DEEP REINFORCEMENT LEARNING FOR PROCESS SYNTHESIS

A PREPRINT

Laurence I. Midgley
Department of Engineering
University of Cambridge
Cambridge, United Kingdom
and,
Department of Chemical Engineering,
University of Cape Town,
Cape Town, South Africa

laurencemidgley@gmail.com

September 23, 2020

ABSTRACT

This paper demonstrates the application of reinforcement learning (RL) to process synthesis by presenting Distillation Gym, a set of RL environments in which an RL agent is tasked with designing a distillation train, given a user defined multi-component feed stream. Distillation Gym interfaces with a process simulator (COCO and ChemSep) to simulate the environment. A demonstration of two distillation problem examples are discussed in this paper (a Benzene, Toluene, P-xylene separation problem and a hydrocarbon separation problem), in which a deep RL agent is successfully able to learn within Distillation Gym to produce reasonable designs. Finally, this paper proposes the creation of Chemical Engineering Gym, an all-purpose reinforcement learning software toolkit for chemical engineering process synthesis.

***K*eywords** deep reinforcement learning · process synthesis · distillation train

1 Introduction

Reinforcement learning (RL), is a type of machine learning in which an agent makes a sequence of decisions within an environment to maximize an expected reward. RL has had many recent successful applications, including mastering games such as chess and Go [1]. Reinforcement learning has been recently applied to chemical engineering problems, notably process control [2]. Computer aided process synthesis are currently dominated by optimisation techniques such as MINLP [3]. Preliminary work has been done showing the possibility of reinforcement learning for process synthesis using a simple problems simulated using a hand-crafted simulator [4]. This paper builds on this work to present a clear demonstration of RL for process synthesis.[1]

2 Reinforcement learning background

2.1 Reinforcement learning task definition

Reinforcement learning tasks are structured as a Markov Decision Process (MDP)(Figure 1). In an MDP, for each time step of the environment, the agent (1) makes an observation (or partial observation) ω_t, of the state, of the environment s_t, (2) selects an action α_t from the set of possible actions A, based on the agent's policy π, after which (3) the environment transitions to a new state s_{t+1}, and (4) the agent receives reward r_t. The transition of the environment

[1] Code available at https://github.com/lollcat/DistillationTrain-Gym

may be stochastic or deterministic. The environment starts in a given state and follows this sequence of steps until the terminal state is reached. The goal of the agent is to learn a policy that maximizes the total expected reward it receives during the episode, given by $\max_\pi E\left[\sum_t r_t\right]$.

