Control of A Polyol Process Using Reinforcement Learning

Wenbo Zhu [*] **Ricardo Rendall** [**] **Ivan Castillo** [***]
Zhenyu Wang [***] **Leo H. Chiang** [***] **Philippe Hayot** [****]
Jose A. Romagnoli [*]

[*] *Department of Chemical Engineering, Louisiana State University, Baton Rouge, Louisiana, 70803, United States*
[**] *Chemometrics and AI, Dow Inc., Terneuzen, The Netherlands*
[***] *Chemometrics and AI, Dow Inc., Lake Jackson, Texas, 77566, United States*
[****] *Alkoxylation Technology Center, Dow Inc., Terneuzen, The Netherlands*

Abstract: Reinforcement learning is a branch of machine learning, where an agent gradually learns a control policy via a combination of exploration and interactions with a system. Recent successes of model-free reinforcement learning (RL) has attracted tremendous attention from the process control community. For instance, RL has been successfully applied in very complex control tasks (e.g., games such as chess or Go that contain large state spaces) and is shown to be robust to uncertainties. These findings indicate that there is a significant potential to leverage RL methods to improve the control of chemical processes. In this work, RL was applied to a detailed and accurate simulation of an industrial polyol process. To manufacture the desired product, the RL controller is required to achieve the target ending conditions determined by four key parameters; meanwhile, economic factors are also considered in this process, including batch reaction time and total feed amounts. The obtained results show a high consistency between RL and the current optimal operating conditions. Additionally, an improvement opportunity was identified by extending current control bounds of the manipulated variables. This work illustrates that RL is capable of handling complicated industrial systems, even under realistic operating constraints.

Keywords: Deep Learning, Reinforcement Learning, Batch Process

1. INTRODUCTION

Reinforcement learning (RL), as one of the major machine learning paradigms, has delivered numerous success stories in the past decade. The origin of RL can be traced back to Sutton (1985), which aims to solve the dynamic optimization problem without the knowledge of an exact mathematical model of the target environment. In 2013, RL was first combined with deep neural networks by Mnih et al. (2013) in the Atari game challenge and achieved a similar level as a human player, which opened the prelude of deep RL. Two years later, the AlphaGo algorithm defeated one of the best human Go players, Lee Sedol, and attracted attention from the global media. After that, RL received tremendous interest in various decision-making and control tasks, including robotic arm controlling (Haarnoja et al. (2018)), behavior imitation (Brys et al. (2015)), autonomous driving (Pan et al. (2017)), and optimizer design (Li and Malik (2017)).

The objective of RL is to train a smart agent that can automatically learn the optimal control policy by self-exploration of the target system. Through exploration, the agent can collect important data and learn the optimal policy by maximizing the cumulative rewards from the target system, which composes a data-driven decision-making framework. RL has generated great interest from the control community for process control applications. Spielberg et al. (2019) utilized the actor-critic algorithm to accomplish a setpoint control task for continuous processes, which gives promising results through testing in different systems. Ma et al. (2019, 2020) and Li et al. (2018) presented the application of RL for the control of polymerization reactions, demonstrating the capability of RL for nonlinear chemical systems. Rather than directly replacing the entire control loop, the patent from ExxonMobil (Badgwell et al. (2019)) demonstrates the approach to incorporate DRL into the existing PID control systems, which can adaptively tune the PID parameters to improve control performance. In addition to control applications, Petsagkourakis et al. (2020) and Zhou et al. (2017) presented a RL application for process optimization in chemical reactors.

The success of RL on above case studies provides the opportunity to apply it on complex systems. In this work, RL was applied to the control of a simulated polyol process that mimics the behavior observed in the actual industrial process. Compared to the aforementioned case studies, the industrial polyol process faces many additional challenges in reality. First, the success of the batch depends on producing a polyol product that meets the specification

requirements. This is achieved when four key parameters are within their desirable ranges at the end of a batch. Additionally, economic benefits are maximized when the batch time is reduced, given the same amount of total feed. Therefore, this work focuses on the application of RL in a simulated polyol process that takes into account all the aforementioned challenges. A RL-based controller is designed to guide the reaction into the desired ending condition; meanwhile, the batch time is optimized to achieve better economic benefits. The obtained results are further verified and compared with the current optimal operating conditions.

