Available online at www.sciencedirect.com

ScienceDirect

Current Opinion in Chemical Engineering

Review

Deep reinforcement learning for process design: Review and perspective

Qinghe Gao and Artur M Schweidtmann

The transformation toward renewable energy and feedstock supply in the chemical industry requires new conceptual process design approaches. Recently, deep reinforcement learning (RL), a subclass of machine learning, has shown the potential to solve complex decision-making problems and aid sustainable process design. However, its suitability in static process design still needs to be examined. We discuss the advantages and disadvantages of RL for process design. Then, we survey state-of-the-art research through three major elements: (1) information representation, (2) agent architecture, and (3) environment and reward. Moreover, we discuss perspectives on underlying challenges and promising future works to unfold the full potential of RL for process design in chemical engineering.

Address
Delft University of Technology, Department of Chemical Engineering, Van der Maasweg 9, Delft 2629 HZ, the Netherlands

Corresponding author: Schweidtmann, Artur M (a.schweidtmann@tudelft.nl)

Current Opinion in Chemical Engineering 2024, **44**:101012

This review comes from a themed issue on **Model-based process design**

Edited by **Kai Sundmacher**

For complete overview of the section, please refer to the article collection, "Model-based Process Design"

Available online 14 March 2024

https://doi.org/10.1016/j.coche.2024.101012

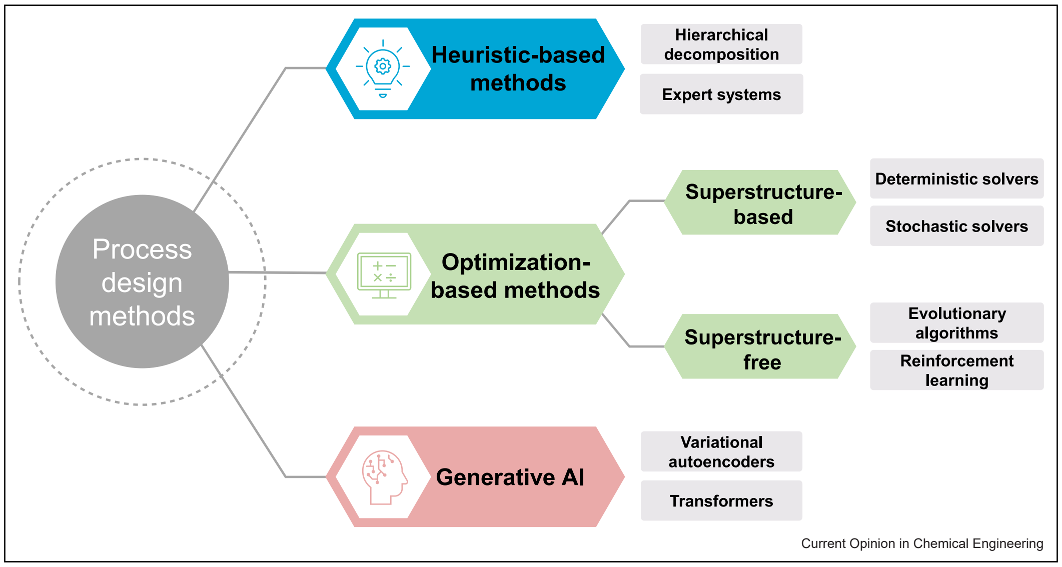
Introduction

The chemical industry is facing a rapid paradigm shift toward a circular economy based on renewable energy and feedstock supply [1,2]. This poses several challenges for conceptual process design due to the increasing complexity of the design task, the lack of experienced engineers, and the pressure on improving sustainability and profitability while shortening development times. Thus, there is a need for new methodologies and tools that support engineers to design sustainable processes in a more efficient way.

Computer-aid process design is widely used in process systems engineering (PSE) for conceptual process design [3,4], which can be classified into three methodologies as illustrated in Figure 1: (1) heuristic-based methods, (2) optimization-based methods, and (3) the emerging field of generative artificial intelligence (AI). Heuristic-based methods rely on a set of rules derived from experience, insights, and engineering knowledge, making them the most commonly used approaches due to their ease of application. Optimization-based approaches are commonly used to identify optimal design and operating variables for given process structures. Particularly, derivative-based optimization algorithms are already available in commercial process simulation software to determine those. To determine optimal process structures, superstructure-based methods are the *de facto* state of the art, where possible process alternatives are modeled and subsequently solved by mixed-integer nonlinear optimization (MINLP) methods [5,6]. Within this paradigm, two solver types are typically utilized: deterministic (e.g. branch and bound algorithms such as Branch-And-Reduce Optimization Navigator or McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization) and stochastic solvers (e.g. genetic algorithms). While superstructure methods have been very successful in PSE, they also have many shortcomings that limit industrial applications [7], including (manual) setup of all process alternatives, the need to implement process models in an optimization environment, and the difficulties of solving resulting MINLPs. Superstructure-free methods dispense the need for predefined superstructures. For example, evolutionary algorithms typically adopt a two-level decomposition approach [6]. First, they generate alternative flowsheets, which are then evaluated through an optimization algorithm. Finally, a few very recent works proposed the use of generative AI methods for the generation of process structures [8,9]. However, those works require large training data and are not the focus of this review.

Recently, deep reinforcement learning (RL) has shown its potential to solve complex sequential dynamic decision-making problems at human-like or even superhuman level [10–12]. RL is a computational approach for

Figure 1



Overview of computer-aided process design methods.

