Delft University of Technology

Accelerating process synthesis with reinforcement learning
Transfer learning from multi-fidelity simulations and variational autoencoders

Gao, Qinghe; Yang, Haoyu; Theisen, Maximilian F.; Schweidtmann, Artur M.

DOI
10.1016/j.compchemeng.2025.109192

Publication date
2025

Document Version
Final published version

Published in
Computers and Chemical Engineering

Citation (APA)
Gao, Q., Yang, H., Theisen, M. F., & Schweidtmann, A. M. (2025). Accelerating process synthesis with reinforcement learning: Transfer learning from multi-fidelity simulations and variational autoencoders. *Computers and Chemical Engineering*, *201*, Article 109192. https://doi.org/10.1016/j.compchemeng.2025.109192

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Computers and Chemical Engineering 201 (2025) 109192

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/cace

Accelerating process synthesis with reinforcement learning: Transfer learning from multi-fidelity simulations and variational autoencoders

Qinghe Gao, Haoyu Yang, Maximilian F. Theisen, Artur M. Schweidtmann *

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

ARTICLE INFO

Keywords:
Reinforcement learning
Process synthesis
Transfer learning

ABSTRACT

Reinforcement learning has shown some success in automating process design by integrating data-driven models that interact with process simulators to learn to build process flowsheets iteratively. However, one major challenge in the learning process is that the reinforcement learning agent demands numerous process simulations in rigorous process simulators, thereby requiring long simulation times and expensive computational power. We propose employing transfer learning to enhance the reinforcement learning process in process design. This study examines two transfer learning strategies: (i) transferring knowledge from shortcut process simulators to rigorous simulators, and (ii) transferring knowledge from process variational autoencoders (VAEs). Our findings reveal that appropriate transfer learning can significantly improve both learning efficiency and convergence scores. However, transfer learning can also negatively impact the learning process when there are substantial discrepancies in decision range and reward function. This suggests that pre-trained process data should match the complexity of the fine-tuning task.

1. Introduction

Conceptual process design is a complex task executed by engineers. Classical conceptual process design methods are either manual works that demand long development times or superstructure methods that are limited to the pre-defined superstructures and difficult to solve the resulting mixed-integer nonlinear optimization problems (MINLPs) (Mitsos et al., 2018). Additionally, artificial intelligence (AI) has also been leveraged in conceptual process design. Notably, a number of expert systems (Siirola and Rudd, 1971; Mahalec and Motard, 1977) were proposed in the first wave of AI where experts integrate domain experience and facts into a knowledge base to perform reasoning for design problems. However, those expert systems were difficult to maintain and extend, which hindered further developments in conceptual process design.

Recently, reinforcement learning (RL), a subclass of machine learning (ML), has demonstrated excellent performance in solving complicated sequential decision-making problems such as gaming (Mnih et al., 2013; Silver et al., 2018), process control (Nian et al., 2020), and molecular design (Olivecrona et al., 2017). RL is a computational approach in which an agent iteratively interacts with its environment to sequentially make decisions and achieve its objectives (Sutton and Barto). Several initial first steps have been made towards applying RL in conceptual process design, as summarized in our recent review paper (Gao and Schweidtmann, 2024). Perera et al. (2020) first utilized an RL agent as a supporter for suggesting dispatch strategies for energy systems design based on the pre-defined structure. Khan and Lapkin (2020) utilized RL to search the optimal flowsheet with a hydrogen production process case study, where a value-based agent was used to estimate the value for the next processing step based on the intimidate flowsheet. Furthermore, Midgley (2020) leveraged a soft-actor–critic RL agent to design a separation process for non-azeotropic mixtures. In their approach, the agent first chooses whether to add a new distillation column and then chooses the corresponding continuous operation variables, and the flowsheet is simulated in the open-source process simulator COCO. Additionally, Göttl et al. (2021) represented a full flowsheet in a matrix for two competing players as state space and proposed hierarchical reinforcement learning (HRL) to design more advanced processes including recycles. HRL is an approach that decomposes a complex task into simpler subtasks. For example, in the context of process design, 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. Khan and Lapkin (2022) also implemented HRL where a high-level agent chooses a sub-objective of the process section and a low-level agent chooses unit types and corresponding control variables within the section. Moreover, Plathottam et al. (2021) introduce RL in a solvent extraction process design by

* Corresponding author.
E-mail address: a.schweidtmann@tudelft.nl (A.M. Schweidtmann).

https://doi.org/10.1016/j.compchemeng.2025.109192
Received 28 January 2025; Received in revised form 18 April 2025; Accepted 12 May 2025
Available online 31 May 2025
0098-1354/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

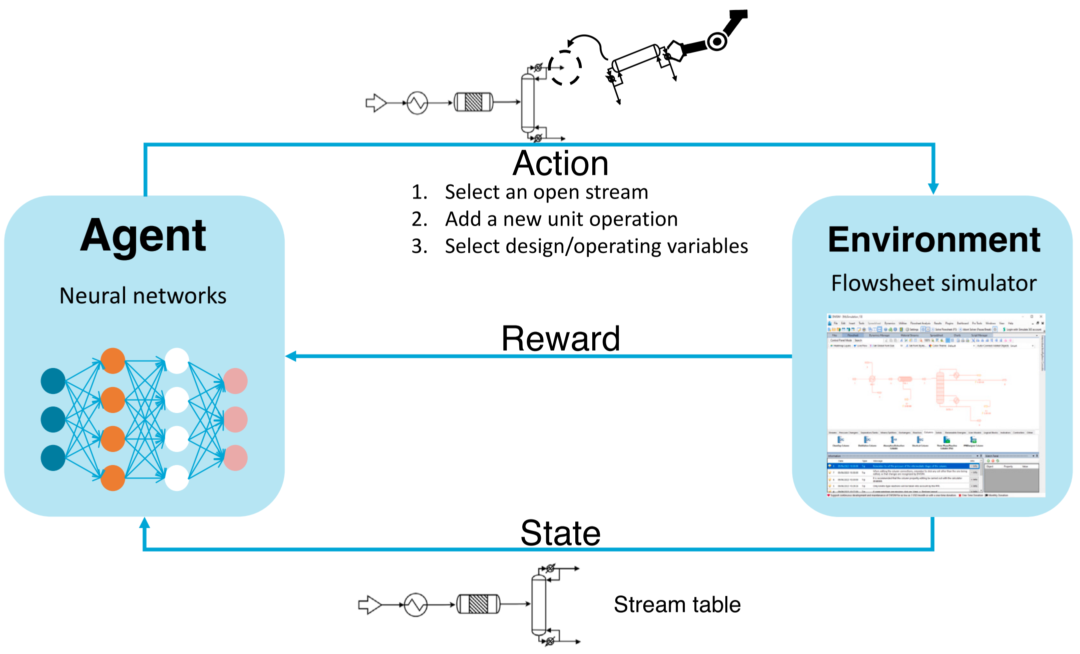

Fig. 1. General framework of RL in process design.

optimizing design variables for predefined flowsheets. In a recent publication, Stops et al. (2022) introduced a graph-based representation of the process, which more naturally captures the flowsheet topologies, and utilized the HRL agent to design processes. Reynoso-Donzelli and Ricardez-Sandoval (2024a,b) further implemented a masked RL agent which allows the exclusion of infeasible or irrelevant actions based on process constraints, ensuring that the RL agent focuses only on viable process decisions. Furthermore, several studies deployed hybrid approaches within the RL framework. For example, Tan et al. (2024) combined RL with mathematical programming to perform the design of heat exchanger networks. Also, Göttl et al. (2024) utilized RL for the discrete decisions alongside a genetic algorithm for the continuous decisions for hybrid action spaces, which is illustrated by two real-world use-cases: Synthesis of chemical process flowsheets and design of optical multi-layer films.

Although there are some successful applications for RL in process design, one major remaining challenge is that the RL training process is trial-and-error-based. Thereby, the learning process typically requires a large number of process simulations, which demands expensive computational power and long simulation times. Therefore, to mitigate this problem, most works previously deployed shortcut process simulation methods to efficiently simulate the process, which may potentially lead to inaccurate results. Recently, van Kalmthout et al. (2022) leveraged a rigorous process simulator (Aspen Plus) to perform RL in process design tasks. Recently, Göttl et al. (2025) demonstrated a generalized RL framework that can adapt to different chemical systems during the inference time without retraining, which significantly reduces the computational burden associated with RL-based process design. However, the long simulation time and convergence issue in the training phase still hinder further advances.

Transfer learning has been used to accelerate the learning process in many domains such as computer vision (Hussain et al., 2018), and natural language processing (Ruder et al., 2019). Transfer learning is a method that transfers knowledge from different but relevant domains to the target domain, thereby improving the learning performance (Zhuang et al., 2021). In the context of RL in process design, Wang et al. (2022) first trained the RL agent with one case study and then transferred the RL agent to another case study to demonstrate the generalizability of the RL agent. With their following work, Tian et al. (2024) further applied RL to a hydrodealkylation example. Additionally, our previous work (Gao et al., 2023) demonstrates that pretraining RL agents on the shortcut process simulation method and then fine-tuning the transferred agents in the rigorous simulator can not only accelerate the learning process but also improve the convergence score. Therefore, transfer learning has huge potential to mitigate long simulation time problems with rigorous simulators for RL in process design. However, there exist many possible strategies to perform transfer learning. Comparing them and identifying promising ones is still an open research question. Also, the effectiveness of transfer learning is not always guaranteed and it has been observed that transfer learning can bring negative contributions to the fine-tuning task, namely negative transfer (Zhang et al., 2023). The effect has not yet been discussed in the context of RL for process design.

We propose and investigate two transfer learning strategies: (i) transfer learning from shortcut simulation methods to a rigorous process simulator and (ii) transfer learning from generative AI models. In the first strategy, we first pre-train the RL agent with shortcut simulation methods and then further fine-tune it with a rigorous process simulator, DWSIM. Specifically, we explore different transfer methods within the RL agent architecture and apply these to an illustrative case study comprising equilibrium reactions, azeotropic separation, and recycles. In the second strategy, we utilize the encoder of a variational autoencoder (VAE), which was pre-trained on flowsheet data, and then further fine-tuned with the rigorous process simulator DWSIM.