2. BACKGROUND

This section summarizes the history of RL development in past decades, covering key concepts from the Markov decision process to value-based and policy-based approaches. Besides these fundamental concepts, latest actor-critic algorithms and critical improvements to RL are also presented in this section. The introduction of RL follows the development path shown in Figure 1. It starts from the definition of the Markov decision process (MDP) and then different RL algorithms are addressed including value-based, policy-based, and actor-critic methods. Additionally, the multi-armed bandit in RL is discussed in this section, including the epsilon-greedy approach, stochastic policy, and entropy. Each of the above aspects of RL is marked in different colors. Fundamental concepts of RL such as MDP, value-based, and policy-based approaches are colored in blue, and the recent development including actor-critic methods and the multi-armed bandit are colored in yellow.

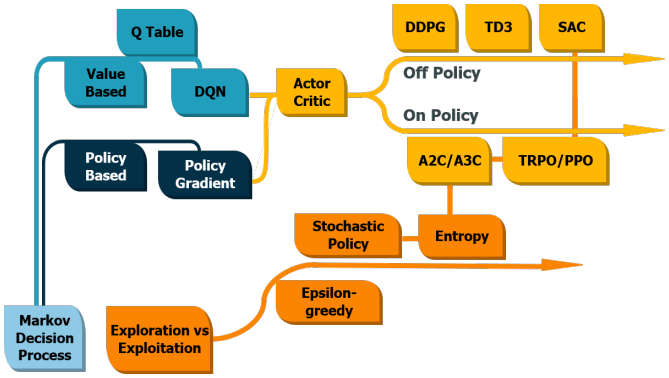


Fig. 1. Summary of RL algorithm development

2.1 Markov Decision Process

The theoretical basis of reinforcement learning focuses on the Markov decision process (MDP) (Bellman (1957)). In a Markov process, a dynamic system is defined by two components, the state S, and the transition probability P. Specifically, the state S in the Markov process should follow the Markov property that the conditional probability distribution of future states only depends on the present state, not on any past states, which is written as follows:

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, \ldots, S_t) \tag{1}$$

To formulate a decision-making objective of the Markov process, the reward function R is introduced to evaluate the return at each time, t with the state S_t. The optimization objective can be written as:

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

where γ is the discount factor and the value is usually set in the range [0, 1]. Based on Equation 2, the state value function $V(s)$ is defined to evaluate the return at each state, S_t:

$$V(s) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \tag{3}$$

Incorporated with the action (A_t) taken at each time step, the action value function can be derived from Equation 3 as follows:

$$Q(s,a) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a] \tag{4}$$

In MDP, the action (A_t) is determined by the policy, π, which is a distribution over actions given states and is defined as follows:

$$A_t = \pi(S_t) \tag{5}$$

Both value functions obey the Bellman equation.

2.2 Value-Based Methods

RL aims at finding optimal action sets for MDP. The value-based approach (Bellman (1966); Bertsekas et al. (1995)) is the most straightforward method to solve this problem by maximizing value functions. Thus, the optimal policy can be written as follows:

$$\pi^*(s) = \text{argmax}Q(s,a) \tag{6}$$

In model-free RL, value functions are estimated by trials and samples from the environment. Temporal difference (TD) learning, proposed by Sutton (1988) is a widely used method in RL for sampling and value evaluation, which outperforms the Monte Carlo method in sample efficiency. Based on this idea, Q learning (Sutton and Barto (2018)) is developed, where the action value function (Q function) is used to determine the actions with the highest expected returns. In Q learning, Q values can be stored in different cells of the table corresponding to the state and action values, which is then named as Q table method. To improve the training efficiency of the Q table, particularly the cases with high state and action dimensions, a deep neural network is introduced to represent the Q function instead of using the Q table approach. Such a method is called deep Q network (DQN) Mnih et al. (2013), which shows remarkable performance in Atari games.

