Reinforcement Learning-Driven Process Design: A Hydrodealkylation Example

Yuhe Tian*, Ayooluwa Akintolaa, Yazhou Jianga, Dewei Wangb, Jie Bao, Miguel A. Zamarripa, Brandon Paulc, Yunxiang Chenb, Peiyuan Gao, Alexander Noringc, Arun Iyengarc, Andrew Liub, Olga Marinab, Brian Koeppelb, Zhijie Xub

* West Virginia University, Department of Chemical and Biomedical Engineering, Morgantown, WV, US
b Pacific Northwest National Laboratory, US
c KeyLogic Systems LLC, Morgantown, WV, US
* Corresponding Author: yuhe.tian@mail.wvu.edu.

ABSTRACT

In this work, we present a follow-up work of reinforcement learning (RL)-driven process design using the Institute for Design of Advanced Energy Systems Process Systems Engineering (IDAES-PSE) Framework. Herein, process designs are generated as stream inlet-outlet matrices and optimized using the IDAES platform, the objective function value of which is the reward to RL agent. Deep Q-Network is employed as the RL agent including a series of convolutional neural network layers and fully connected layers to compute the actions of adding or removing any stream connections, thus creating a new process design. The process design is then informed back to the RL agent to refine its learning. The iteration continues until the maximum number of steps is reached with feasible process designs generated. To further expedite the RL search of the design space which can comprise the selection of any candidate unit(s) with arbitrary stream connections, we investigate the role of RL reward function and their impacts on exploring more complicated versus intensified process configurations. A sub-space search strategy is also developed to branch the combinatorial design space to accelerate the discovery of feasible process design solutions particularly when a large pool of candidate process units is selected by the user. The potential of the enhanced RL-assisted process design strategy is showcased via a hydrodealkylation example.

Keywords: Process Design, Process Synthesis, Machine Learning, Reinforcement Learning, Optimization

1. INTRODUCTION

Process synthesis aims to determine the optimal selection of unit operations and their flowsheet interconnections at the optimal operating conditions [1]. However, it is not a trivial task to select the optimal process design considering the plethora of plausible unit operations and flowsheet configurations which have been investigated in chemical process industry.

Optimization-based approaches [2-3] offer a systematic strategy utilizing mathematical programming to synthesize the overall flowsheet based on a superstructure which contains all the possible flowsheet structural alternatives of interest. The representation of chemical processes also plays a key role to ensure fit-for-purpose design creativity and sufficient modeling accuracy with tractable computational formulations (e.g., state-task network [4], unit-port-conditioning streams representation [5], phenomena-based representation [6-7]). Although the efficacy of optimization-based process synthesis has been well demonstrated in many chemical and energy systems, major challenges remain on: (i) the request of a user-specified superstructure which heavily relies on engineering expertise to ensure the quality of optimal design solutions, and (ii) algorithmic complexity to solve the resulting large-scale mathematical optimization problems, etc.

To address these challenges, recent research efforts have been made to drive process synthesis using reinforcement learning (RL) in place of optimization algorithms [8-9]. RL-driven process design typically starts from a maximum pool of unit operations without...
requesting any superstructure pre-specification. The intelligent RL agent will select among the available unit operations and generate arbitrary process design structures. The objective function value of the derived process design serves as the reward to keep training RL agent to generate better process designs. Despite the advantages, key open research questions lie in the search efficiency and algorithmic scalability as the number of available unit operations increases.

In this paper, we present a follow-up work to the RL-driven process design approach introduced by Wang et al. [8] integrated with the Institute for Design of Advanced Energy Systems Process Systems Engineering (IDAES-PSE) Framework [10]. To enhance the efficiency of RL-driven design, we will discuss the role of reward functions and the implementation of a sub-space branch and search strategy. Section 2 provides a brief overview of the methodology workflow and highlights the interactions between IDAES-PSE and reinforcement learning. Section 3 introduces the hydrodealkylation case study and the process design analyses. Section 4 presents concluding remarks and ongoing work.