2. RL for process synthesis

A RL problem can be formulated as a Markov decision process (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 $R : S \times A \times S \rightarrow \mathbb{R}$. Fig. 1 shows a framework in the context of RL in process design. The agent learns to design processes by iteratively placing unit operations together with corresponding design/operating variables, simulating the process in the environment, and maximizing the long-term rewards, thereby obtaining optimal processes. Specifically, the states $\mathbf{s}$ represent the information including flowsheet topology as well as design variables, operating variables, thermodynamic stream data, flowrates, and compositions. The agent, typically consisting of neural networks, takes the current states $\mathbf{s}$ as input and based on that takes actions

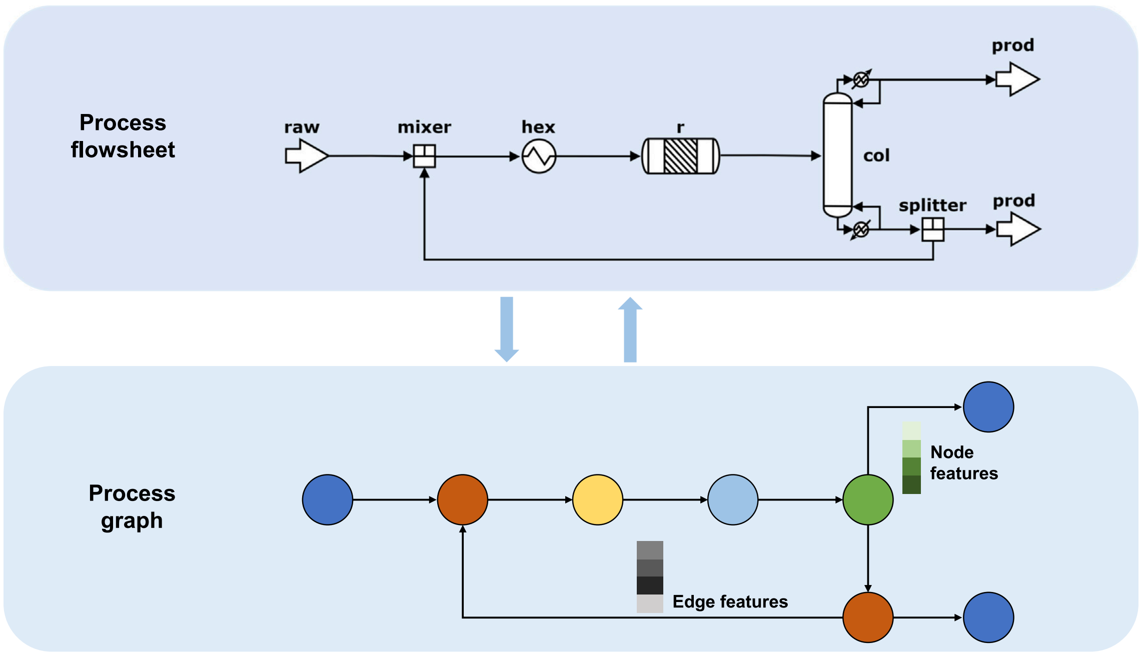

Fig. 2. One example of process graph. Feeds, unit operations, and products are represented as nodes which are associated with node features. Streams are represented as edges which are also associated with corresponding edge features.

a. Notably, the action space can be hybrid and hierarchical. A hybrid action space means the actions include discrete choices (e.g., the selection of open streams, or unit operation types) and continuous choices (e.g., the length of a reactor, or operating temperature). A hierarchical action space indicates that the agent can make decisions sequentially. 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. In chemical processes, the design variables are usually determined during the initial design phase and typically remain fixed throughout the operation. While operating variables can be adjusted during operation. After a new unit operation is added, the new flowsheet is simulated in an environment, which is typically a process simulation software. Subsequently, the environment sends a numerical reward $\mathbb{R}$ to the agent, which is the corresponding objective for optimization such as net present value. Ultimately, the agent learns to design optimal flowsheets by repeating the whole steps and maximizing the cumulative rewards.

2.1. *Information representation*

Chemical processes contain various information such as topology, design variables, operating variables, thermodynamic stream data, flowrates, and compositions. Presenting this information in a meaningful way is critical for the learning process of the RL agent. In this work, we present the flowsheets as directed homogeneous graphs as Fig. 2 shows. Within the process graph, the feeds, unit operations, and products are denoted as nodes, namely vertices $v \in V$, and streams are denoted by directed edges $e_{vw} \in E$ linking two nodes v and w. Notably, each node and edge is associated with one node feature vector $\mathbf{f}^{v} \in F^{V}$ and edge feature vector $\mathbf{f}^{e_{vw}} \in F^{E}$, respectively. The node feature vector typically contains types of unit operations, corresponding design variables, and operating variables. The edge feature vector includes thermodynamic stream data, compositions, and flowrates. Additionally, in the uncompleted flowsheet, open streams are linked to “undefined” nodes because open edges are prohibited in graph representations. These undefined nodes serve as temporary placeholders for future unit operations. Essentially, incorporating a new unit operation involves substituting an “undefined” node with a specific type of unit operation node.

2.2. *Agent architecture*

In this section, we introduce the proposed agent architecture, as Fig. 3 shows. The architecture is actor–critic based and consists of three parts: A graph encoder, hierarchical actor networks, and critic networks. First, a graph neural network (GNN) is utilized to encode flowsheet graphs into a flowsheet fingerprint. The actor then takes actions at three levels: (1) select an open stream, (2) select a unit operation and (3) select a design/operating variable. The critic networks evaluate the value of the flowsheet fingerprint to optimize the reward functions. In Section 2.2.1, Section 2.2.2, and Section 2.2.3, we will introduce the graph encoder, hierarchical actor networks, and the critic network, respectively.

2.2.1. *Graph encoder*

The flowsheet graphs are processed by a GNN, as illustrated in Fig. 4. The architecture of the proposed GNN consists of two primary phases: message passing and readout. During the message passing phase, graph convolutional networks (GCNs) are utilized to update the node embeddings for the flowsheet graph. Specifically, Fig. 5 illustrates the message passing step for a node represented in red. Initially, the message function $\mathbf{m}$ combines the information from neighboring nodes (colored yellow) and their connecting edges into a message. Subsequently, this message is utilized to update the feature of the targeted node using an update function $\mathbf{u}$, such as the sum function. This update process is iteratively applied to each node in every layer $l \in L$ within a GCN, allowing nodes to cumulatively gather information from their neighbors up to a distance determined by the number of layers L. Afterward, in the readout phase, the structure information learned during the message passing phase for each node is aggregated into a flowsheet fingerprint by a pooling function.

2.2.2. *Hierarchical actor networks*

The architecture of actor networks is primarily shaped by a hierarchical and hybrid action space, as illustrated in Fig. 6. This framework encompasses three decision-making levels: (i) selecting an open stream, (ii) choosing a specific unit operation, and (iii) identifying a design or operating variable. The initial two levels involve making discrete choices, whereas the final level is continuous, which together are

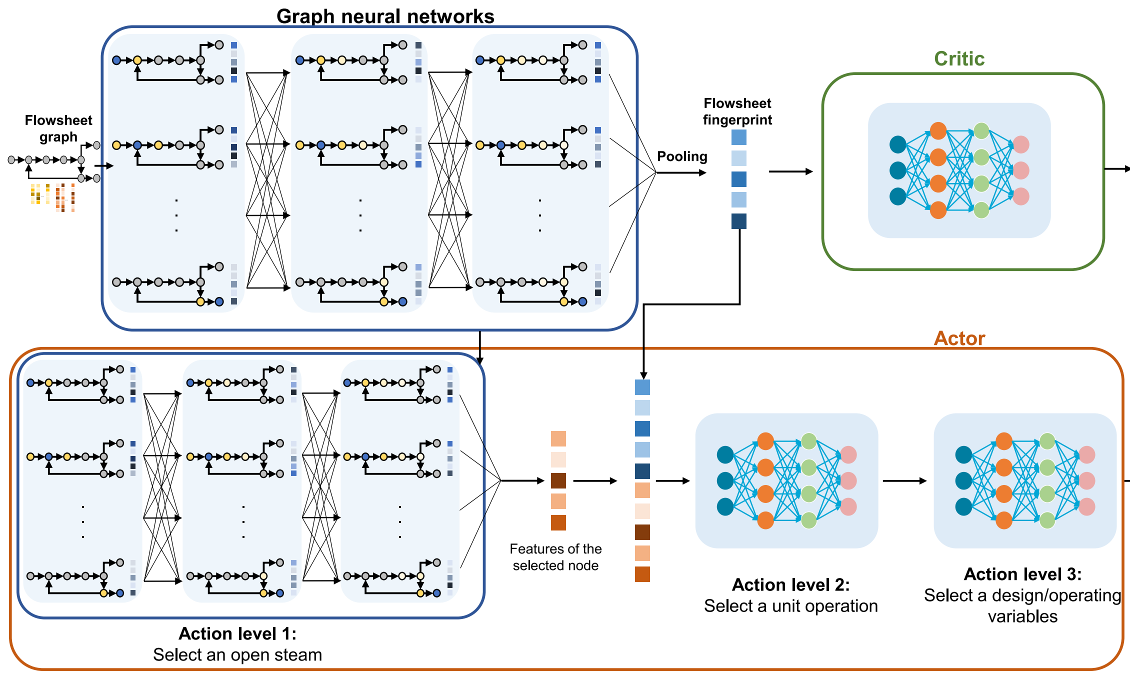

Fig. 3. The example agent architecture.