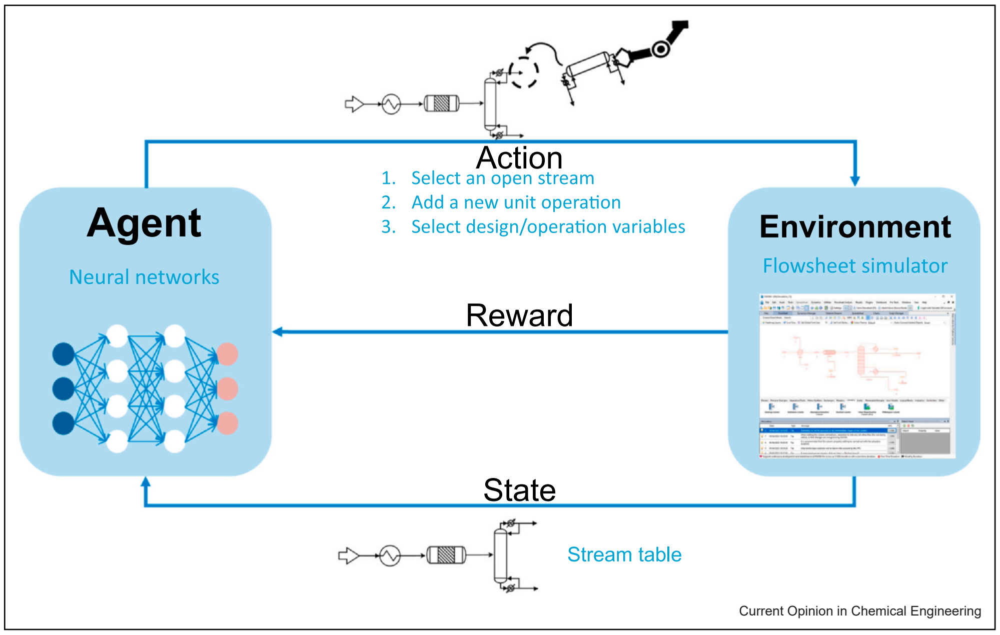
goal-directed learning and decision-making through the direct interaction of an agent with its environment [13]. RL is primarily developed to solve discrete-time dynamic optimization problems formulated as Markov decision processes (MDP). Consequently, RL is based on the Bellman optimality equation, which is similar to the Hamilton–Jacobi–Bellman equation and Pontryagin's maximum principle for continuous state and action spaces in the control theory. Also, RL has seen its first applications in chemical engineering for process control [14,15] and scheduling [16,17]. Notably, Yokogawa is even using RL to operate an industrial chemical process since 2022.

This review and perspective paper aims to provide a critical examination of the application of RL for process design. In the context of process design, RL can be considered a superstructure-free and model-free method, which iteratively places unit operations with corresponding design and operating variables. It evaluates the resulting flowsheets at every iteration and aims to maximize the objective functions. In the recent literature, there have been a few first steps toward applying RL as a static optimization algorithm for stationary process design, including absorption-stripping process [18], energy systems design [19,20], unit operation design [21•], separation process [22••,23–29••], solvent extraction process design [30], single mixed-refrigerant process design [31], and synthesis reaction process design [32–34].

The use of RL for stationary process design is controversial in scientific discussions. Most notably, RL is better suited for dynamic, sequential-decision problems rather than static ones. Also, clear comparisons and benchmarks between RL and other design methods are lacking in previous literature. Therefore, at the moment, the suitability of RL for process design is questionable and remains an open question. Here, we present the main differences between RL and existing superstructure-free methods in the context of process design:

- Computational efficiency (static vs dynamic optimization) — RL is better suited for dynamic, sequential-decision problems rather than static ones. However, prior studies have applied RL to stationary process design, constituting static optimization problems. This may drastically increase complexity and decrease computational efficiency (called 'sample efficiency' in RL). As a side note, the capability of RL in dynamic optimization paves the way for solving integrated design and operation problems (see section *Integrated process control operation and design*).
- Iterative build-up vs full flowsheet generation — RL sequentially generates flowsheets (unit by unit), differing from evolutionary superstructure-free methods that construct flowsheets as a whole. While this sequential build-up increases computational complexity, there are also potential advantages. Generating feasible flowsheets and simulating them is challenging, often causing convergence problems in evolutionary superstructure-free methods. In contrast, the iterative strategy of RL promotes convergence, and intermediate simulations of incomplete flowsheets may provide valuable information for learning. However, this additional information comes at the cost of additional simulation time.

Figure 2



General framework of RL for process design.

- Inference (online) vs optimization (offline) — A significant distinction between RL and other optimization methods lies in the solution time for similar, recurring problems. RL involves training a policy once, which is then utilized during inference to rapidly predict near-optimal solutions, offering a significant advantage in time-sensitive control applications. In contrast, classical optimization algorithms solve problems individually, typically requiring long runtimes for each problem instance. In process design, long optimization times are usually not an issue (unless it becomes intractable in the case of large nonconvex MINLPs). Thus, classical optimization methods are usually well suited. However, the rapid inference capability of RL may also provide new opportunities for process design. For example, RL agents might be integrated into flowsheet simulation software to automatically and immediately suggest near-optimal design options to users. Also, fast solutions of design problems can be advantageous when a large number of design problems need to be solved (e.g. as subproblems in larger optimization studies).
- Learning capacity — RL possesses a substantially greater learning capacity compared to standard evolutionary methods (e.g. more trainable parameters). For instance, state-of-the-art deep RL algorithms can incorporate extensive networks of learnable parameters, potentially exceeding billions of parameters. This enhanced learning potential facilitates inference and allows for retaining information, unlike genetic algorithms, which typically lose details about previous populations. This substantial learning capacity of RL holds the potential for learning more complex dependencies between design actions and results. However, the high learning capacity also presents severe drawbacks, such as a vast amount of training data and extensive training duration (measured in epochs within RL).