2. RL-DRIVEN PROCESS DESIGN WITH IDAES-PSE FRAMEWORK

2.1. Overview of the Methodology

The methodology workflow is summarized in Fig. 1 with stepwise procedure detailed in what follows [8].

![Figure 1. The methodology workflow.](image)

2.1.1. Step 1 – Candidate process-units pool

The design process starts with the users selecting a maximum set of candidate process units which can be used for process design in the next steps. An indicative list of the process unit models, which are supported in the IDAES model library, includes feed (coupled with mixer), product (coupled with flash), flash drum, mixer/splitter, heat/cooler, heat exchanger, stoichiometric reactor, etc. Users are required to specify the types and the maximum number of units available (e.g., maximum of 3 flash drums), while no pre-postulation of flowsheet connections or superstructures is needed.

2.1.2. Step 2 – Flowsheet representation

After defining the candidate process units, an initial process design will be generated via random initializations in the form of inlet-outlet stream matrix. Table 1 presents an example of the stream matrix representation if the current process flowsheet comprises a heater followed by a stoichiometric reactor. Lists of unit inlets and outlets are first generated to describe the flowsheet connections. The stream matrix then maps the inlet-outlet relationships using 0-1 variables, which serves as the observations to the reinforcement learning algorithm.

<table>
<thead>
<tr>
<th>Feed. outlet</th>
<th>Heater. outlet</th>
<th>Reactor. outlet</th>
<th>Flash. outlet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product.inlet</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Heater.inlet</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Reactor.inlet</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Flash.inlet</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Stream matrix representation.

2.1.3. Step 3 – Connect rules pre-screening

As the flowsheet structure is generated via random initialization (at the first iteration) or via RL (at the successive iterations), it is essential to pre-screen if the structure satisfies general connectivity rules before proceeding with rigorous simulation. Some examples of such rules include:

- At least one unit operation must be selected.
- The outlet of a unit operation cannot be connected with its own inlet.
- Liquid outlet streams cannot connect to compressors or expanders.
- Heaters cannot directly connect to coolers, and compressors cannot directly connect to expanders (turbines).
- There must be a reactor between an outlet product or flash and an inlet feed.

2.1.4. Step 4 – Simulate and optimize process design using IDAES platform

If the structural screening is passed, the flowsheet
is to be simulated and optimized using the IDAES platform with its built-in process-units modeling library, property packages, and mathematical solvers [10]. It is worth noting that the RL reward function is defined as a function of the key process design objectives (e.g., product flow and purity). In this way, the reinforcement learning is continually rewarded to identify better design solutions.

2.1.5. Step 5 – RL agent to generate new design alternatives

This step aims to intelligently learn from the observations (step 2) and reward (step 4), thus making a decision on the next action (i.e., generate a new design). Herein, the RL agent adapts the Deep Q-Network (DQN) as illustrated in Fig. 2. DQN utilizes a series of convolutional neural network layers to extract key features from the 2D stream matrices followed by successive fully connected layers to compute the $Q$ value as per Eq. 1. The $Q$ value will dictate the value of each action, i.e. adding or removing any unit operation to or from a certain location of the current flowsheet structure. The max ($Q_{\text{next}}$) term selects the best action in the future $Q$ values.

$$ Q = \text{reward} + \gamma \cdot \max (Q_{\text{next}}) $$

where $\text{reward}$ is calculated in step 4 as a function of the objective function, $\gamma$ is the decay factor, $Q_{\text{next}}$ is the set of future $Q$ values by taking each possible action.