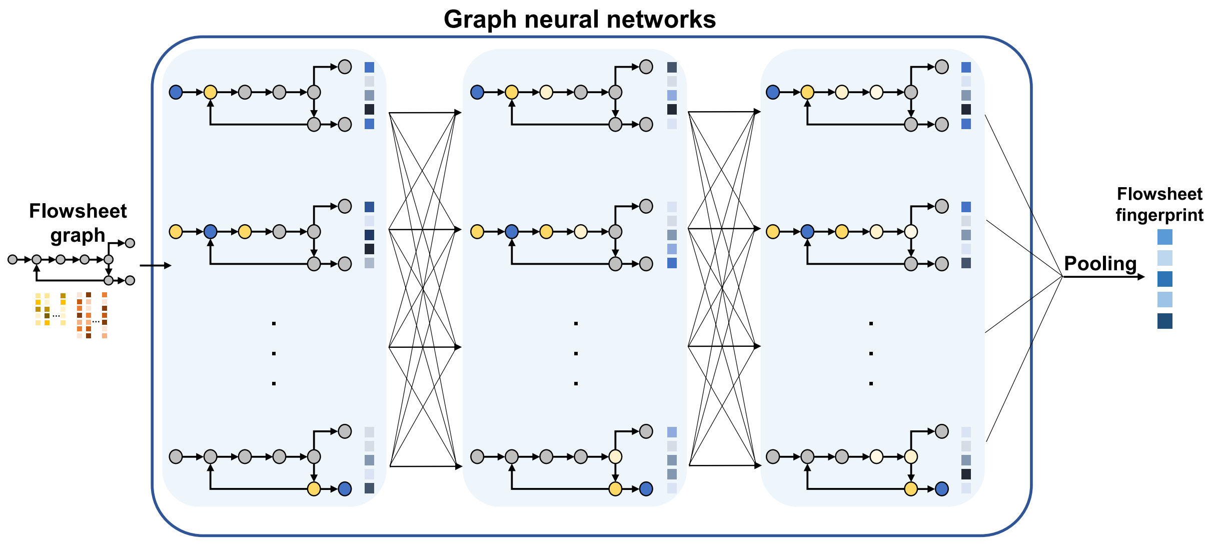

Fig. 4. The example architecture of the graph encoder, which is adapted from Schweidtmann et al. (2020).

referred to as a hybrid action space. In the first level, the agent selects an open stream which marks the location for subsequent expansion within the flowsheet.

In the second level, the agent selects the type of unit operation to be added, which could be a heat exchanger, a reactor, a distillation column, or even a recycling process through the introduction of an additional splitter and mixer. This step also includes the option of choosing an open stream as a product, effectively terminating the expansion for that particular pathway. The decisions in the third level are directly tied to those in the previous level. For instance, upon the addition of a reactor, the agent is tasked with determining a specific design variable, such as the reactor length. For simplicity, only one variable is selected for each unit operation. It is important to note that within the current implementation, the recycle stream can be integrated solely into the feed stream.

The implementation of actor networks echoes the hierarchical and hybrid action space, as depicted in Fig. 3. In action level 1, the updated graph (before the readout phase) in the graph encoder is further processed by two additional GCN layers. Within these two GCN layers, all node features are compressed to a single dimension, similar to the approach used in node classification tasks. Furthermore, we apply a zero mask vector to effectively eliminate the selection probability of all nodes, with the exception of those categorized as “undefined.” This ensures that nodes corresponding to feed and unit operations are excluded from selection. This selected node feature is concatenated with the flowsheet fingerprint, forming the input for subsequent action levels. In action level 2, a Multilayer Perceptron (MLP) calculates probabilities for each possible unit operation. For action level 3, each unit operation is associated with its own MLP, which predicts two parameters, α and β, for a beta distribution $B(\alpha, \beta)$ A continuous decision regarding the respective design/operating variable is then made based on this beta distribution.

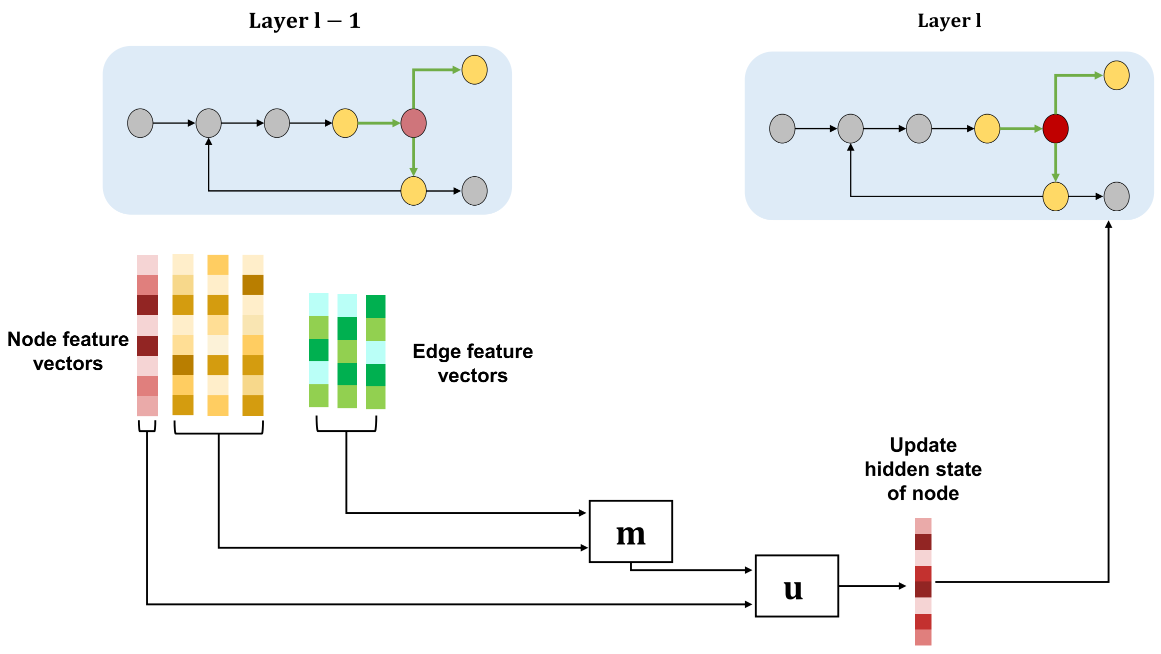

Fig. 5. An overview of node message passing in graph convolution layers.

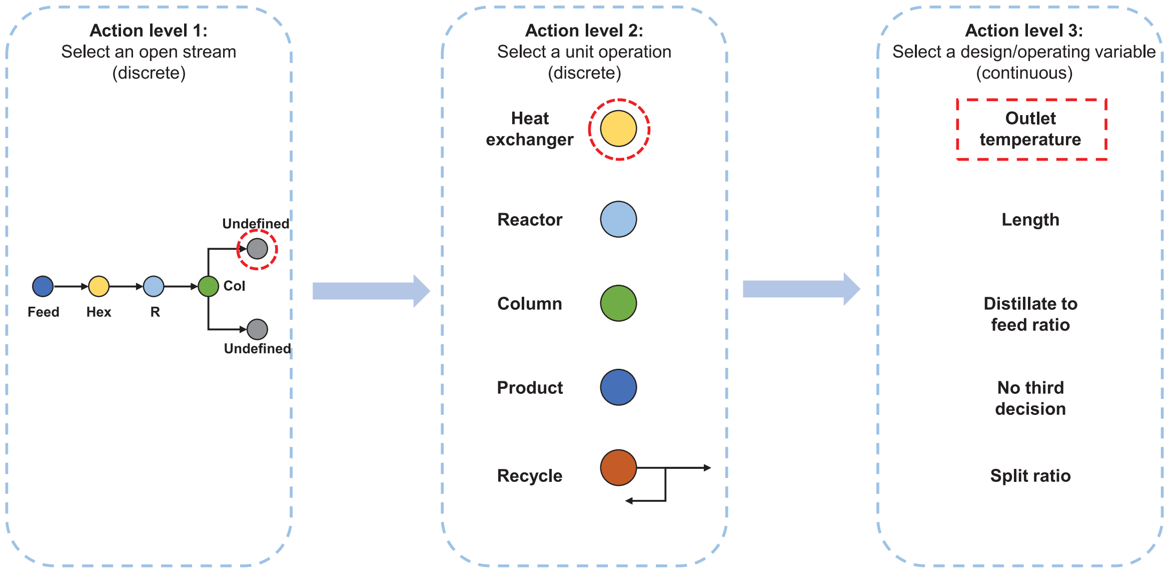

Fig. 6. An overview of node message passing in graph convolution layers.

2.2.3. *Critic networks*

The critic network estimates the value of the current state, providing crucial feedback to the actor network (Sutton and Barto). This value estimates the total expected rewards from the current state to the end of an episode under the current policy. This feedback helps the actor adjust its policy to maximize long-term rewards. Specifically, as shown in Fig. 3, in our framework we calculate a generalized advantage estimation $\hat{A}$, which assesses whether actions taken yield better or worse outcomes than predicted, which is crucial for computing the critic network's loss function.

3. Case study

The proposed framework is illustrated through a case study centered on a hypothetical reaction synthesis process and its according separation. This process operates within a chemical system comprising four compounds: A, B, C, and D. Additionally, we consider a scenario in which the resulting mixtures are assumed to behave ideally and are separated exclusively through distillation. In all simulations, the initial conditions feature a feed composed of an equimolar binary mixture of A and B. The molar flow rate is set to 100 mol s^{-1}, with the feed temperature maintained at 300 K.

3.1. *Environment*

Two different environment settings are utilized in this study: Short-cut process models and DWSIM. The short-cut process methods for pre-training are illustrated in our previous work (Stops et al., 2022). Here, we introduce the DWSIM process simulator as an RL environment. As introduced in Section 2.2.2, the agent can choose between reactors, distillation columns, and heat exchangers as unit operations. Besides, the agent can also decide to add recycles or claim open streams as products. The types of unit operations and the corresponding design variables are defined below.

3.1.1. Reactor

A plug flow reactor (PFR) is deployed to convert reactants to the desired products. Specifically, the following reversible reaction takes place as Eq. (1) shows and C is the desired product.

$$A + B \rightleftharpoons C + D \tag{1}$$

To further enhance the reaction engineering challenge, we assumed the reaction kinetics of our hypothetical reaction to follow those of Xu and Chuang on the esterification of acetic acid with methanol to produce methyl acetate (Xu and Chuang, 1996) However, it is important to note that while we have adopted their kinetic data, our case study differs from their experimental setup. Specifically, we have simplified the model by not considering catalyst loading and simulating the reactor isothermally which indicates the constant reaction rates. We note that using a PFR was partly an arbitrary choice for this case study, as it is a common reactor type. Additionally, the reactor cross-sectional area is determined by $N/10$, where N is the inlet molar flow. The design variable is the reactor length, which will be determined by the agent in the third-level continuous decision process. The range is from 3 to 30 m.

3.1.2. Distillation column

A distillation column is provided in the environment to separate product C from the quaternary system. Rigorous distillation columns are used instead of shortcut columns in the previous work (Stops et al., 2022) to account for more realistic factors such as varying stage conditions. The rigorous column models provide multiple possible choices of design parameters, from which the distillate to feed ratio (D/F) is selected as the third-level decision. Other adjustable parameters such as the number of stages and reflux ratio are set as fixed values (35 and 1.5, respectively). The D/F ratio can range from 0.3 to 0.7.