When comparing RL with standard optimization methods, its most notable advantages include larger learning capacity and inference ability. However, at the same time, RL typically demands significantly more training simulations than static optimization solvers, which leaves doubt on whether its potential advantages outweigh the disadvantages in the context of process design. Current literature applying RL to process design neglects its inference capabilities, learning capacity, and dynamic optimization capabilities, predominantly utilizing the training phase of RL as an evolutionary optimization strategy for static problems. Thereby, they essentially merge the drawbacks of both worlds. Additionally, there is a lack of computational comparison between RL and traditional process design methods. In the following, we critically examine the existing literature on RL in process design (section *State of the art*) and highlight future perspectives (section *Perspectives*).

State of the art

The general framework of RL in process design is shown in Figure 2. The agent learns to design processes by iteratively placing unit operations with design and operating variables and simulating the resulting processes in the environment, ultimately obtaining the optimal policy π^*, which designs optimal processes. Mathematically, this problem can be formulated as MDP: $M = \{S, A, T, R\}$ with states $\mathbf{s} \in S$, actions $\mathbf{a} \in A$, the transition function T: $S \times A \times S \rightarrow [0,1]$, and the reward function R: $S \times A \times S \rightarrow \mathbb{R}$. In the context of process design, the states **s** represent the flowsheet topology as well as all relevant design specifications, operating variables, thermodynamic stream data, flowrates, and compositions. The agent takes the current states **s** as input to take actions **a**. These actions can include design and operating variables. In chemical processes, design variables are usually determined during the initial design phase and typically remain fixed throughout the operation, such as equipment size. Operating variables can be adjusted during operation (e.g. flow rates and pressures). Furthermore, the actions contain discrete choices (e.g. the selection of open streams, unit operation types or number of stages) as well as continuous choices (e.g. the length of a reactor or operating flowrates), namely, hybrid action space. Usually, decisions in process design are also hierarchical. For example, the agent first determines an open stream to add a unit operation, then the type of unit operation, then design variables, and, finally, operating variables. After a new unit operation is added, the new flowsheet is simulated in an environment (e.g. a process simulation software). After finishing a flowsheet, a numerical reward $\mathbb{R}$ is returned to the agent. This corresponds to the objective for optimization. By repeating the design of multiple flowsheets and receiving corresponding rewards, the agent learns to design processes that maximize the reward.

In this section, we survey the RL state-of-the-art literature (summarized in Table 1) based on (1) information representation, (2) agent architecture, and (3) environment and reward.

Information representation

Chemical processes comprise various information, such as process topology, thermodynamic states, flowrates, concentrations, design variables, operating variables, components, and underlying mechanistic knowledge. The meaningful representation of the chemical process information is critical for the learning and generalization of RL agents. For RL in process design, there are currently two methods for information representations: Matrix [18–32] and graph [33••,34].

In matrix-based representation, flowsheets are represented by fixed-size matrices. Within the flowsheet matrix, the connectivity, stream compositions, thermodynamic stream data, and design variables are usually concatenated. For example, Göttl et al. [24–26] represented flowsheets as 16×28 matrices, where each row represents a stream and encompasses four parts: $\{\mathbf{v}, \mathbf{u}, \mathbf{d}, \mathbf{t}\}$. $\mathbf{v}_i$ has five entries, which describe the molar fractions and total molar flowrate of stream i. $\mathbf{u}_i$ stores the type of the unit operation that is downstream of stream i as one-hot encoding. Furthermore, $\mathbf{d}_i$ stores the connectivity of unit operations and has 16 entries (i.e. this corresponds to the adjacency matrix). Finally, $\mathbf{t}_i$ has two entries: the first entry indicates whether the task is terminated (0 if not terminated), and the second entry indicates whether stream i is still unused (0 if unused). Most of the previous publications use matrix representations of flowsheet states (c.f. Table 1).

Table 1

An overview of the reviewed literature and different choices of elements in RL for process design. We use Repr. to indicate information representation. Furthermore, we utilize Dis., Cont., and Hier. to denote discrete, continuous, and hierarchical, respectively, action space of RL. Within the decisions of RL, we use Topo., Des., and Oper. to represent actions involved in changing flowsheet topologies, selecting design variables, and operating variables, respectively.

Ref.	Repr.	Agent architecture	Environment	Action space			Decisions		
				Dis.	Cont.	Hier.	Topo.	Des.	Oper.
[22••]	Matrix	AC (SAC)	COCO	✓	✓		✓	✓	✓
[23•]	Matrix	AC (SAC)	Aspen Plus	✓	✓			✓	✓
[18]	Matrix	AC (–)	Aspen Plus		✓				✓
[30]	Matrix	AC (PPO)	Short-cut		✓			✓	
[24,25••,26]	Matrix	AC (two players)	Short-cut	✓		✓	✓		
[28,29••]	Matrix	AC (PPO)	Short-cut	✓	✓	✓	✓	✓	✓
[20]	Matrix	AC (ACER)	Short-cut	✓				✓	
[19]	Matrix	Policy based (policy search)	Short-cut		✓				✓
[21•]	Matrix	Policy-based (PG)	Short-cut		✓			✓	✓
[32]	Matrix	Value-based (Deep Q-Network)	IDAES	✓			✓		
[31]	Matrix	Value-based (Deep Q-Network)	UniSim		✓			✓	✓
[27]	Matrix	Value-based (Q-learning)	Short-cut	✓	✓		✓	✓	
[33••]	Graph	AC (PPO)	Short-cut	✓	✓	✓	✓	✓	✓
[34]	Graph	AC (PPO)	DWSIM	✓	✓	✓	✓	✓	✓

