108505
29. Huang S, Ontañón S. A closer look at invalid action masking in policy gradient algorithms. *FLAIRS*. 2022;35:35. doi:10.32473/flairs.v35i.130584
30. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. arXiv:1707.06347. 2017.
31. Fan Z, Su R, Zhang W, Yu Y. Hybrid actor-critic reinforcement learning in parameterized action space. arXiv:1903.01344. 2019.
32. Wiewiora E. Reward shaping. In: Sammut C, Webb GI, eds. *Encyclopedia of Machine Learning*. Springer; 2010:863-865. doi:10.1007/978-0-387-30164-8_731
33. Schweiger CA, Floudas CA. Interaction of design and control: optimization with dynamic models. In: Hager WH, Pardalos PM, eds. *Optimal Control: Theory, Algorithms, and Applications. Applied Optimization*. Springer; 1998:388-435. doi:10.1007/978-1-4757-6095-8_19
34. Adam Cornell University Computational Optimization Open Textbook - Optimization Wiki. 2024. https://optimization.cbe.cornell.edu/index.php?title=Adam
35. Caballero J, Odjo A, Grossmann I. Flowsheet optimization with complex cost and size functions using process simulators. *AIChE J*. 2007; 53:2351-2366. doi:10.1002/aic.11262
36. Luyben WL. Improving the conventional reactor/separation/recycle DME process. *Comput Chem Eng*. 2017;106:17-22. doi:10.1016/j.compchemeng.2017.05.008
37. Douglas J. *Conceptual Design of Chemical Processes*. McGraw-Hill Book Comapny; 1988.

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: Reynoso-Donzelli S, Ricardez-Sandoval LA. A reinforcement learning approach with masked agents for chemical process flowsheet design. *AIChE J*. 2024;e18584. doi:10.1002/aic.18584