3.1.3. Heat exchanger

For temperature changes, we provide the agent with the DWSIM heater model. In the proposed framework, the heat exchanger is simulated based on the outlet temperature, which is used as a design variable determined by third-level decisions. A temperature range from 278.15 to 330.05 K is defined, where the upper limit refers to the lowest boiling point of the components, which is product C.

3.1.4. Recycle

The recycling action consists of additional units that include a splitter and a mixer. Firstly, the process stream is split into a recycle stream and a purge stream. Secondly, the recycled stream is merged with the selected feed using a mixer. Thereby, the split ratio is the third-level decision of the agent, which lies in the range of 0.1 to 0.9. In the current setting, the recycle destination is always the feed stream. In principle, it is possible to predefine the destination of a recycle stream within our framework. For instance, one might specify that the recycle stream can only be directed to the inlet before the reactor. When the agent selects a recycle action, the RL environment can then check whether a reactor exists. If so, the recycle stream is placed at the reactor inlet; if not, the action is deemed invalid, and a penalty is given. However, in a more realistic scenario, the RL agent should have the flexibility to determine the optimal recycle location. Future work could address this by introducing an additional action level 4, leveraging GNNs for edge (stream) classification, similar to action level 1 (node classification). When the RL agent selects a recycle action, the action level 4 GNN will predict the most suitable recycle destination.

3.2. Reward

We calculate the reward according to Eq. (2) when a flowsheet is completed.

$$r = \sum P_{\text{products}} - \sum C_{\text{feed}} - \sum \left(C_{\text{operation}} + 0.15 * C_{\text{invest}}\right)_{\text{units}} \tag{2}$$

where P_{products} is the revenue of the sold product (Seider et al. 2008), C_{feed} is the costs of feeds, $C_{\text{operation}}$ is the operation costs (Smith, 2016) and C_{invest} is the total capital investment which is multiplied by factor 0.15 (Seider et al. 2008). In the case of negative rewards, a reduction factor of 10 is applied to encourage the exploration of the design space. Additionally, a reward of 0 € is given when the incomplete flowsheets can converge after every single action because the economic value is difficult to assess for an incomplete flowsheet. Whenever the agent fails the simulation by taking infeasible actions, the episode will be terminated immediately, and a negative reward -10M € is given.

3.3. Constraints

In our framework, we incorporate constraints at three levels: variable bounds, soft constraints, and masking. Hard constraints are a promising future work. First, we apply variable bounds (aka design constraints) to define the ranges of feasible values for each decision. For example, we predetermine the allowable PFR length so the RL agent can only select feasible lengths. Second, we introduce soft constraints that penalize infeasible actions through the reward function without strictly forbidding them. For instance, if an action leads to nonconvergence, the agent incurs a large negative reward, discouraging it from selecting similar actions again. A temperature constraint on the PFR could be also handled through a soft constraint method. Finally, we also employ an action-masking strategy. For example, in action level 1, when selecting an open stream, we apply action masking to prevent the selection of "illegal" streams, ensuring that only valid open streams are chosen. However, the consideration of nonlinear constraints is an active field of research in RL. There exist a number of methods that could be integrated into our framework as a future work. For example constrained RL algorithms (Achiam et al., 2017) and safe exploration techniques (García and Fernández, 2015) have been successfully applied in control settings to handle constraints, and could similarly be leveraged for process design tasks in future work.

3.4. Training process

The method described is implemented using Python 3.9, with the training scheme based on the Proximal Policy Optimization (PPO) algorithm developed by OpenAI (Schulman et al., 2017). This approach involves several epochs of minibatch updates, where each minibatch is formed by sampling transition tuples that have been saved in memory. The agent's networks are updated through gradient descent. The final loss is the weighted sum of each individual actor network, their entropies, and the critic's loss, ensuring a balanced update mechanism that considers both exploration (via entropy) and exploitation (via actor and critic losses).

The agent's decision-making capabilities are enhanced throughout the training process by adjusting its weights and biases. In our training setup, the agent's parameters are updated every 20 steps. At each update interval, the agent executes a batch loss computation using 30 randomly selected samples from its memory, subsequently updating its parameters through the backpropagation algorithm.

The experiments are performed on a Windows server with a 3.5 GHz 24 cores Intel(R) Xeon(R) W-2265 CPU, NVIDIA GeForce RTX 3090 GPU, and 64 GB memory. The hyperparameters can be found in Table 8 and Table 9.

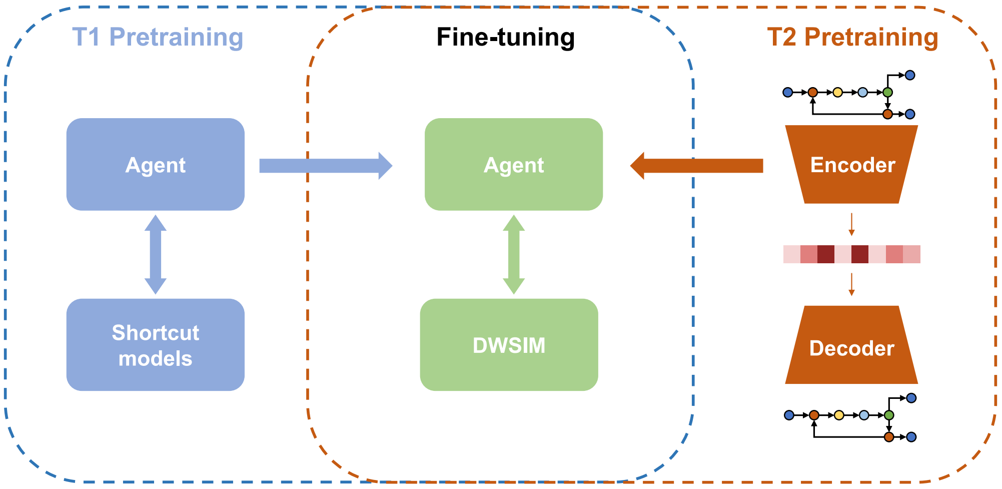

Fig. 7. Overview of transfer learning. In T1 pretraining, the agent interacts with shortcut models to build foundational knowledge. The agent is then fine-tuned with the rigorous process simulator DWSIM. Concurrently, T2 pretraining involves an encoder–decoder architecture, and then the part of the encoder is transferred and fine-tuned with the rigorous process simulator DWSIM.

Table 1
Overview of transfer learning models strategies.

Model components	$T1$					$T2$
	E1	E2	E3	E4	E5	E6
Graph encoder	✓	✓	✓	✓	✓	✓
Actor level 1	–	✓	✓	✓	✓	–
Actor level 2	–	–	✓	✓	✓	–
Actor level 3	–	–	–	✓	✓	–
Critic network	–	–	–	–	✓	–

3.5. *Transfer learning*

Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a second task, incorporating pre-existing knowledge to improve performance or reduce training time. We propose to utilize two different transfer learning strategies as Fig. 7 illustrates. Table 1 depicts detailed transferred components of the models for two strategies.

In the first transfer learning strategy ($T1$), we pre-train the agent using a shortcut process model from our prior research (Stops et al., 2022), followed by fine-tuning with the rigorous DWSIM simulator on the same case study. Throughout both phases, the agent maintains an identical architecture. We explore four distinct experiments by transferring various combinations of model component groups E1 to E5.

In the second transfer learning strategy ($T2$), a graph VAE is developed to learn from syntactical process data and actual process topology data. The actual process topology data comprises 730 flowsheets from simulation files (Vogel et al., 2023), P&ID-like flowsheets without control structure (Hirtreiter et al., 2023), and mined flowsheets from literature and the internet (Balhorn et al., 2022). The synthetic dataset consists of 7952 flowsheets, which are generated based on a Markov chain-like sampling with fixed probabilities and predefined local patterns, such as reaction, thermal separation, or recycling (Vogel et al., 2023). Only the encoder component of the process VAE is then transferred and fine-tuned using the rigorous process simulator DWSIM (group E6). For implementation details of the VAE, we refer readers to our paper (Theisen et al., 2025) .

4. Results and discussion

In this part, we will present the results of the two different transfer learning strategies separately. Specifically, in Section 4.1, we will discuss the results of learning curves from the with and without transfer learning groups. Additionally, Section 4.2 presents the flow sheets generated from all the groups.

4.1. *Learning curves*

Fig. 8 and Table 2 display the learning curves and convergence scores for the baseline group (C1) and two transfer learning strategies T1 and T2. Every group conducted over 10000 episodes for three individual runs because we observed that the learning curves plateaued and showed no further improvement after 10,000 episodes. The score represents the moving average of the rewards over every 100 episodes. Specifically, the solid line in Fig. 8 represents the mean of three individual runs and the shaded area represents the standard deviation for each group. The training for the agent without transfer learning took 72 h over 10000 episodes. For the groups (E1–E5) in the transfer learning strategies T1, the pre-training took 2 h over 10000 episodes and the further training took 72 h over 10000 episodes. Furthermore, for the group (E6) in the transfer learning strategies T2, the pre-training took 6 h and the further fine-tuning process took 72 h over 10000 episodes. Table 2 provides an overview of the convergence plateaus for groups with and without transfer learning. The convergence score, measured in millions of euros (ME), and the number of episodes required to reach various profitability thresholds (0 M€, 20 M€, 40 M€, 60 M€) are reported.