In graph-based representation, flowsheets are represented by directed homogeneous or heterogeneous graphs. Flowsheet graphs consist of nodes and edges. Unit operations are represented by nodes, also referred to as vertices $v \in V$, and streams are represented by edges $e_{vw} \in E$ connecting two nodes v and w. Importantly, node feature vector $\mathbf{f}^{v} \in F^{V}$ and edge feature vector $\mathbf{f}^{e_{vw}} \in F^{E}$ are associated with each node and edge, respectively. Within the node feature vectors, types of unit operation, design specifications, and operating points are encoded. The edge feature vectors contain thermodynamic states, concentrations, and flowrates. In the past, only our previous works used graph representations of flowsheets for RL [33••,34].

The comparison between flowsheet matrices and flowsheet graphs is still an open research question in the context of RL in process design. Flowsheet matrices are easier to implement than flowsheet graphs and are used by the majority of the literature as presented in Table 1. Flowsheet matrices are processed by RL agents using multilayer perceptrons (MLPs) or convolutional neural networks (CNNs). However, every flowsheet graph has $N!$ different adjacency matrices. CNNs and MLPs are not permutation equivariance

$$f(\mathbf{P}^{T}\mathbf{x}\mathbf{P}) \neq \mathbf{P}^{T} f(\mathbf{x})\mathbf{P}$$

where $\mathbf{P}$ is a permutation matrix, $\mathbf{x}$ is the input matrix, and f is an MLP or CNN [35]. This means that such models depend on the arbitrary order of rows/columns in the flowsheet matrix and thus cannot generalize over flowsheet topologies. Also, the neighborhood of an entry in the matrix does not correspond to physical connectivity, which makes learning using MLPs/CNNs more difficult as it requires learning long-range interactions. In contrast, (message passing) graph convolutional networks (GCNs) are permutation equivariance, and they learn from the actual connectivity of flowsheet graphs. Furthermore, MLPs and CNNs require fixed-size inputs, for example, a predefined maximum number of unit operations and streams, while GCNs are size independent [36].

Agent architecture

RL agents consist of two components: a policy and a learning algorithm. The policy describes the behavior of the agent, mapping the current state $\mathbf{s}$ into an action $\mathbf{a}$: $\pi(\mathbf{s}) = \mathbf{a}$. It is parameterized by function approximators, such as MLPs. The learning algorithm is used to continuously update the policy based on the actions, states, and rewards. Depending on the learning algorithms, the agent can be characterized into three types: value based, policy based, and actor-critic (AC) based.

Value-based agents learn a functional approximator of the value function ($V_{\pi}(\mathbf{s})$) to take actions. The value function outputs the expected returns after the current process step t given a state $\mathbf{s}$ and a policy π: $V_{\pi}(\mathrm{s}) = \mathbb{E}_{\pi}[G_t|\mathrm{s}]$, where returns G_t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{k} R_{t+k+1} = \sum_{k=0}^{\infty} \gamma^{k} R_{t+k+1} \tag{1}$$

where γ^{k} are the discount rates to determine the present value of future rewards, and k is the process step from t to the end of the episode. Similarly, we can also derive the state-action value function, namely, the quality function Q_{π}, which calculates the expected returns given a state $\mathbf{s}$ and action $\mathbf{a}$, following policy π: $Q_{\pi}(\mathrm{s}, \mathbf{a}) = \mathbb{E}_{\pi}[G_t|\mathrm{s}, \mathbf{a}]$. Depending on the calculated V-/Q-value, different search algorithms such as best-first search or nearest neighbors are used to choose the final action. In the context of RL in process design, three works [27,31,32] deployed Q-learning–based agent to perform process synthesis tasks. However, traditional value-based agents can only take discrete actions, which hinders further development because continuous decision-making of design or operating variables is vital in process design tasks.

Policy-based agents directly learn a functional approximator of the policy function. Specifically, the policy approximator π_{θ} maps the current states $\mathbf{s}$ to the actions $\mathbf{a}$: $\pi_{\theta}(\mathbf{s}) = \mathbf{a}$. And the optimal policy π^{*} is obtained by maximizing the expected return $\mathbb{E}_{\theta}[G_t]$ through policy gradient or policy search methods. In the context of RL in process design, Sachio et al. [21•] and Perera et al. [19] utilized policy gradient (PG) methods and policy search methods to perform process design tasks, respectively. Compared with the value-based approach, the policy-based agent can handle both discrete and continuous actions. However, policy-based methods are known for high variance and suboptimal local solutions [37].

AC agents combine the advantages of value-based and policy-based methods. AC consists of an actor, working as a functional approximator of the policy function, and a critic, serving as a functional approximator of the value function. Therefore, AC agents explicitly optimize both value and policy functions and are able to process both discrete and continuous action spaces. In the context of RL in process design, different types of AC agents have been used, such as Proximal Policy Optimization (PPO) [29••,30,33••,34], Soft Actor-Critic (SAC) [22••,23•], two-player game [24–26], and Sample Efficient Actor Critic with Prioritized Experience Replay (ACER) [20].