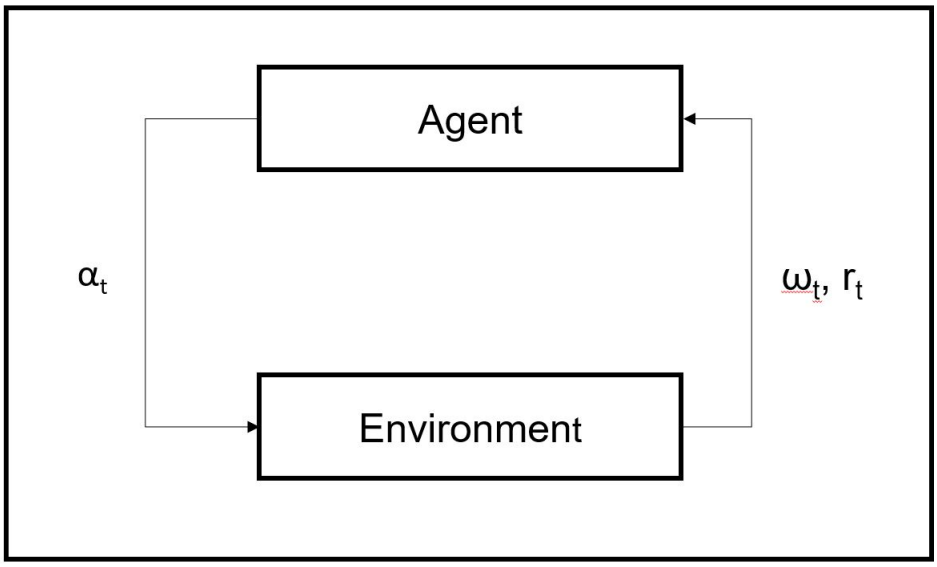

Figure 1: Markov Decision Process Diagram

2.2 Soft Actor Critic

This paper utilizes an adapted version of the Soft Actor Critic (SAC) agent [5, 6]. This section gives a short summary of SAC, while Section 3.3.1 describes the adaption added to SAC to fit Distillation Gym. SAC utilizes an actor critic architecture where the Q-function, which predicts the value of a state conditional on an action, is approximated by a neural network (the "critic"), while the "actor" is a neural network that produces a set of actions aimed at maximizing the Q-value estimated by the "critic". Instead of the typical value function, SAC uses a soft value function with an additional entropy regularization term that encourages exploration. The soft Q-function is given by,

$$Q^\pi(s,a) = \mathrm{E}_{s_t\sim P,\alpha'\sim\pi}[\sum_{t=0}\gamma^t R\left(s_t,a_t,s_{t+1}\right) + \alpha\sum_{t=1}\gamma^t H\left(\pi\left(s_t\right)\right) \mid s_0 = s, a_0 = a] \tag{1}$$

where H, is the entropy of the policy at a given time step and α is a temperature parameter controlling the trade-off between maximising the reward and maximising the policy's entropy. The relation of the soft Q-function the soft value-function is then given by,

$$V\left(\mathbf{s}_t\right) = \mathbb{E}_{\mathbf{a}_t\sim\pi}\left[Q\left(\mathbf{s}_t,\mathbf{a}_t\right) - \alpha\log\pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\right] \tag{2}$$

The parameters θ, of the critic network, can be found by minimising,

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta\left(\mathbf{s}_t,\mathbf{a}_t\right) - \left(r\left(\mathbf{s}_t,\mathbf{a}_t\right) + \gamma\mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\theta}}\left(\mathbf{s}_{t+1}\right)\right]\right)\right)^2\right] \tag{3}$$

Where $\mathcal{D}$ represents a replay buffer of experience. The term for the value-function is implicitly modelled through the relation to the Q-value given in Equation 2. The parameters ϕ, of the actor network can be found by minimizing,

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\left[\mathbb{E}_{\mathbf{a}_t\sim\pi_\phi}\left[\alpha\log\left(\pi_\phi\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\right) - Q_\theta\left(\mathbf{s}_t,\mathbf{s}_t\right)\right]\right] \tag{4}$$

A "reparameterization trick" is then used where the policy reparametrized using,

$$\mathbf{a}_t = f_\phi\left(\epsilon_t;\mathbf{s}_t\right) \tag{5}$$

where, ϵ_t is noise sample from an independent distribution (e.g. spherical Gaussian). Which allows for the actor cost function to be given as,

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}\left[\alpha\log\pi_\phi\left(f_\phi\left(\epsilon_t;\mathbf{s}_t\right) \mid \mathbf{s}_t\right) - Q_\theta\left(\mathbf{s}_t, f_\phi\left(\epsilon_t;\mathbf{s}_t\right)\right)\right] \tag{6}$$

The temperature parameter, α, can be set to a constant, or tuned throughout training for improved performance. In order to prevent over-estimation of the Q-function in the training of the actor, SAC uses two Q-networks, where the minimum Q-value of the networks is taken [7, 8]. Target Q-networks with soft-updates are also used to stabilize training [9]. For a detailed description of SAC, see [5, 6].