2.3 Policy-Based Methods

Alternatively, the policy-based method is another approach for solving MDP with better action presentations. Instead of learning the value functions and then selecting predefined actions accordingly, the policy-based method aims at learning the policy directly. The policy can be represented by a neural network with a set of parameters θ, denoted as $\pi(a|s;\theta)$. To find the optimal policy, policy

gradient (Sutton and Barto (2018); Sutton et al. (2000)) is used to train the policy network, which is given as follows:

$$\nabla J(\theta) = \mathbb{E}_{\pi\theta}[\nabla ln\pi(a|s;\theta)Q_{\pi}(s,a)] \tag{7}$$

Nevertheless, the Q function is not explicit in the policy-based method. In the REINFORCE approach developed by Williams (1992), the Q function is approximated through the Monte-Carlo method using samples from each episode to update the parameters, θ, where the policy gradient can be written as:

$$\nabla J(\theta) = \mathbb{E}_{\pi\theta}[G_t \nabla ln\pi(a|s;\theta)] \tag{8}$$

where G_t is the score function obtained from one episode to approximate the value function.

3. METHOD

In this work, an actor-critic RL method, named soft actor-critic (SAC) Haarnoja et al. (2018) was used as the control framework for the simulated polyol process. The actor-critic RL methods were developed to further improve RL's learning efficiency, which combine the advantage of both value-based methods and policy-based methods. The actor-critic approach consists of two parts: the value network, and the policy network. The value network is the same as in the value-based approach for value function representation, which is called "critic". Parameters of the value network are usually updated by the mean squared error between the predicted value and the sampled value from the environment. The policy network is utilized for generating actions, which is called "actor". In SAC, an entropy term is considered in the loss function to regulate the randomness of the policy distribution (Mnih et al. (2016); Haarnoja et al. (2018)), which encourages the random exploration of the environment.

Due to the long time dependence of the batch process, it can hardly satisfy the Markov property. To address this partial observation issue, the memory-based RL framework (Heess et al. (2015); Peng et al. (2018)) is adopted for both of the actor and critic networks, which uses long-short term memory (LSTM) recurrent neural network to integrate historical information. The architecture of the policy network and the value network is illustrated in Figure 2. Such two branch architecture is adopted from Peng's work Peng et al. (2018), which shows great performance comparing to other structures. After tuning the size of hidden layers, the dimension number is set as 64 for both the fully connected layer and the LSTM layer.

As an off-policy RL method, SAC uses a replay buffer to store states, actions, rewards, and recurrent states for training. The value network (critic) is updated by the mean squared error between the predicted Q value, Q_p with the target Q value Q_t.

$$L_q = \frac{1}{n}\sum_{i=1}^{n}(Q_p - Q_t)^2 \tag{9}$$

where the target Q value is calculated from sampled data incorporating with entropy loss. The entropy term in the target value function encourages the RL agent to achieve a better exploration performance.