Upon learning convergence, all agents demonstrate the capacity to create feasible flowsheets, each achieving average scores exceeding 60 million euros. The baseline group without transfer learning (C1) has a convergence score of 60.24 ± 3.25 M€. In contrast, groups E1 to E5, which incorporate transfer learning, all achieve higher convergence scores. For example, E1, which transfers only the graph encoder, improves the convergence score by approximately 8.1% compared with C1. Remarkably, E5 with all transferred agent architectures has the highest convergence score 66.81 ± 2.84 M€, representing an improvement of 11.0% over C1. The results demonstrate that agents can successfully transfer the information learned in the shortcut environment to enhance performance when fine-tuning in rigorous environments. Furthermore, we utilize reaching episodes on certain thresholds to measure and compare the training efficiency and speed. For group C1, reaching the 0 M€, 20 M€, 40 M€, and 60 M€ milestones required 126, 1267, 2413, and 3975 episodes, respectively. Remarkably, groups E3, E4, and E5 demonstrate a significantly quicker learning process than C1. For example, all three groups can generate feasible flowsheets (> 0 M€) in the first episode, indicating the transferred information from the shortcut column contributes to making feasible action in the

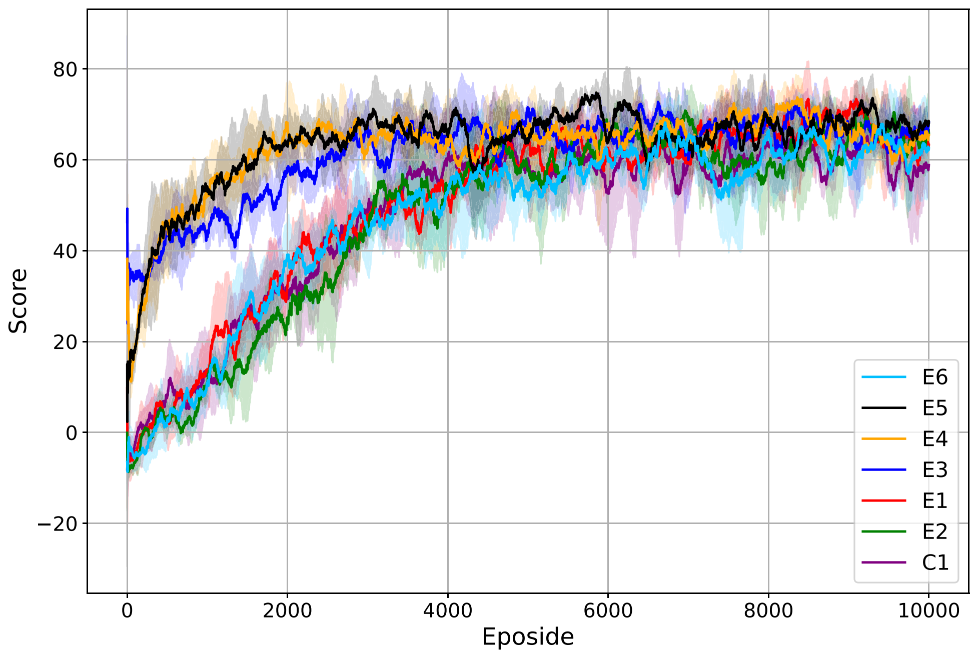

Fig. 8. Learning curves of all groups, smoothed scores plotted with standard deviations. C1 represents the score from the agents without transfer learning. E2–E5 represent the scores from the agent with transfer strategy T1. E6 represents the score from the agent with transfer strategy T3. The detailed transfer learning strategies are depicted in Table 1.

very early stage when fine-tuning in rigorous simulation. Notably, E3 in the first episode can reach the 40 M€ threshold which decreases the training episode time by 99.92%. Additionally, E5 with transferring all components substantially boosts learning velocity with only 1436 episodes for 60 M€ milestones, which decreases the training episode time by 63.88% than C1. On the other hand, transfer learning strategies E1 and E2 led to less improved or even slower learning processes. On the other hand, transfer learning strategies E1 and E2 resulted in less improvement and even slower learning processes. Specifically, both groups required 4303 and 4276 episodes, respectively, to reach the 60 million euro milestone, increasing the training episodes by 8.22% and 7.55%. This suggests that for E1 and E2, the agents may not efficiently utilize the state representations and initial policy guidance provided by the pre-trained encoder and first-level actor, which hinders comprehensive policy optimization and efficient exploitation of the learned state representations. In contrast, transferring additional actor levels and the critic, as seen in E3, E4, and E5, helps to better refine policies and value estimations, leading to faster convergence and higher overall performance.

The learning curve of E6 with the transferred VAE encoder demonstrates a similar performance compared with C1. Quantitatively, the convergence score of E6 increases by 1.46% compared to C1. However, E6 requires significantly more episodes (333) to reach 0 M€, and ultimately 5505 episodes to reach 60 M€, indicating a less efficient learning process. This discrepancy may arise from the misalignment between the pre-training process data and the target process in RL. The pre-training process data consists of 730 flowsheet topologies from different resources (Balhorn et al., 2022; Schweidtmann, 2024), which are comparatively larger than the target process in RL and typically lack steam information. The processes used for pretrapping the VAE are further a diverse range of processes with a broader range of chemicals, unit operations, and topologies. Therefore, the encoder from VAE during the pertaining process potentially yields mismatched learning semantics between the pre-training phase and fine-tuning phase. The mismatched semantics preserved in the encoder hinder further learning and converging in the fine-tuning within the target RL task.

In summary, transfer learning from a shortcut process simulator to a rigorous simulator significantly enhances the learning performance of agents, as evidenced by all the higher convergence scores of groups

Table 2
Overview of convergence plateaus of with and without transfer learning groups.

Group	Convergence score (M€)	Reaching episodes			
		0 M€	20 M€	40 M€	60 M€
C1	60.24 ± 3.25	127	1268	2414	3976
E1	65.12 ± 3.35	**1**	1063	2136	4303
E2	63.12 ± 3.66	201	1599	2712	4276
E3	66.59 ± 2.44	**1**	**1**	**1**	2373
E4	66.62 ± 3.03	**1**	**1**	324	1503
E5	**66.81** ± **2.84**	**1**	113	311	**1436**
E6	61.13 ± 3.45	333	1261	2125	5505

E1 to E5. The varying degrees of improvement in learning efficiency between groups may also indicate the differential impact of the types of knowledge transferred, the strategies employed, and the adaptation to the new environment. In terms of learning curves, overall, group E5 demonstrates the highest convergence scores and the most efficient learning process. On the other hand, the results of transfer learning from a pre-trained VAE highlight the challenges of applying transfer learning in RL for process design, emphasizing the importance of aligning pre-training tasks closely with the target RL tasks to achieve meaningful performance improvements.

4.2. *Generated flowsheets*

In this part, we discuss the generated flowsheets from all groups. Fig. 9 and Table 3 show the generated flowsheet with the highest score obtained from the pre-training phase of the shortcut model, which resembles our previous work (Stops et al., 2022). Furthermore, Fig. 10 displays the flowsheet from C1 (250 episodes). Notably, agents without transfer learning tend to create unfeasible and lengthy flowsheets including multiple unnecessary heat exchangers in the early stage. This is due to the random exploration as part of the trial-and-error nature of the RL. With random initialization, agents have an inadequate learning experience and a preference to consistently select heat exchangers because of their relatively low capital cost and ease of convergence. Fig. 11 visualizes the flowsheet from E3 (1 episode). Notably, with the transferred graph encoder and actor networks 1 and

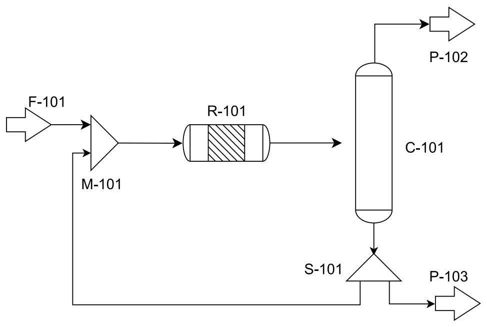

Fig. 9. The generated flowsheet with the highest score 41.66 M€ obtained from the shortcut model.

Table 3
Design variables of unit operations for the flowsheet shown in Fig. 9.

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	8.07
T-101	D/F ratio	\	0.53
S-101	Split ratio	\	0.34

2, agents can already generate a concise and feasible flowsheet in the first episode compared with C1. This additionally verifies that the agent can leverage the learned information from the shortcut process model and make quick and feasible decisions at a very early stage. Moreover, the decision-making ability of agents is reinforced by accumulated experiences, and thus, better initialization can contribute to long-term outcomes. For example, as Table 2 shows, the convergence score of E2 increases by 8.41% than C1.

To delve deeper into the performance of agents from different groups, we discuss the flowsheets with the highest scores during the training phase from different groups. Fig. 9 illustrates the shortcut-method flowsheet that achieved the highest reward (41.66 M€). The feed stream (F-101) first passes through a plug flow reactor (R-101), and the resulting quaternary mixture is then separated in a distillation column (C-101). The final product, component C, obtains from product stream P-102 at 56.4% purity. Additionally, Fig. 12 presents the flowsheet with the highest reward (106.84 M€) from the C1 group. The corresponding design and operating variables are detailed in Table 4. The feed (F-101) passes through two consecutive PFRs (R-101 and R-102), where products C and D are produced by the esterification of A and B. The resulting quaternary mixture is then separated in a distillation column (C-101). The overhead product from C-101 is further distilled in another column (C-102). In this setup, 90% of the bottom product is recycled and mixed back into the feed, while the mixture of B and D exits through product stream P-104. Additionally, C with 77.4% purity is obtained from product stream P-102, which increases by 37.2% compared with the flowsheet obtained from the shortcut methods, and pure D is obtained from product stream P-103. For the E3 group, the highest-scoring flowsheet achieved 107.53 M€, representing a 0.65% increase compared to the C1 group. As shown in Fig. 13, the flowsheet generated by the E3 group is similar to that of C1, with one notable difference: the inclusion of an additional cooler (H-101) before the second distillation column (C-102). At lower temperatures, the vapor–liquid equilibrium conditions change, which can favor the separation of components with higher relative volatilities. This modification potentially makes C more readily separable due to its properties at lower temperatures, resulting in a slightly higher purity of C in the E3 flowsheet compared to C1. Both flowsheets utilize two consecutive reactors, likely because the predefined reactor length range (3 to 30 m) is too short (see Tables 3 and 5).

Further investigation into the flowsheet generated by the E5 group, as shown in Fig. 14 and detailed in Table 6, achieves a highest score of 95.43 M€, which is 10.68% and 11.25% lower than the scores

Table 4
Design variables of unit operations for the flowsheet shown in Fig. 12.

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	23.93
R-102	Length	m	24.36
T-101	D/F ratio	\	0.70
T-102	D/F ratio	\	0.70
S-101	Split ratio	\	0.90

Table 5
Design variables of unit operations for the flowsheet shown in Fig. 13.

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	23.65
R-102	Length	m	25.26
C-101	D/F ratio	\	0.69
H-101	Outlet temperature	K	278.15
C-102	D/F ratio	\	0.70
S-101	Split ratio	\	0.90

Table 6
Design variables of unit operations for the flowsheet shown in Fig. 14.