3 Distillation Gym

3.1 Overall Description

Distillation Gym is a reinforcement learning environment involving the synthesis of simple distillation trains to separate compounds. In Distillation Gym, an RL agent is tasked with designing a common tree structured simple distillation train (Figure 2), which separates a user defined multicomponent stream. The user is tasked with instantiating a specific problem within distillation gym, defining:

1. The simulation system: component specification, physical and chemical properties of the system
2. Feed stream specification: component flowrates, conditions (temperature, pressure)
3. Product definition: required purity, pure product selling stream prices

Given the user specified problem, the RL agent is tasked with designing a distillation train in order to maximize the expected return. For a given step of the environment the reward is given,

$$r(s_t, \alpha_t) = revenue - TAC \tag{7}$$

where TAC is the total annual cost of the additional column specified by action α_t, and revenue is the annual revenue corresponding to the sum of the value of any product produced in the distillate and bottoms of the additional column. The distillation train is designed through sequential addition of new columns to the existing process, where the agent specifies each column's pressure (controlled by a valve before the column), number of stages, reflux ratio and reboil ratio. The agent operates only through the addition of new distillation columns to stream with unconnected endpoints, which results in the simple distillation train's resulting simple tree structure. There are no other permitted changes to the process design (e.g. adding recycle streams or editing existing units). All unit simulation is performed using COCO and ChemSep [10, 11]. The goal of Distillation Gym is to create a simple illustration of RL's application to process synthesis. Interfacing with a process simulator, is a key component of this demonstration, as more general process synthesis RL tasks would most likely also follow this form.

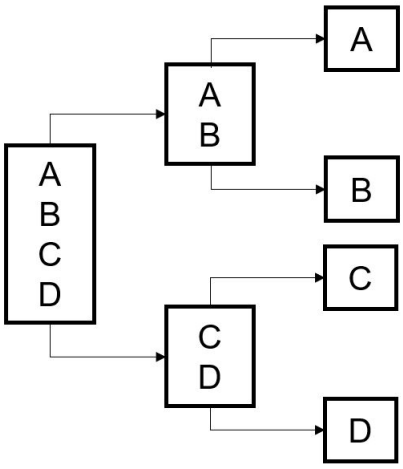

Figure 2: Distillation Gym's simple tree-structured column design

3.2 Markov Decision Process Framing

Current most computational process synthesis techniques typically frame process synthesis as an optimisation problem where an objective function governed by a strict set of parameters and equations is maximised. Defining process synthesis as a reinforcement learning tasks, instead frames process synthesis as a sequential decision-making task

(MDP). To define the MDP for process synthesis, the state, action, state-transition and reward structure need to be defined. The state should represent all the necessary information describing the current process configuration necessary for future design decisions. Typically for process synthesis tasks, the state should therefore most likely include information on all of the streams (resembling the stream table), as well as relevant additional information (e.g. on the unit operations). To feed the state into a neural network, the array that describes the state needs to be of a fixed shape. For process synthesis, this constraint can cause some difficulty as the number of streams changes as the process is designed. One solution to this is to start with a large empty state (resembling a stream table populated with 0's), which can be filled as the process is designed. The action should represent changes to the design of the process configuration (e.g. adding a new unit operation). The state transition is the running of the process simulation to calculate the updated information describing the process at its current point of design (e.g. calculating the flowrates in all of the streams). The reward describes the degree to which the objective of the design task (e.g. maximizing profit) is achieved. One intuitive way to think of process synthesis as an MDP is to imagine a human designing a process within a process simulator; what actions do they take? What information is relevant to their decisions? What is their goal?

In the Distillation Gym, because of the specific structure of the distillation problem, an adjusted form of the MDP can be created which allows for a significant simplification to the state. Specifically, instead of the state describing the process as a whole, through the change to the structure of the MDP, the state can instead describe a single stream. Algorithm 1 below gives a description of the MDP for Distillation Gym.

Algorithm 1: Distillation Gym Environment Episode Structure

```
initialization                                                    // configure flowsheet
D = deque()                          // deque of the process streams with unconnected ends
D.append(FeedStream)
done = False
while not done do
    CurrentStream = D[0]
    observe CurrentStream.state
    choose whether to separate CurrentStream
    if chose to seperate then
        specify column       // inlet pressure, number of stages, reboil ratio, reflux ratio
        simulate flowsheet                              // using interface with COCO simulator
        reward = - TAC
        for Stream ∈ (DistillateStream, BottomsStream) do
            if Stream.purity >= PuritySpecification then
                reward += Stream.revenue
            else
                // if stream is doesn't meet product specification then it may be further
                   separated
                D.append(Stream)
            end
        end
        // Can now remove current stream from D as it is now attached to a column
        D.popleft()
    else
        // if decide not to separate then CurrentStream becomes outlet to process so is
           removed from D
        D.popleft()
    end
    if D is empty or max steps have been reached then
        done = True                                                // episode is over
    end
end
```