2.2. Integration of RL and IDAES

This integrated tool can be accessed via the open-source IDAES platform. The information flow between IDAES and RL interactions is summarized in Fig. 3 from software perspective. An interactive user interface has also been developed as a Jupyter Notebook (i.e., Notebook Interface.ipynb) which allows the users to select candidate process units, adjust RL algorithm parameters (e.g., learning rate), execute RL-driven design, and report the results. The RL algorithm provides new process designs to the IDAES platform for optimization, and the IDAES platform supplies RL with the rewards based on optimization results. The flowsheets are represented respectively using unit list arrays in IDAES and stream matrices in RL, the conversion of which is performed by the obs2list file to enable mutual communication.

Figure 2. RL agent using Deep Q-Network.

Therefore, a new process design will be generated by the DQN agent which will be sent to Step 3 for connectivity rule pre-screening and IDAES optimization, etc. until a maximum number of iterations is reached. The outcome of this RL-driven process design strategy is a number of feasible process designs with improved quality against the objective function.

This RL-driven process design method can be applied for grassroots chemical process design, as will be showcased in Section 3. It can also significantly contribute to: (i) Retrofitting to improve existing flowsheet designs, and (ii) Integrating new process technologies (e.g., fuel cell, novel intensified reactors) to existing flowsheet infrastructure. In the latter cases, this method may demonstrate more superior computational efficiency as the design space is smaller.

3. CASE STUDY: HYDRODEALKYLATION PROCESS DESIGN

In this section, we apply the afore-introduced methodology to design a hydrodealkylation (HDA) process. Based on the scenario analyses, we discuss several implementations to further improve the RL-driven design efficacy by refining reward function and performing sub-space branching.

3.1. Problem Statement

An HDA process is to be designed to produce benzene ($C_6H_6$) from toluene ($C_6H_5CH_3$) and hydrogen ($H_2$) at elevated temperatures. The reaction scheme is shown in Eq. 2, which takes place in the vapor phase. The undesired side reactions are not considered in this case study (e.g., diphenyl production). Ideal vapor-liquid equilibrium can be used to describe the mixture separation behavior.

$$ C_6H_5CH_3 + H_2 \rightarrow C_6H_6 + CH_4, \Delta H = -1.08 \times 10^5 \text{ J mol}^{-1} (2) $$
The feed streams are given in Table 2, consisting of a vapor stream of hydrogen and methane and a liquid stream of toluene. Available process units include feed (coupled with mixer), product (coupled with flash operation), heater, cooler, stoichiometric reactor, mixer, flash, splitter, and compressor. The operating constraints of each type of units are also given, such as heat outlet temperature between 500-600 K. Product specification is set to obtain a vapor product with benzene purity higher than 0.55. The design objective is to determine process solution(s) with optimal vapor product benzene flowrate.

### Table 2: HDA feed conditions.

<table>
<thead>
<tr>
<th></th>
<th>Vapor feed</th>
<th>Liquid feed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature (K)</td>
<td>303.2</td>
<td>303.2</td>
</tr>
<tr>
<td>Pressure (kPa)</td>
<td>350</td>
<td>350</td>
</tr>
<tr>
<td>Flowrate (mol/s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Benzene</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Toluene</td>
<td>0.30</td>
<td>0.30</td>
</tr>
<tr>
<td>Hydrogen</td>
<td>0.02</td>
<td>0</td>
</tr>
<tr>
<td>Methane</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.2. Scenario Analysis

We first investigate a series of scenarios with different candidate process units selected by user, i.e. different maximum unit sets that can be used for HDA design.

#### 3.2.1. Scenario 1

In this scenario, at most 1 heater and at most 1 reactor can be used to design the HDA process (Table 3). Feed and product are regarded as defaults to be included in process design. Applying the RL-driven approach, the first design solution obtained is shown in Fig. 4. This design solution is referred as Design 1-1, which represents Scenario 1, Solution 1. It is identified at the 63rd episode.

The flowsheet comprises a heater with an outlet temperature of 536.9 K followed by a reactor with an outlet temperature of 897.8 K. The flash coupled with outlet product only produces a vapor product stream at 705.7 K around 1 atm. The objective value is 0.225 mol/s benzene flowrate, with benzene purity at 0.75 and revenue at \( \$7.65 \times 10^5 \) /year.