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	27.22
C-101	D/F ratio	\	0.46
H-101	Outlet temperature	K	279.23
R-102	Length	m	23.29
C-102	D/F ratio	\	0.46
R-103	Length	m	13.76
S-101	Split ratio	\	0.9

Table 7
Design variables of unit operations for the flowsheet shown in Fig. 15.

Unit operation	Design variable	Unit	Value for best run
R-101	Length	m	29.24
C-101	D/F ratio	\	0.66
R-102	Length	m	22.40
C-102	D/F ratio	\	0.30
S-101	Split ratio	\	0.83

from C1 and E3, respectively. The flowsheet from E5 includes several unsuitable designs: two unnecessary reactors are placed after the distillation columns (C-101 and C-102), and an unnecessary heat exchanger (H-101) is added before the product stream (P-101). The underlying reason is that discrepancies in the reward function and design variables between the shortcut and rigorous simulators can introduce additional bias, leading to lower scores for the highest-scoring flowsheet compared to C1 and E3. E5 requires fewer episodes to reach the highest convergence score threshold (60 M€), as indicated in Table 2. This further reflects that E5 has a more stable training process with consistent high-reward flowsheets as the score is the moving average of the previous 100 episodes.

Table 7 depicts the generated flowsheet with the highest score of 93.09 M€, which is a decrease of 12.87% compared to group C1. One clear unsuitable design is that one reactor (R-102) is included after the distillation column (C-101), which is quite similar to the flowsheet from E5. Additionally, C with 57.9% purity is obtained from product stream P-102, which is comparatively lower than the other groups. The possible underlying reason is that the learned semantics from the pre-trained encoder greatly mismatched the task in the fine-tuning RL task. This leads to a suboptimal solution within the same training episodes.

4.3. Discussion on negative transfer

Transfer learning not only contributes to higher convergence scores but also reduces training time, as demonstrated by the E3 group. However, in other groups — particularly those employing T2 strategies

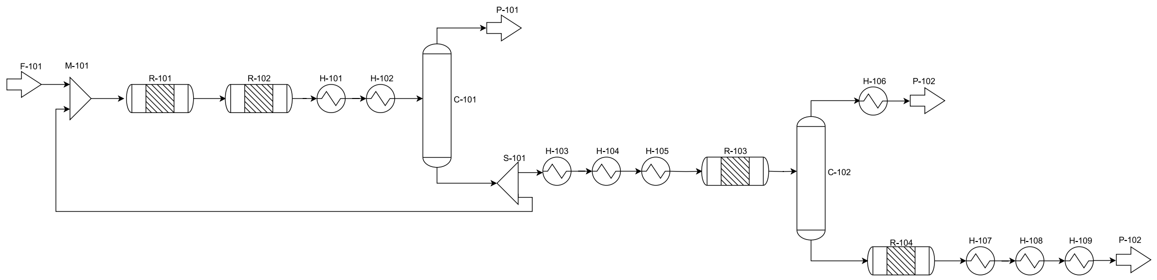

Fig. 10. An example of process flowsheet diagram of group C1, number of episodes = 250.

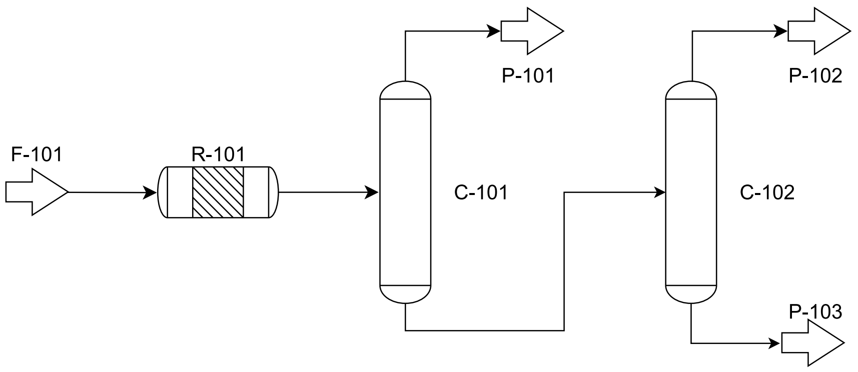

Fig. 11. An example of process flowsheet diagram of group E3, number of episodes = 1.

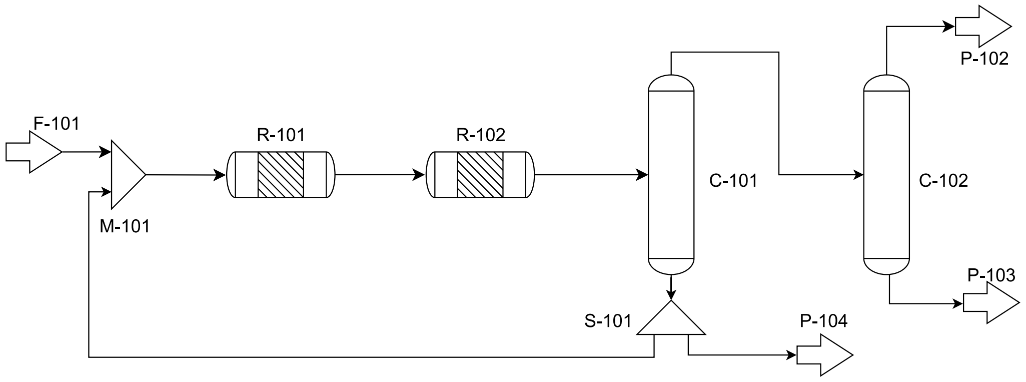

Fig. 12. The flowsheet with the highest score (106.84 M€) from C1.

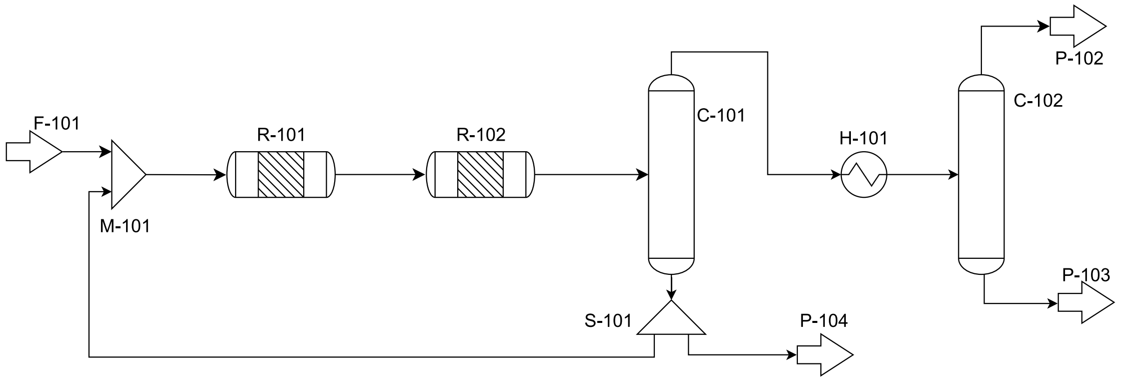

Fig. 13. The flowsheet with the highest score (107.53 M€) from E3.

— negative transfer becomes evident. For T1 strategies, this effect is especially significant in the E4 and E5 groups, which, unlike E3, also transfer the actor-level 3 and critic networks, respectively. A key factor in the negative transfer observed for the E4 group is the substantial discrepancy between the design/variable ranges used in the pre-training phase and those used in the fine-tuning phase. For

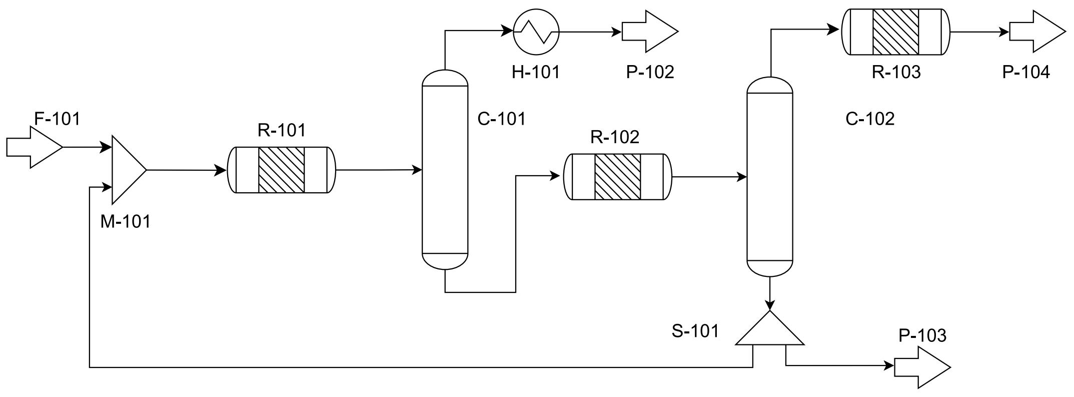

Fig. 14. The flowsheet with the highest score (95.43 M€) from E5.

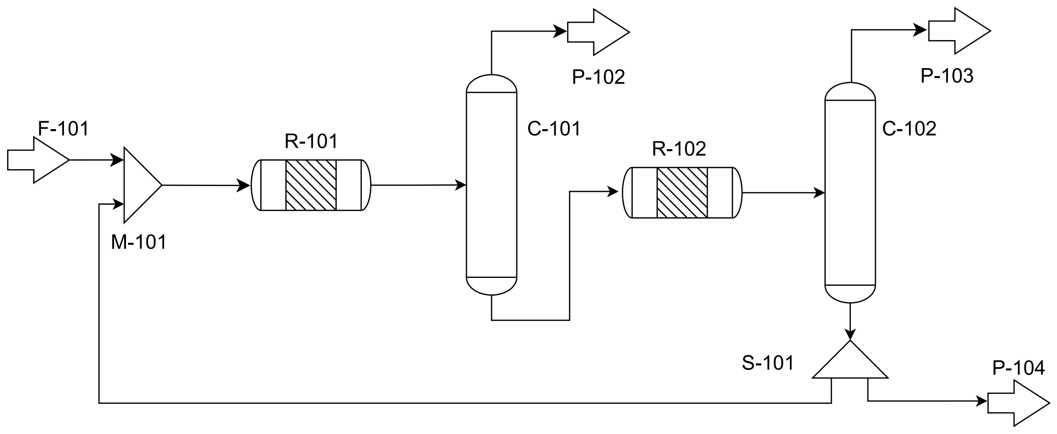

Fig. 15. The flowsheet with the highest score (93.09 M€) from E6.

instance, while the reactor length range for pre-training spans 0.05 to 20 m, it shifts to 3 to 30 m during fine-tuning. Such a mismatch can initially place the model in a suboptimal portion of the design space, slowing adaptation and ultimately affecting the final convergence score within the same training episodes. Similarly, in the E5 group, the potential cause of negative transfer lies in the differences between the pre-training and fine-tuning objective functions. For example, during fine-tuning, we introduced various utility supplies for the heat exchanger (e.g., low/medium/high-pressure steam) to capture different temperature differences. By contrast, the utility supply was fixed during pre-training. This shift in how the objective function is formulated places the critic networks in a different operating regime. As a result, the networks struggle to accurately approximate the new objective function, ultimately leading to negative transfer.