The reason this structure is possible is because actions only effect streams downstream in the process. This means that it is only necessary for the current stream being separated to be fed to the agent as the state, as the other process streams are completely independent to this decision. As shown in Algorithm 1, instead of standard linear MDP structure (state $\rightarrow$ next-state), a tree shaped decision process (referred to as tree-MDP) is used, where each state produces two next states (corresponding to the tops and bottoms of the distillation column). Instead of the next state coming immediately following the state, D, the deque of next states (unconnected streams) is sequentially stepped through. The episode is

over when either (1) the maximum number of steps are reached or (2) all of the leaf node states are terminal, which is equivalent to D being empty. This tree structured decision process can also be interpreted as a partially observable MDP where the agent observes the current stream that it is separating, and both the outlet streams that the separation produces instead of observing the whole system. Total episode reward can still be calculated by summing the rewards over all time steps. However, the value of a state is now given by the immediate reward, and the value of all of the future rewards of states downstream in the tree decision process structure - as shown by the recursively defined value function below.

$$V(s)_\pi = E_{s\prime\sim P,a\sim\pi}\left[r + \gamma V\left(s'_{\text{tops}}\right)_\pi + \gamma V\left(s'_{\text{bottoms}}\right)_\pi\right] \tag{8}$$

3.3 Distillation train synthesis examples

3.3.1 Overview

Two example problems are given below, both largely based (in terms of feed composition and property package definitions) on examples from the ChemSep example page [11] and are common distillation problems. In both problems a Soft Actor Critic agent was adapted to include the tree-MDP, using the adapted soft Q-function, as shown in the recursively defined Q-function below.

$$Q(s,a)_\pi = E_{s'\sim P,a,\sim\pi}\left[r + \gamma\left(Q\left(s'_{\text{tops}}, a'_{\text{tops}}\right) - \alpha H\left(\pi\left(s'_{\text{tops}}\right)\right)\right. + \gamma\left(Q\left(s'_{\text{bottoms}}, a'_{\text{bottoms}}\right) - \alpha H\left(\pi\left(s'_{\text{bottoms}}\right)\right)\right] \tag{9}$$

The agents action of whether or not to separate a specific stream is determined by whether the soft Q-value was positive (choose separate) or negative (choose not to separate). A consequence of this is that the Q-value of the next state (used in Equation 9), has a lower bound of zero - because in situations where the Q-value was negative, the agent would decide not to separate the stream, causing the state to be a terminal leaf. Because the remainder of the actions (column specification) are continuous, this addition allowed for a simple extension to add the binary decision of deciding whether or not to separate a given stream.

In both examples the agent was able to learn within the environment, to produce better and better distillation train configurations. In Figure 3 and Figure 5, this is shown by both the improvement in average return, and the improvement of the "best designs" (shown by the peaks in the return). In both examples, the agent is able to achieve progressively higher sum total revenue (through obtaining higher product recovery) and lower sum total TAC. For each example, autogenerated block-flow-diagrams (BFD) of the best design throughout training are given, as well as the key outcome metrics. Overall, these problems successfully demonstrate that the framing of process synthesis as a reinforcement learning task.