To run a total of 300,500 episodes for this scenario, the total computational time is 2.9 minutes. A substantial portion of the time is spent on RL learning to generate process structures which can pass the pre-screening rules. Only around 4 seconds are consumed by IDAES simulation and optimization. From the aspect of conceptual design, Design 1-1 is actually the best solution of all plausible designs with the maximum benzene flowrate and revenue as well as a most simplified flowsheet.

#### Table 3: User specified candidate units – Scenario 1.

<table>
<thead>
<tr>
<th></th>
<th>Heater</th>
<th>Cooler</th>
<th>Reactor</th>
<th>Mixer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Flash</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### 3.2.2. Scenario 2

The pool of candidate process units is increased in this scenario, allowing the use of at most 1 unit of each type as defined in Table 4. The first flowsheet design solution (Design 2-1, representing Design Scenario 2, Solution 1) is identified at the 151,507th episode, as depicted in Fig. 5. The objective value is 0.180 mol/s benzene flowrate, with benzene purity at 0.75 and revenue at \( \$2.49 \times 10^5 \) /year. The inlet feed streams are first heated to 537.6K before entering the reactor. The reactor effluent is cooled down from 898.4K to 325.0K giving a liquid-vapor mixture. The outlet product flash operation then re-heats the product temperature to 595 K and reduces the pressure to around 1 atm. A total of 0.18 mol/s benzene flowrate is obtained from the outlet vapor stream, while no liquid stream is obtained. Another 0.045 mol/s benzene flowrate exists in the outlet exhaust (or purge) stream. While this flowsheet is feasible, it is not a good design. The cooler decreases the energy efficiency and results in the loss of benzene product in the vapor phase via the outlet exhaust.

Another flowsheet design solution (Design 2-2) is generated at the 284,253rd episode, as shown in Fig. 6. The objective value is 0.225 mol/s benzene flowrate, with benzene purity at 0.75 and revenue at \( \$5.08 \times 10^5 \) /year. To run a total of 300,500 episodes for this scenario, the total computational time is 12.0 minutes.

Several observations can be made for Design 2-2: (i) Similar to Design 2-1, the cooler decreases the energy efficiency, (ii) Compressor is redundant as pressure change is not an essential requirement, (iii) Despite the
fact that the splitter is not completely connected to mixer (one of the pre-screening constraints), the loop of mixer, compressor, and splitter is redundant.

3.2.3. Scenario 3

The pool of candidate process units is further increased to allow the use of at most 2 units of each type (Table 5). To run 300,500 episodes for this scenario, the total computational time is 24.6 minutes. A flowsheet design solution (Design 3-1) is found at the 788,161st episode, which is identical with Design 1-1. No more design solutions are found up to 4,150,000 episodes.

<table>
<thead>
<tr>
<th>Heater</th>
<th>Cooler</th>
<th>Reactor</th>
<th>Mixer</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

3.2.4. Summary of Observations

As the design space increases (i.e., user selects more unit operations to use), the following can be observed based on the above scenario analyses:

- More computational efforts are required to identify any feasible flowsheet design solution. For example, to generate the first feasible design, Scenario 1 at 63rd episode, Scenario 2 at 151,507th episode, Scenario 3 at 788,161st episode.
- The RL algorithm tends to explore the use of more unit operations, despite the possible use of redundant units and/or stream loops (e.g., cooler, compressor, recycle loop).
- The solutions generated from larger combinatorial design space may not recover that from smaller design space. For example, Scenario 2 cannot recover Design 1-1, which is the best design solution.