The choice of agent architecture for RL in process design is an open question. AC RL is deemed to be a viable option because it combines the advantages of value based and policy based and can handle both discrete and continuous decisions. Specifically, PPO is the most popular algorithm in process design tasks, with the advantage of less complicated implementation and a stable learning process. However, PPO is an on-policy algorithm, which means the optimized policy is the same as the policy for action selection. Therefore, PPO is less data efficient than off-policy algorithms, such as SAC and ACER, which may take less time and fewer training episodes. Moreover, AC RL comes with challenges, including complex implementation, computational demands when optimizing both actor and critic networks concurrently, and potential convergence issues [13,37].

Environment and reward

The environment simulates the processes and computes a reward as feedback to the agent. Selecting an appropriate accuracy level for the environment is a vital task for RL in process design and depends on the task-specific requirements and modeling intent. There are two main levels of accuracy: Shortcut and rigorous simulators. Shortcut simulators utilize approximated process models to ensure tractability but can be inaccurate. Rigorous simulators involve more accurate process models that require longer computation times, as previous studies indicated [23•,34]. In the past, RL for process design has used multiple process simulation software, including open-source (DWSIM, IDAES), noncommercial (COCO), and commercial (UniSim, Aspen Plus) alternatives. Additionally, Seidenberg et al. [29••] leveraged knowledge graphs to retrieve information about the design task, process knowledge, and the current state of the process. Notably, this knowledge graph was part of a manually implemented environment and not directly accessible to the RL agent. Thus, the agent also relied on a flowsheet matrix representation as states.

Previous research optimized toward a single economic objective [18,20–27••,29••,33••,34]. Also, some works integrate purity, recovery, power consumption, and product flow rate into scalar reward functions [30–32].

Perspectives

Despite the first demonstrations of RL for process design, it is still unclear if RL outperforms existing design methods. In our view, the big research challenge is the generalization of RL models and the use of its inference capabilities. The training phase of current RL frameworks is essentially used like a derivative-free optimization approach (e.g. a genetic algorithm) to optimize the process topology for one particular case study. Thus, a retraining is needed for a new case study, and the agent fails to transfer its learning to new situations. In general, deep RL has an inference and a high learning capacity. The derivative-free optimization approach with RL does not use the full potential of RL. However, useful application of inference requires generalization across multiple case studies. This generalization requires an extension of the information representation and agent architecture to account for process-relevant knowledge. This includes domain expertise, prior process data, and physical constraints, which are typically employed by engineers when designing chemical processes. Integrating this information would allow the RL agent to see what 'drives the process' and ultimately unlock the full potential of RL by learning from multiple processes. We envision that RL will generalize (to some extent) and ultimately design processes at inference time. In this section, we provide our perspectives on underlying challenges and a number of other promising future works.

Information representation

Information representation is critical for RL since it encapsulates the current state of the environment, which directly affects decision-making for agents. However, current information representations still lack mechanistic knowledge and relevant process information. Furthermore, it neglects the appropriate representation of molecules. Integrating the above information will significantly benefit the RL agent in generalization. Numerous representation methods could be potentially incorporated into RL in process design for process and molecular information representation. For example, Simplified Flowsheet Input-Line Entry-System (SFILES) [38], SFILES 2.0 [39], eSFILES [40], and knowledge graphs [29••] could be used to enhance process representations. Similarly, molecular descriptors [41], molecular graphs [42], SMILES [43], knowledge graphs [44], and hypergraphs [45] could be used to encode molecular information.

Agent architecture

In this section, we identify the limitations and potential improvements of the current agent architecture.

Integration of mechanistic knowledge

Current RL algorithms are not sufficient to transfer knowledge into the new processes because RL agents have a limited understanding of mechanistic knowledge and physical properties. Future work could consider implementing a physics-informed RL agent by encoding information-rich representations, such as knowledge graphs or hypergraphs, to inform the agent. Furthermore, fundamental concepts, such as thermodynamic driving forces (Gibb's free energy), could be included in the RL agents. This allows the agent to learn general concepts that can be translated into other problems because they are based on physics.

Integration of prior data

RL for process design is currently initialized randomly, which can lead to suboptimal solutions, excessive training times, and frequent convergence issues. Meanwhile, there is a large number of existing digitized chemical process data from simulation files and images

[46], which can potentially accelerate the learning process of the agent. Transfer learning improves learning performance by transferring knowledge from different but relevant domains [47]. In RL for process design, three works [21•,32,34] already leveraged transfer learning to accelerate the learning process, for example, from short-cut simulators to rigorous simulators [34] and from one case study to another case study [32]. However, in the current transfer learning setting, the agent is still not learning from existing chemical process information. Future work can consider leveraging encoder–decoder models such as variational autoencoders or transformers to learn from existing flowsheets and then applying transfer learning to the agent.

Stochastic decision-making

Considering the uncertainty of energy/feedstock prices and demand is a major challenge for renewable processes [7]. However, current RL agents ignore fluctuations in energy/feedstock prices and demand. Future research could separate design and operating variables in the RL agent. This allows the inclusion of multiple scenarios for flexible operation. Besides, additional encoders or actors can be included to process stochastic energy prices, demands, and raw material compositions as additional inputs at an operational level. Therefore, the agent can automatically select the operating variables based on stochastic energy prices and demand in two-stage stochastic programming settings.

Constrainted decision-making

Constrained decision-making is crucial for RL in process design to ensure optimal and safe performance. However, standard RL agent frameworks cannot enforce constraints but include constraints as 'soft' penalties in the reward functions [23•,33••,34]. Future work should focus on integrating constraints directly in the agent structure. As a first step, an additional critic network could be built to account for safety constraints, guiding RL agents to explore appropriate regions in policy optimization [48].