3.3.2 Problem 1: BTX separation

User specification of problem

- **Simulation System**: Benzene, Toluene and p-Xylene. Flowsheet settings copied from COCO file on ChemSep example page [11]
- **Starting stream definition**: Equimolar (3.35 mol/s of each compound) feed at 25°C, 1 atm
- **Product definition**: Required purity of 95%, price ($/tonne): 488, 488, 510 [12]

Results

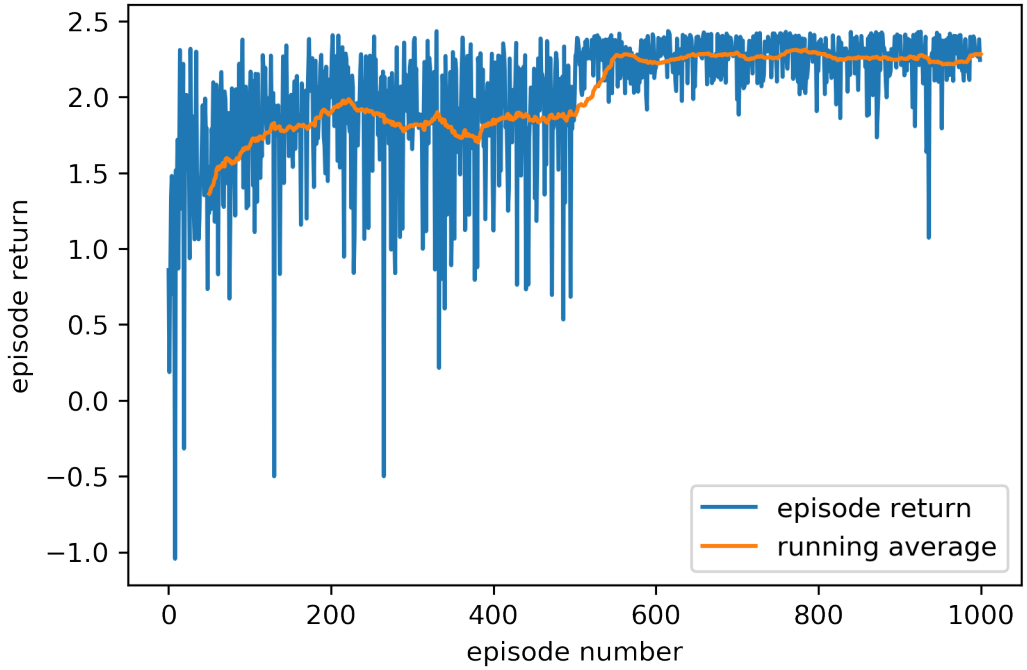

Figure 3: Agent training on BTX problem

Table 1: Performance metrics for best design outcome on BTX problem

Total Revenue	$ 13.17 million
Total Column TAC	$ 0.45 million

Table 2: Recoveries for best design outcome on BTX problem

Compound	Recovery
Benzene	98.2%
Toluene	99.7%
p-Xylene	> 99.9%

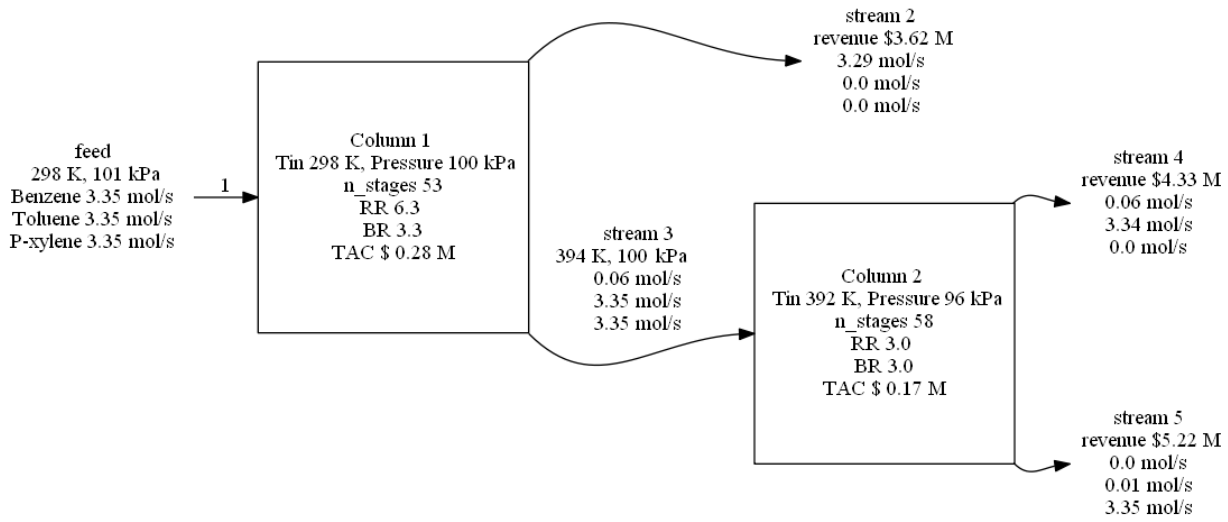

Figure 4: Best Design for BTX problem (autogenerated BFD)
For simplicity, the autogenerated BFD has a single block for the valve-distillation column pair