The training data used for the VAE (Theisen et al., 2025) appears to be the potential cause of negative transfer for the T2 strategy. As shown in Fig. 16, the VAE was pre-trained on a dataset comprising both real (simulation files and flowsheet data from the literature) and synthetic datasets. For the node distribution, the real dataset averages 21.51 nodes per flowsheet, while the synthetic dataset averages 18.50. In terms of edges, the real dataset averages 23.02 edges per flowsheet, whereas the synthetic dataset averages 18.23. Additionally, Fig. 16 illustrates that the real dataset has a long right tail with some flowsheets containing up to 200 nodes and edges. Clearly, both the real and synthetic datasets include significantly larger topologies than those used in the fine-tuning phase, where the final flowsheet contains only around 10 nodes. Additionally, the processes in the pre-training dataset differ significantly from the target process. For instance, many of the pre-training examples only involve separation processes, such as benzene-toluene separation or ethanol-water distillation. These are inherently different from the target process in our case study — a combined esterification and separation process — which poses challenges for effective transfer learning. Furthermore, only the flowsheet topology (without edge features) was used when pre-training the VAE, whereas the fine-tuning phase includes edge attributes such as temperature, pressure, and molecular fractions. This discrepancy in both the scale of flowsheet topologies and the nature of edge features potentially places the model in a suboptimal regime at the fine-tuning phase, ultimately contributing to the observed negative transfer.

For future works, several approaches could potentially help mitigate negative transfer in both the pre-training and fine-tuning phases. In pre-training, we suggest using unit operations and objective functions similar to those in the fine-tuning phase. For instance, if the fine-tuning phase involves a compressor or a catalyst-based reactor, one could implement or reuse corresponding shortcut models with comparable variable bounds and the same objective function. If a shortcut model is unavailable, surrogate models can serve as an alternative. A surrogate model is a computationally efficient approximation of a complex system, enabling rapid predictions, analysis, and optimization without sacrificing critical accuracy. Recent studies have shown the effectiveness of surrogate modeling for processes such as distillation columns (Lu et al., 2021) and reactors with catalysts (Lastrucci et al., 2024). For VAEs, it is crucial that the pre-training dataset closely resembles the fine-tuning dataset, such as the number and types of unit operations, node and edge features, as well as the same target process with different topologies. This alignment can potentially reduce negative transfer. Additionally, supervised pre-training — such as cost prediction for flowsheets or imitation learning — offers another potential strategy. Imitation learning is a machine learning approach in which an agent acquires behaviors or decision-making strategies by mimicking demonstrations provided by an expert. However, collecting a sufficiently large dataset of flowsheets with labels, or obtaining expert

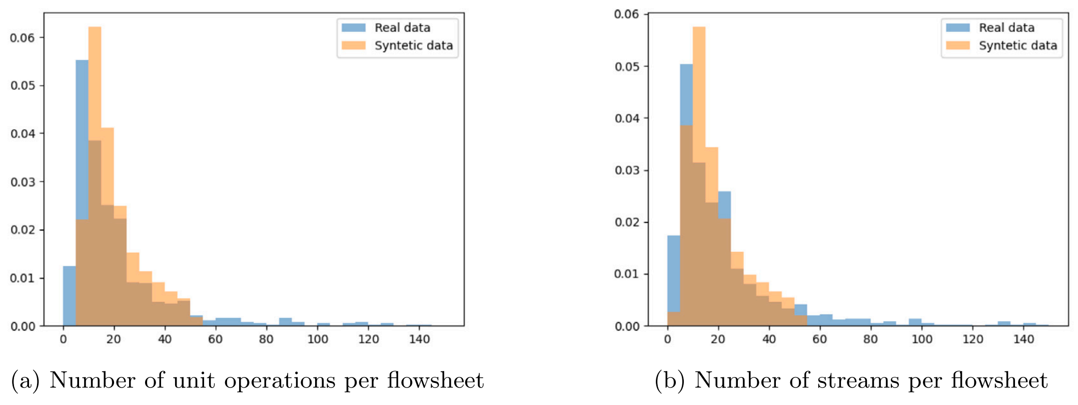

Fig. 16. Graph distributions for the synthetic and the real dataset.

demonstrations for imitation learning, remains a significant challenge in practice.

For the fine-tuning phase, our results indicate that both the actor networks for continuous decisions and the critic network have a vital impact on transfer learning performance. If there is a substantial mismatch in design/operating variable ranges or objective functions between the pre-training and fine-tuning phases, we recommend refraining from transferring these two components and instead training them from scratch. Another possible strategy is to initially freeze or remove the actor networks to focus on optimizing the topology first, then apply a separate method — such as a genetic algorithm — to optimize the design/operating variables (Göttl et al., 2024). For VAEs, if the flowsheet distribution in the pre-training dataset diverges significantly from the target flowsheet distribution in fine-tuning, we likewise advise training the model from scratch. Alternatively, advanced techniques like Low-Rank Adaptation (LoRA) can be leveraged to help manage large parameter spaces and reduce the need for extensive retraining. For instance, applying LoRA to the actor network constrains updates to low-rank factors in the policy parameters, ensuring that the core policy representation remains largely intact. Similarly, using LoRA on the critic or on an encoder network focuses updates on a limited subset of parameters relevant to value estimation or feature extraction, respectively. In each case, the low-rank constraint could potentially prevent the overwriting of beneficial pre-trained representations, thereby reducing the risk of negative transfer.

In conclusion, the learning curves and optimal flowsheets demonstrate that suitable transfer learning can potentially accelerate the training speed and boost the convergence score. Specifically, within T1 transfer learning strategies, the most suitable transfer learning strategy may involve transferring the encoder and the first two actor levels (E3), corresponding to the ability to accurately interpret states and determine the structure of the discrete process unit network, respectively. Conversely, transferring the design variables and critic network may hinder exploration and lead to suboptimal solutions. Moreover, as transfer learning T2 shows, it is of great importance to align pre-training tasks closely with the target RL tasks to achieve meaningful performance improvements.

5. Conclusions

We propose utilizing transfer learning to enhance the learning process of RL in process design. Specifically, we investigate two transfer learning strategies: (i) transfer learning from shortcut process simulators to rigorous simulators (T1) and (ii) transfer learning from process VAE (T2). Our results demonstrate that appropriate transfer learning can significantly improve both learning efficiency and convergence scores. For instance, transferring pre-trained encoder and actor levels 1 and 2 (E3) reduced convergence training episodes by 63.88% and increased convergence scores by 0.65% compared to the group without transfer learning (C1).

Conversely, transfer learning can also negatively impact the learning process. Transferring continuous action networks and critic networks can hinder agent learning if there are substantial discrepancies in decision range and reward function between pre-trained and fine-tuning environments. Additionally, pre-training the encoder with existing flowsheet data yielded only minor improvements in convergence scores and sometimes slowed down the fine-tuning process. This suggests that pre-trained process data should match the complexity of the fine-tuning task.

This work provides guidance on effectively utilizing transfer learning strategies to enhance the efficiency and effectiveness of RL in process design. Future research could explore incorporating more comprehensive flowsheet topology data with additional unit operations and stream information to pre-train the graph encoder. Additionally, RL could potentially integrate with other GenAI tools to support further engineering workflow (Schweidtmann, 2024).

CRediT authorship contribution statement

Qinghe Gao: Writing – original draft, Software, Methodology, Investigation, Data curation. **Haoyu Yang:** Writing – review & editing, Investigation. **Maximilian F. Theisen:** Writing – review & editing, Investigation. **Artur M. Schweidtmann:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

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.

Acknowledgment

This work was supported by the Netherlands Organisation for Scientific Research (NWO) under the Veni grant.

Appendix

See Tables 8 and 9.

Data availability

The data that has been used is confidential.

Table 8
Hyperparameters for the architecture and pre-training procedure of the actor–critic agent.

Parameter		Value
Learning rate	α	0.0002
Policy clipping factor	ϵ	0.3
Discount factor	γ	1.0
λ-return factor	λ	0.95
Number of steps after which the agent learns	n_B	60
Batch size during learning	n_{MB}	30
Number of epochs	n_E	4
Weight for loss of level 1 actor	c_0	0.1
Weight for loss of level 2 actor	c_1	1.0
Weight for loss of level 3 actor	c_2	0.5
Weight for loss of critic	c_3	0.2
Weight for entropy of level 1 actor	d_1	0.001
Weight for entropy of level 2 actor	d_2	0.3
Weight for entropy of level 3 actor	d_3	0.001
Hidden layers edge processing for fingerprint	–	10
Message passing steps for fingerprint	–	6
Hidden layers dimension level 1 actor	–	12
Hidden layers dimension level 2 actor	–	256
Hidden layers dimension level 3 actor	–	256
Hidden layers dimension critic	–	256
Feature size flowsheet fingerprint	–	50

Table 9
Hyperparameters for the architecture and fine-tuning procedure of the actor–critic agent with rigorous simulator.