Environment and reward

In this section, we offer our perspectives on the limitations of the environment and reward setup and provide several suggestions for future work.

Standardized simulation interfaces

RL agents frequently interact with process simulators during the training process, and the interaction relies on individual interfaces as Table 1 presents. However, current interfaces are usually simulator specific, which means that a new interface needs to be implemented from scratch whenever a new process simulator is included. This process is highly repetitive and inefficient, especially for incorporating multifidelity process models. Future work could implement a standardized simulation interface that enables the agent to exchange data efficiently and uniformly between different process simulators. This interface could potentially make use of existing standards such as CAPE-OPEN [49] and DEXPI+ [50].

Multifidelity process models

Current research only leverages a single-fidelity model for RL in process design tasks. However, RL agents greatly benefit from pretraining on low-fidelity process simulators and subsequent fine-tuning on high-fidelity process simulators [34]. Therefore, future research can focus on developing an agent that can dynamically select between multiple-fidelity models during training. Specifically, a probabilistic model can be developed to guide the RL actor based on multifidelity critics to reduce training times and resolve convergence issues.

Multiobjective rewards

Current RL frameworks for process design are not suitable for sustainable process design because they are limited to a single objective function. In the future, the current agent structures could be extended to include multiple objectives. For example, the critic network could predict multiple rewards, which will be processed by multiobjective optimization to generate corresponding weights for each objective (e.g. economic, environmental, safety) [51].

Integrated molecular and process design

Current RL frameworks lack the co-design of molecules. However, the design or selection of molecules is a critical task in many process design tasks, for example, co-design of working fluids, solvents, or products [52].

Also, RL has already been used for molecular design [53]. Therefore, future work should consider integrating these concepts using RL. For instance, future work could first use an RL agent for molecular design (e.g. based on Ref. [53]) to design a solvent. Then, a mechanistic or data-driven model [42] can be used to estimate the relevant properties of the generated molecules. Subsequently, the properties are utilized to simulate the process within the RL for the process design framework. This essentially adds a new hierarchy level to the existing RL for the process design framework.

Integrated process operation and design

Integrating process design and process control becomes increasingly relevant as renewable energy and feedstock demand fluctuates. For example, the design of a process could be optimized while simultaneously optimizing its operation under changing feedstock compositions or energy prices. Another example is the optimal design of batch processes while considering optimal operation strategies. These problems are usually formulated as mixed-integer dynamic optimization problems, which are difficult to solve. RL is a tool to solve discrete-time dynamic optimization problems, which makes it suitable for integrating process operation and design. However, current RL works only consider process design and operation separately or focus on a specific unit operation design and operation [21•]. Future research could integrate process design with process operation through the RL. For instance, future work could extend the hierarchical RL agent into separate design and operation agents. Then, the design agent defines the design variables, and the operation agent subsequently optimizes operating variables given the current design. Notably, this would require the use of a dynamic simulation environment and will lead to high computational demands. Thus, future research is needed to solve the resulting multiscale problem efficiently.

Benchmarking with established methods

Numerous established methods are available to solve design optimization problems, including deterministic (e.g. Branch-And-Reduce Optimization Navigator or McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization) and stochastic solvers (e.g. genetic algorithms, Bayesian optimization) [54]. It is still questionable how RL compares against these traditional approaches for steady-state process design. Dynamic solution approaches such as RL can in principle be used to solve static optimization problems but are likely significantly less efficient. However, many process design problems in chemical engineering are actually (mixed integer) dynamic optimization problems (c.f. section *Integrated process operation and design*). In such instances, RL may be an efficient solution approach.

Future work should carefully assess the advantages and disadvantages of using RL for steady-state process design and static optimization in general. We recommend conducting comparisons to benchmark different methods. Moreover, we envision the development of new ML-based algorithms that integrate some of the advantages of RL (e.g. large learning capacity and inference capabilities) in the context of process design. For example, encoder–decoder models could be combined with active learning to predict process flowsheet graphs directly [8].

Conclusions

We reviewed the state-of-the-art RL in process design in terms of information representation, agent architecture, environment, and reward. RL has shown initial promising results for process design, but its suitability in static process design still needs to be examined. Additionally, a detailed comparison with existing process design methods is missing, and current RL frameworks show limited generalization capabilities. Therefore, we advocate that future research should benchmark RL with other process design methods. Additionally, to unlock the full potential of RL, new concepts for meaningful information representation are required. Furthermore, the integration of mechanistic knowledge, existing process data, uncertainties, and constraints would be highly beneficial for optimal decision-making. Finally, future RL frameworks could also integrate molecular design and process operation into the conceptual process design.

Data Availability

The authors are unable or have chosen not to specify which data have been used.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References and recommended reading

Papers of particular interest, published within the period of review, have been highlighted as:

- • of special interest
- •• of outstanding interest

1. Meramo-Hurtado SI, González-Delgado AD: **Process synthesis, analysis, and optimization methodologies toward chemical process sustainability**. *Ind Eng Chem Res* 2021, **60**:4193-4217.
2. Martinez-Hernandez E: **Trends in sustainable process design — from molecular to global scales**. *Curr Opin Chem Eng* 2017, **17**:35-41.
3. Umeda T: **Computer aided process synthesis**. *Comput Chem Eng* 1983, **7**:279-309.
4. Dimian AC, Bildea CS: **Chemical Process Design: Computer-Aided Case Studies**. John Wiley & Sons; 2008.
5. Yee TF, Grossmann IE: **Simultaneous optimization models for heat integration—II. Heat exchanger network synthesis**. *Comput Chem Eng* 1990, **14**:1165-1184, https://doi.org/10.1016/0098-1354(90)85010-8
6. Mencarelli L, *et al.*: **A review on superstructure optimization approaches in process system engineering**. *Comput Chem Eng* 2020, **136**:106808, https://doi.org/10.1016/j.compchemeng.2020.106808
7. Mitsos A, *et al.*: **Challenges in process optimization for new feedstocks and energy sources**. *Comput Chem Eng* 2018, **113**:209-221, https://doi.org/10.1016/j.compchemeng.2018.03.013
8. E Hirtreiter, LS Balhorn, and AM Schweidtmann: Toward Automatic Generation of Control Structures for Process Flow Diagrams With Large Language Models; AIChE Journal. 2023. 10.1002/aic.18259.
9. Nabil T, Commenge J-M, Neveux T: **Generative approaches for the synthesis of process structures**. Computer Aided Chemical Engineering. Elsevier; 2022:289-294.
10. V. Mnih et al. : Playing Atari With Deep Reinforcement Learning; 2013. 10.48550/ARXIV.1312.5602.
11. M. Kempka et al. : ViZDoom: A Doom-Based AI Research Platform for Visual Reinforcement Learning; In: *2016 IEEE Conference on*

Computational Intelligence and Games (CIG) . IEEE. 2016. 10.1109/cig.2016.7860433.

12. Silver D, *et al.*: **A general reinforcement learning algorithm that masters chess, shogi, and go through self-play**. *Science* 2018, **362**:1140-1144, https://doi.org/10.1126/science.aar6404

13. Sutton RS, Barto AG: **Reinforcement Learning: An Introduction**. MIT Press; 2018.

14. Hoskins J, Himmelblau D: **Process control via artificial neural networks and reinforcement learning**. *Comput Chem Eng* 1992, **16**:241-251, https://doi.org/10.1016/0098-1354(92)80045-b

15. SPK Spielberg, R Gopaluni, and P Loewen: Deep Reinforcement Learning Approaches for Process Control; In: *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*. IEEE. 2017. 10.1109/adconip.2017.7983780.

16. Hubbs CD, *et al.*: **A deep reinforcement learning approach for chemical production scheduling**. *Comput Chem Eng* 2020, **141**:106982, https://doi.org/10.1016/j.compchemeng.2020.106982

17. Lee YH, Lee S: **Deep reinforcement learning based scheduling within production plan in semiconductor fabrication**. *Expert Syst Appl* 2022, **191**:116222, https://doi.org/10.1016/j.eswa.2021.116222

18. Chen J, Wang F: **Cost reduction of CO_2 capture processes using reinforcement learning based iterative design: a pilot-scale absorption–stripping system**. *Sep Purif Technol* 2014, **122**:149-158, https://doi.org/10.1016/j.seppur.2013.10.023

19. Perera ATD, *et al.*: **Introducing reinforcement learning to the energy system design process**. *Appl Energy* 2020, **262**:114580, https://doi.org/10.1016/j.apenergy.2020.114580

20. Caputo C, *et al.*: **Design and planning of flexible mobile micro-grids using deep reinforcement learning**. *Appl Energy* 2023, **335**:120707, https://doi.org/10.1016/j.apenergy.2023.120707

21. Sachio S, *et al.*: **Integrating process design and control using**
• **reinforcement learning**. *Chem Eng Res Des* 2022, **183**:160-169, https://doi.org/10.1016/j.cherd.2021.10.032.
This work integrates process control and specification of design variables for predefined unit operations using RL.

22. LI Midgley: Deep Reinforcement Learning for Process Synthesis;
•• 2020. 10.48550/ARXIV.2009.13265. arXiv: 2009.13265 [cs.LG]..
First work leveraging hybrid action space for separation process design.

23. SCPA van Kalmthout, LI Midgley, and MB Franke: Synthesis of
• Separation Processes With Reinforcement Learning; 2022. 10.48550/ARXIV.2211.04327. arXiv: 2211.04327 [cs.LG]..
This work critically discusses the challenges of rigorous simulation environments for RL.

24. Göttl Q, Grimm DG, Burger J: **Automated synthesis of steady-state continuous processes using reinforcement learning**. *Front Chem Sci Eng* 2021, **16**:288-302, https://doi.org/10.1007/s11705-021-2055-9

25. Göttl Q, *et al.*: **Automated flowsheet synthesis using hierarchical**
•• **reinforcement learning: proof of concept**. *Chem Ing Tech* 2021, **93**:2010-2018, https://doi.org/10.1002/cite.202100086.
The first work incorporating a hierarchical agent architecture into RL for process design.

26. Göttl Q, Grimm DG, Burger J: **Using reinforcement learning in a game-like setup for automated process synthesis without prior process knowledge**. Computer Aided Chemical Engineering. Elsevier; 2022:1555-1560, https://doi.org/10.1016/b978-0-323-85159-6.50259-1

27. Khan A, Lapkin A: **Searching for optimal process routes: a reinforcement learning approach**. *Comput Chem Eng* 2020, **141**:107027, https://doi.org/10.1016/j.compchemeng.2020.107027

28. Khan AA, Lapkin AA: **Designing the process designer: hierarchical reinforcement learning for optimisation-based process design**. *Chem Eng Process Process Intensif* 2022, **180**:108885, https://doi.org/10.1016/j.cep.2022.108885

29. Seidenberg JR, Khan AA, Lapkin AA: **Boosting autonomous**
•• **process design and intensification with formalized domain knowledge**. *Comput Chem Eng* 2023, **169**:108097, https://doi.org/10.1016/j.compchemeng.2022.108097.
The first work proposing a knowledge graph to integrate prior knowledge into the RL environment.