3.3.3 Problem 2: Hydrocarbon separation

User specification of problem

- **Simulation System**: Ethane, Propane, Isobutane, N-butane, Isopentane, N-pentane. Flowsheet settings copied from COCO file on ChemSep example page [11].
- **Starting stream definition** : Compound flows (mol/s): 17, 1110, 1198, 516, 334, 173. Conditions: 105°C, 17.4 atm.
- **Product definition**: Required purity of 95%, price ($/tonne): 125, 204, 272, 249, 545, 545 [13]

Results

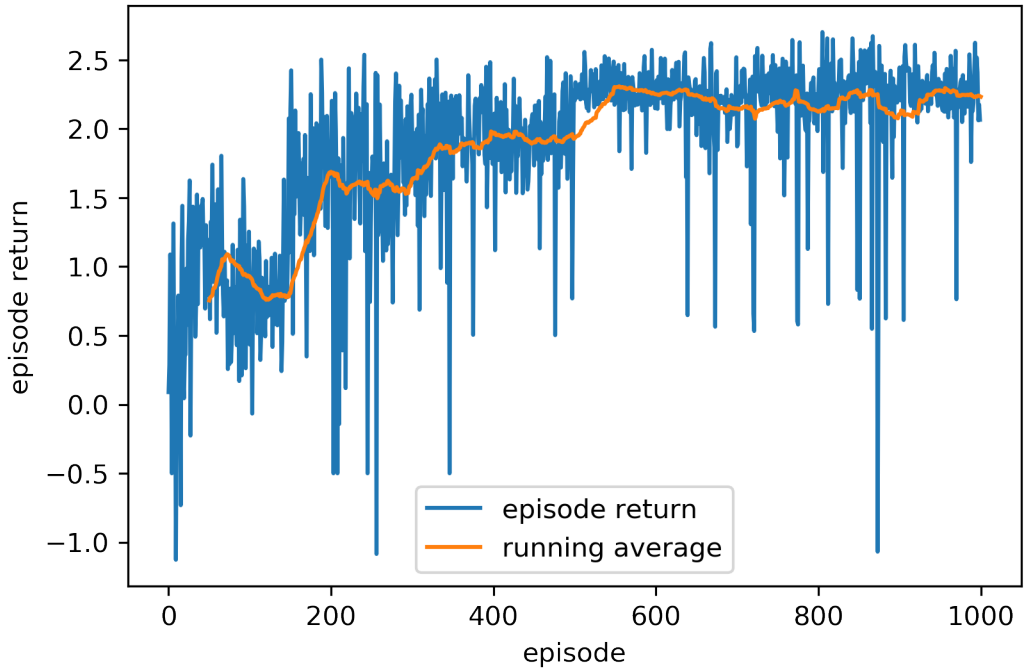

Figure 5: Agent training on Hydrocarbon problem

Table 3: Performance metrics for best design outcome on Hydrocarbon problem

Total Revenue	$ 1588 million
Total Column TAC	$ 119 million

Table 4: Recoveries for best design outcome on Hydrocarbon problem

Compound	Recovery
Ethane	0%
Propane	98.9%
Isobutane	97.3%
N-butane	91.1%
Isopentane	99.6%
N-pentane	97.0%