Parameter		Value
Actor level 1 learning rate	α_1	1×10^{-6}
Actor level 2 learning rate	α_2	1×10^{-5}
Actor level 3 learning rate	α_3	1×10^{-6}
Critics network learning rate	α_4	1×10^{-6}
Policy clipping factor	ϵ	0.3
Discount factor	γ	1.0
λ-return factor	λ	0.95
Number of steps after which the agent learns	n_B	20
Batch size during learning	n_{MB}	30
Number of epochs	n_E	4
Weight for loss of level 1 actor	c_0	0.1
Weight for loss of level 2 actor	c_1	1.0
Weight for loss of level 3 actor	c_2	0.5
Weight for loss of critic	c_3	0.2
Weight for entropy of level 1 actor	d_1	0.001
Weight for entropy of level 2 actor	d_2	0.3
Weight for entropy of level 3 actor	d_3	0.001
Hidden layers edge processing for fingerprint	–	10
Message passing steps for fingerprint	–	6
Hidden layers dimension level 1 actor	–	12
Hidden layers dimension level 2 actor	–	256
Hidden layers dimension level 3 actor	–	256
Hidden layers dimension critic	–	256
Feature size flowsheet fingerprint	–	50

References

Achiam, J., Held, D., Tamar, A., Abbeel, P., 2017. Constrained policy optimization. http://dx.doi.org/10.48550/ARXIV.1705.10528.

Balhorn, L.S., Gao, Q., Goldstein, D., Schweidtmann, A.M., 2022. Flowsheet recognition using deep convolutional neural networks. In: 14th International Symposium on Process Systems Engineering. Elsevier, pp. 1567–1572.

Gao, Q., Schweidtmann, A.M., 2024. Deep reinforcement learning for process design: Review and perspective. Curr. Opin. Chem. Eng. 44, 101012. http://dx.doi.org/10.1016/j.coche.2024.101012.

Gao, Q., Yang, H., Shanbhag, S.M., Schweidtmann, A.M., 2023. Transfer learning for process design with reinforcement learning. In: 33rd European Symposium on Computer Aided Process Engineering. Elsevier, pp. 2005–2010.

García, J., Fernández, F., 2015. A comprehensive survey on safe reinforcement learning. J. Mach. Learn. Res. 16 (1), 1437–1480.

Göttl, Q., Asif, H., Mattick, A., Marzilger, R., Plinge, A., 2024. Automated design in hybrid action spaces by reinforcement learning and differential evolution. In: KI 2024: Advances in Artificial Intelligence. Springer Nature Switzerland, pp. 292–299.

Göttl, Q., Pirnay, J., Burger, J., Grimm, D.G., 2025. Deep reinforcement learning enables conceptual design of processes for separating azeotropic mixtures without prior knowledge. Comput. Chem. Eng. 194, 108975. http://dx.doi.org/10.1016/j.compchemeng.2024.108975.

Göttl, Q., Tönges, Y., Grimm, D.G., Burger, J., 2021. Automated flowsheet synthesis using hierarchical reinforcement learning: Proof of concept. Chem. Ing. Tech. 93 (12), 2010–2018. http://dx.doi.org/10.1002/cite.202100086.

Hirtreiter, E., Schulze Balhorn, L., Schweidtmann, A.M., 2023. Toward automatic generation of control structures for process flow diagrams with large language models. AIChE J. 70 (1), http://dx.doi.org/10.1002/aic.18259.

Hussain, M., Bird, J.J., Faria, D.R., 2018. A study on CNN transfer learning for image classification. In: Advances in Computational Intelligence Systems. Springer International Publishing, pp. 191–202.

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

Khan, A.A., Lapkin, A.A., 2022. Designing the process designer: Hierarchical reinforcement learning for optimisation-based process design. Chem. Eng. Process. - Process. Intensif. 180, 108885. http://dx.doi.org/10.1016/j.cep.2022.108885.

Lastrucci, G., Theisen, M.F., Schweidtmann, A.M., 2024. Physics-informed neural networks and time-series transformer for modeling of chemical reactors. In: 34th European Symposium on Computer Aided Process Engineering / 15th International Symposium on Process Systems Engineering. Elsevier, pp. 571–576.

Lu, J., Wang, Q., Zhang, Z., Tang, J., Cui, M., Chen, X., Liu, Q., et al., 2021. Surrogate modeling-based multi-objective optimization for the integrated distillation processes. Chem. Eng. Process. - Process. Intensif. 159, 108224. http://dx.doi.org/10.1016/j.cep.2020.108224.

Mahalec, V., Motard, R., 1977. Procedures for the initial design of chemical processing systems. Comput. Chem. Eng. 1 (1), 57–68. http://dx.doi.org/10.1016/0098-1354(77)80008-2.

Midgley, L.I., 2020. Deep reinforcement learning for process synthesis. http://dx.doi.org/10.48550/ARXIV.2009.13265.

Mitsos, A., Asprion, N., Floudas, C.A., Bortz, M., Baldea, M., Bonvin, D., Caspari, A., Schäfer, P., 2018. Challenges in process optimization for new feedstocks and energy sources. Comput. Chem. Eng. 113, 209–221. http://dx.doi.org/10.1016/j.compchemeng.2018.03.013.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013. Playing atari with deep reinforcement learning. http://dx.doi.org/10.48550/ARXIV.1312.5602.

Nian, R., Liu, J., Huang, B., 2020. A review on reinforcement learning: Introduction and applications in industrial process control. Comput. Chem. Eng. 139, 106886. http://dx.doi.org/10.1016/j.compchemeng.2020.106886.

Olivecrona, M., Blaschke, T., Engkvist, O., Chen, H., 2017. Molecular de-novo design through deep reinforcement learning. J. Cheminformatics 9 (1), http://dx.doi.org/10.1186/s13321-017-0235-x.

Perera, A., Wickramasinghe, P., Nik, V.M., Scartezzini, J.-L., 2020. Introducing reinforcement learning to the energy system design process. Appl. Energy 262, 114580. http://dx.doi.org/10.1016/j.apenergy.2020.114580.

Plathottam, S.J., Richey, B., Curry, G., Cresko, J., Iloeje, C.O., 2021. Solvent extraction process design using deep reinforcement learning. J. Adv. Manuf. Process. 3 (2), http://dx.doi.org/10.1002/amp2.10079.

Reynoso-Donzelli, S., Ricardez-Sandoval, L.A., 2024a. Enhancing process flowsheet design through masked hybrid proximal policy optimization. IFAC- Pap. 58 (14), 646–651. http://dx.doi.org/10.1016/j.ifacol.2024.08.410.

Reynoso-Donzelli, S., Ricardez-Sandoval, L.A., 2024b. A reinforcement learning approach with masked agents for chemical process flowsheet design. AIChE J. 71 (1), http://dx.doi.org/10.1002/aic.18584.

Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T., 2019. Transfer learning in natural language processing. In: Proceedings of the 2019 Conference of the North. Association for Computational Linguistics.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. http://dx.doi.org/10.48550/ARXIV.1707.06347.

Schweidtmann, A.M., 2024. Generative artificial intelligence in chemical engineering. Nat. Chem. Eng. 1 (3), http://dx.doi.org/10.1038/s44286-024-00041-5, 193–193.

Schweidtmann, A.M., Rittig, J.G., König, A., Grohe, M., Mitsos, A., Dahmen, M., 2020. Graph neural networks for prediction of fuel ignition quality. Energy Fuels 34 (9), 11395–11407. http://dx.doi.org/10.1021/acs.energyfuels.0c01533.

Siirola, J.J., Rudd, D.F., 1971. Computer-aided synthesis of chemical process designs. From reaction path data to the process task network. Ind. Eng. Chem. Fundam. 10 (3), 353–362. http://dx.doi.org/10.1021/i160039a003.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D., 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362 (6419), 1140–1144. http://dx.doi.org/10.1126/science.aar6404.

Stops, L., Leenhouts, R., Gao, Q., Schweidtmann, A.M., 2022. Flowsheet generation through hierarchical reinforcement learning and graph neural networks. AIChE J. 69 (1), http://dx.doi.org/10.1002/aic.17938.

Sutton, R.S., Barto, A.G., Reinforcement Learning An Introduction. A Bradford Book, p. 552.

Tan, H., Hong, X., Liao, Z., Sun, J., Yang, Y., Wang, J., Yang, Y., 2024. Combining reinforcement learning with mathematical programming: An approach for optimal design of heat exchanger networks. Chin. J. Chem. Eng. 69, 63–71. http://dx.doi.org/10.1016/j.cjche.2023.12.005.

Theisen, M.F., Gao, Q., Schweidtmann, A.M., 2025. Representation Learning for Process Topologies Using Variational Autoencoders. Delft University of Technology, Department of Chemical Engineering, Process Intelligence Research, Van der Maasweg 9, Delft, 2629 HZ, Netherlands, (in preparation).

Tian, Y., Akintola, A., Jiang, Y., Wang, D., Bao, J., Zamarripa, M.A., Paul, B., Chen, Y., Gao, P., Noring, A., Iyengar, A., Liu, A., Marina, O., Koeppel, B., Xu, Z., 2024. Reinforcement learning-driven process design: A hydrodealkylation example. Syst. Control. Trans. 3, 387–393. http://dx.doi.org/10.69997/sct.119603.

van Kalmthout, S.C.P.A., Midgley, L.I., Franke, M.B., 2022. Synthesis of separation processes with reinforcement learning. http://dx.doi.org/10.48550/ARXIV.2211.04327.

Vogel, G., Schulze Balhorn, L., Schweidtmann, A.M., 2023. Learning from flowsheets: A generative transformer model for autocompletion of flowsheets. Comput. Chem. Eng. 171, 108162. http://dx.doi.org/10.1016/j.compchemeng.2023.108162.

Wang, D., Bao, J., Zamarripa-Perez, M., Paul, B., Chen, Y., Gao, P., Ma, T., Noring, A., Iyengar, A., Schwartz, D., Eggleton, E., He, Q., Liu, A., Marina, O., Koeppel, B., Xu, Z., 2022. Reinforcement learning for automated conceptual design of advanced energy and chemical systems. Research Square Platform LLC, http://dx.doi.org/10.21203/rs.3.rs-2248780/v1.

Xu, Z.P., Chuang, K.T., 1996. Kinetics of acetic acid esterification over ion exchange catalysts. Can. J. Chem. Eng. 74 (4), 493–500. http://dx.doi.org/10.1002/cjce.5450740409.

Zhang, W., Deng, L., Zhang, L., Wu, D., 2023. A survey on negative transfer. IEEE/CAA J. Autom. Sin. 10 (2), 305–329. http://dx.doi.org/10.1109/jas.2022.106004.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q., 2021. A comprehensive survey on transfer learning. Proc. IEEE 109 (1), 43–76. http://dx.doi.org/10.1109/jproc.2020.3004555.