30. Plathottam SJ, *et al.*: **Solvent extraction process design using deep reinforcement learning**. *J Adv Manuf Process* 2021, **3**:e10079, https://doi.org/10.1002/amp2.10079

31. Kim S, Jang M-G, Kim J-K: **Process design and optimization of single mixed-refrigerant processes with the application of deep reinforcement learning**. *Appl Therm Eng* 2023, **223**:120038, https://doi.org/10.1016/j.applthermaleng.2023.120038

32. D Wang et al., Reinforcement Learning for Automated Conceptual Design of Advanced Energy and Chemical Systems; 2022. 10.21203/rs.3.rs-2248780/v1.

33. Stops L, *et al.*: **Flowsheet generation through hierarchical**
•• **reinforcement learning and graph neural networks**. *AIChE J* 2022, **69**:e17938, https://doi.org/10.1002/aic.17938.
The first work proposing graph representation for flowsheets and graph neural networks for RL.

34. Gao Q, *et al.*: **Transfer learning for process design with reinforcement learning**. Computer Aided Chemical Engineering. Elsevier; 2023:2005-2010, https://doi.org/10.1016/b978-0-443-15274-0.50319-x

35. Hamilton WL: **Graph Representation Learning**. Springer International Publishing; 2020, https://doi.org/10.1007/978-3-031-01588-5

36. Zhou J, *et al.*: **Graph neural networks: a review of methods and applications**. *AI Open* 2020, **1**:57-81, https://doi.org/10.1016/j.aiopen.2021.01.001

37. Nachum O, *et al.*: **Bridging the gap between value and policy based reinforcement learning**. In *Advances in Neural Information Processing Systems*. Edited by Guyon I, *et al.*. 30 Curran Associates, Inc.; 2017, ⟨https://proceedings.neurips.cc/paper_files/paper/2017/file/facf9f743b083008a894eee7baa16469-Paper.pdf⟩.

38. L. d′Anterroches: Process Flowsheet Generation & Design through a Group Contribution Approach; [CAPEC], Department of Chemical Engineering, Technical University of Denmark. 2005.

39. Vogel G, *et al.*: **SFILES 2.0: an extended text-based flowsheet representation**. *Optim Eng* 2023, **24**:2911-2933, https://doi.org/10.1007/s11081-023-09798-9

40. Mann V, Gani R, Venkatasubramanian V: **Intelligent process flowsheet synthesis and design using extended SFILES representation**. Computer Aided Chemical Engineering. Elsevier; 2023:221-226, https://doi.org/10.1016/b978-0-443-15274-0.50036-6

41. Todeschini R, Consonni V: **Molecular descriptors**. *Recent Adv QSAR Stud* 2010, **8**:29-102.

42. Schweidtmann AM, *et al.*: **Graph neural networks for prediction of fuel ignition quality**. *Energy Fuels* 2020, **34**:11395-11407, https://doi.org/10.1021/acs.energyfuels.0c01533

43. Weininger D: **SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules**. *J Chem Inf Comput Sci* 1988, **28**:31-36, https://doi.org/10.1021/ci00057a005

44. Fang Y, *et al.*: **Molecular contrastive learning with chemical element knowledge graph**. *Proc AAAI Conf Artif Intell* 2022, **36**:3968-3976, https://doi.org/10.1609/aaai.v36i4.20313

45. H. Kajino: Molecular Hypergraph Grammar With Its Application to Molecular Optimization; 2018. 10.48550/ARXIV.1809.02745. arXiv: 1809.02745 [cs.LG].

46. Balhorn LS, *et al.*: **Flowsheet recognition using deep convolutional neural networks**. Computer Aided Chemical Engineering. Elsevier; 2022:1567-1572, https://doi.org/10.1016/b978-0-323-85159-6.50261-x

47. Weiss K, Khoshgoftaar TM, Wang D: **A survey of transfer learning**. *J Big Data* 2016, **3**:1-40, https://doi.org/10.1186/s40537-016-0043-6

48. Yang Q, *et al.*: **Safety-constrained reinforcement learning with a distributional safety critic**. *Mach Learn* 2022, **112**:859-887, https://doi.org/10.1007/s10994-022-06187-8

49. M. Jarke et al. : CAPE-OPEN: Experiences From a Standardization Effort in Chemical Industries; In: *Proc. of 1st IEEE Conference on Standardisation and Innovation in Information Technology* (SIIT 99) (Aachen, Germany). 1999. 25–35.

50. M. Theissen and M. Wiedau: DEXPI P&ID specification, Version 0. 11; 2016.

51. Liu C, Xu X, Hu D: **Multiobjective reinforcement learning: a comprehensive overview**. *IEEE Trans Syst Man Cyber Syst* 2015, **45**:385-398, https://doi.org/10.1109/tsmc.2014.2358639

52. Rehner P, Schilling J, Bardow A: **Molecule superstructures for computer-aided molecular and process design**. *Mol Syst Des Eng* 2023, **8**:488-499, https://doi.org/10.1039/d2me00230b

53. Olivecrona M, *et al.*: **Molecular de-novo design through deep reinforcement learning**. *J Chemin* 2017, **9**:1-14, https://doi.org/10.1186/s13321-017-0235-x

54. Boukouvala F, Misener R, Floudas CA: **Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO**. *Eur J Oper Res* 2016, **252**:701-727, https://doi.org/10.1016/j.ejor.2015.12.018