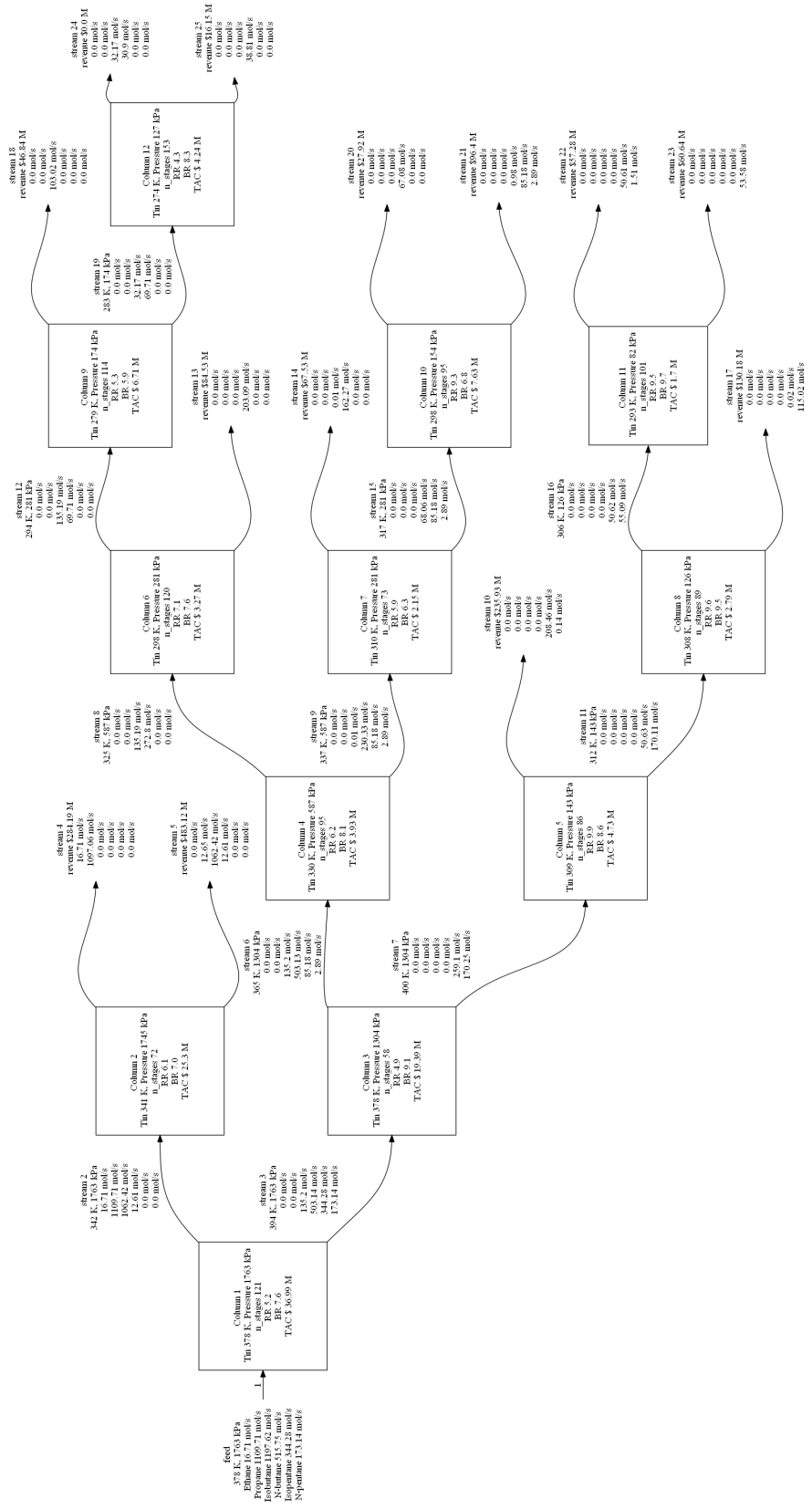

Figure 6: Best Design for Hydrocarbon problem

4 Further developments & Concluding Remarks

Distillation Gym and the corresponding examples have purposely been designed to be simple process synthesis tasks, as the purpose of Distillation Gym is primarily as a demonstration of the possibility of RL for process synthesis. The key functions that Distillation Gym demonstrates are (1) structuring process synthesis as a RL task (2) application of a RL agent, (3) interface with a process simulator (COCO simulator), (4) generality through user specified problem definition, (5) reasonable task complexity (branching process configuration, multivariate unit specification).