3.3. Impact of Reward Function

To drive the RL algorithm automatically toward a more refined design space with a minimum essential number of unit operations, changing the reward function can be of potential help. The current reward function is given in Eq. 1 which accounts for benzene purity and flowrate. This renders the reward values independent of the number of units. Or in other words, RL does not obtain higher rewards by generating more simplified flowsheet designs. In this context, we propose to incorporate the number of unit operations as part of the reward consideration as defined in Eq. 2.

\[
R_{\text{reward}} = 1000 + \frac{\text{flow}_{0.10}}{0.05} \times \text{delta}_{\text{scoreA}} + \frac{\text{purity}_{0.55}}{0.10} \times \text{delta}_{\text{scoreB}}
\]

(1)

\[
R_{\text{reward}} = 1000 + \frac{\text{flow}_{0.10}}{0.05} \times \text{delta}_{\text{scoreA}} + \frac{\text{purity}_{0.55}}{0.10} \times \text{delta}_{\text{scoreB}} +
50 \times (\text{max}\_\text{number}\_\text{units} - \text{actual}\_\text{number}\_\text{units})
\]

(2)

Testing again on Scenario 2, the modified reward function leads to the identification of a second flowsheet design at the 242,654th episode, which is identical to Design 1-1. This is a faster identification compared to the previous 284,253th episode using Eq. 1 while featuring a reduced number of unit operations to identify the best design solution. This reward function is also tested on Scenario 3, in which the user selects maximum 2 units of each type. The first feasible flowsheet design is still
reported at the 788,161st episode. Compared to the reward function of Eq. 1 which cannot generate a second flowsheet design within 4,150,000 episodes, the modified function of Eq. 2 reports a second design at the 3,069,993rd episode. The flowsheet design comprises a heater, a reactor, a splitter, the outlet exhaust, and the outlet product.

The modified reward function has been

**Figure 7:** Search strategy for RL-driven design.

**Figure 8:** Sub-space search illustration for Scenario 2.
demonstrated to drive the RL search to prioritize the use of fewer unit operations and, thus, to accelerate the search of flowsheet designs after the first design is identified. However, this formulation cannot contribute to accelerate or improve the identification of the first feasible flowsheet design.

3.4. Sub-Design Space Branch and Search

As shown in Fig. 7a, the current RL-driven design is set to always explore the entire design space of all candidate process units specified by the user. Arguably, the infeasible flowsheet design space for the arbitrary selections and combinations of these units can be significantly larger than the feasible space. In this way, with a larger number of available unit operations, the RL-driven design may spend the most time learning the infeasible space while directed away from the feasible space. As such, we propose a parallel sub-space search as illustrated in Fig. 7b. Namely, for each of the feasible flowsheet design generated from learning the entire design space, a quick RL search is performed using the list of units in this design as candidate process units (ignoring the specific inlet and outlet connections in this design). Thus, the sub-space search aims to identify, using a reduced pool of unit operations that can construct a feasible flowsheet design, if better design solutions can be generated. This branch and search strategy is implemented for Scenario 2 and the reported results are given in Fig. 8. Better design options are efficiently identified in the sub-space search.

4. CONCLUDING REMARKS

In this paper, we have introduced a reinforcement learning-driven process design approach with application to HDA production. Two improvements are implemented to enhance the RL efficiency to search the combinatorial design space, respectively by refining the reward function and branching the search space. Ongoing work is applying superstructure optimization to expedite the sub-design space search. In other words, RL will identify a minimum essential set of unit operations based on which superstructure optimization will screen this sub-design space to find the optimal design solution. The integrated strategy aims to augment RL exploration ability in large combinatorial design space and superstructure optimization rapid screening in smaller design space when mixed-integer nonlinear optimization becomes tractable.

5. ACKNOWLEDGEMENTS

Financial support from NETL and ARPA-E DIFFERENTIATE project is gratefully acknowledged.

REFERENCES


© 2024 by the authors. Licensed to PSEcommunity.org and PSE Press. This is an open access article under the creative commons CC-BY-SA licensing terms. Credit must be given to creator and adaptations must be shared under the same terms. See https://creativecommons.org/licenses/by-sa/4.0/