The next step in the continuation of this project; Chemical Engineering Gym, an all-purpose software toolkit for generating process synthesis problems, structured as reinforcement learning tasks. Extending the RL for process synthesis approach to tasks is where RL may be advantageous relative to other conventional computational approaches, due to the ability of RL to solve more open-ended problems. More fundamentally, this paper proposes that the framing level of "process synthesis as a sequence of decisions" is more apt for general process synthesis problems than the framing as "optimizing a set of process parameters/equations", which conventional optimization techniques follow.

The key improvement that Chemical Engineering Gym would add, would be to allow problem specifications with a far larger action space. The problem specification should be able to include all of the relevant actions that a human process designer takes within a simulator's GUI when designing a process. This would include allowing the agent to add a greater variety of unit operations to a process (e.g. reactors, heat exchangers), allowing the agent to add streams to the process (e.g. selecting when to "buy" raw materials, specified by a price dataset), allowing the agent greater ability to manipulate flowsheet topology (e.g. adding recycle streams), allowing the agent to select which streams exit the process (aided with a dataset of product prices/waste removal prices), editing existing unit parameters etc. In the extreme, with a large database of chemical prices, a reinforcement learning agent could be simply tasked with searching for novel profitable process designs, starting from a blank flowsheet, within the bounds of what can be accurately simulated.

Chemical Engineering Gym would require a commercial process simulator(s) to interface with, in order to run the process simulation. Currently there is a large room for process simulators to improve interface with external programs (e.g. python). Process simulators do have some interface functionality with external programs, specifically they provide the ability to edit existing unit parameters, run the process simulation and retrieve results. However, much of the functionality in process simulator GUI is left out, most notably changing the flowsheet topology (e.g. through adding new units), which is a key component of process synthesis. Adding improvements to the interface between process simulators and eternal programs would most likely have benefits to the field of computational process synthesis in general.

Progress in within the field of reinforcement learning has been significantly aided by freely available general purpose RL toolkits/frameworks, like OpenAI Gym [14]. Similarly, creating a toolkit for RL within a process synthesis context could greatly aid research within the field. Distillation Gym presents the key fist step towards such a toolkit.

5 Acknowledgements

Jasper van Baten (COCO) and Harry Kooiman (ChemSep) for help on process simulation

References

[1] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, and Thore Graepel. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. Publisher: American Association for the Advancement of Science.

[2] Rui Nian, Jinfeng Liu, and Biao Huang. A review On reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, August 2020.

[3] Qi Chen and I.E. Grossmann. Recent Developments and Challenges in Optimization-Based Process Synthesis. *Annual Review of Chemical and Biomolecular Engineering*, 8(1):249–283, 2017. _eprint: https://doi.org/10.1146/annurev-chembioeng-080615-033546.

[4] Laurence Midgley and Michael Thomson. Reinforcement learning for chemical engineering process synthesis. Technical report, Zenodo, November 2019. https://zenodo.org/record/3556549#.X2mx1mhKhPY.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv:1801.01290 [cs, stat]*, August 2018. arXiv: 1801.01290.

[6] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, and Pieter Abbeel. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[7] Hado V. Hasselt. Double Q-learning. In *Advances in neural information processing systems*, pages 2613–2621, 2010.

[8] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[10] Jasper van Baten and Richard Baur. COCO - the CAPE-OPEN to CAPE-OPEN simulator, 2020. https://www.cocosimulator.org/.

[11] Harry Kooijman and Ross Taylor. Chemsep, 2020. http://www.chemsep.org/index.html.

[12] Echemi: Provide Chemical Products and Services to Global Buyers, 2020. https://www.echemi.com.

[13] EIA. Prices for hydrocarbon gas liquids - U.S. Energy Information Administration (EIA), 2018.

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.