$$Q_t = r(s_i, a_i) + Q(s_i, a_i) - \alpha \log \pi(a_{i+1}|s_{i+1}) \tag{10}$$

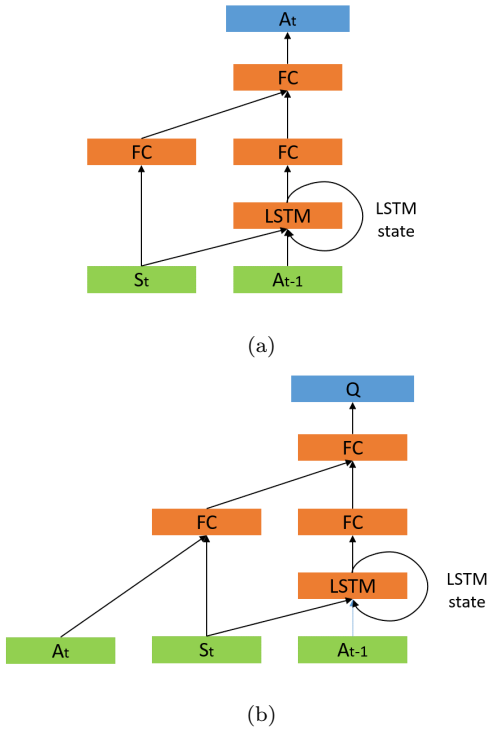


Fig. 2. Demonstration of the LSTM RL framework, (a) LSTM-based policy network, (b) LSTM-based value network. FC refers the fully connected layer with a nonlinear activation function. $LSTM$ refers the long short-term memory network. A, S, and Q represent the action, state and value for RL.

The policy network (actor) is updated by the following loss function:

$$L_{\pi} = -\frac{1}{n}(\alpha \log \pi(a_i|s_i) - Q(s_i, a_i)) \tag{11}$$

Figure 3 illustrates the training cycle of the RL controller. The training of the RL agent starts from its interaction with the reaction environment, which generates data that is subsequently stored in a replay buffer. Once enough samples are accumulated in the replay buffer, the two networks can be updated by backpropagating the loss functions. The trained agent is evaluated in a testing environment. The stochastic policy is used in the training phase for better exploration, while during model evaluation, the policy network becomes deterministic. After each episode, the training environment is reset for the next iteration.

4. CASE STUDY

The case study used in this work is a simulation of an industrial polyol process. In this batch polymerization process, epoxides (ethylene oxide (EO) and propylene oxide (PO)) are the key ingredients catalyzed by KOH in a batch reactor. The procedure of this process can be specified as follows. A low molecular weight alcohol is first mixed with the catalyst in the liquid phase. Then, monomers (EO and PO) are fed into the liquid phase with given rates. The reaction temperature is controlled by a heat exchanger, and the reactor pressure is controlled by a vent system control valve. The rigorous model of the process can be referred to Nie et al. (2013)'s work.

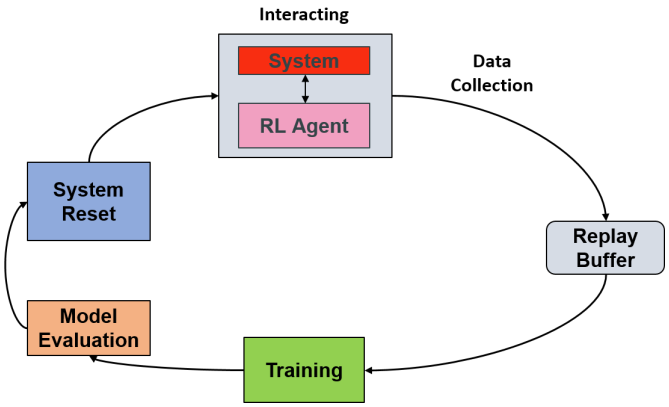


Fig. 3. Training cycle of RL

The objective of this process is to achieve the ending conditions that determine that the product is within specifications, while also minimizing the reaction time. In this process, the ending condition is determined by four key parameters: the unreacted oxide, two quality parameters, and a safety parameter. These parameters at the end of the batch ensure product quality and process safety. Besides the constraints at the end of the batch, there is another constraint on the total feed amounts of EO and PO. Due to the proprietary reasons, the detailed numerical values for this process are not disclosed.

Based on this polymerization process, the training environment for the RL controller can be designed. The three manipulated variables are the EO flow, PO flow, and temperature, where the total feed amount of EO and PO are limited. Once the allowed amount of EO and PO are used, no further EO and PO are fed to the reactor. A shaped reward function is defined to train the RL controller, which is given as follows:

$$r = \begin{cases} c/t & \text{if achieving target condition} \\ -\sum_{i=1}^{4} d(s_i, s_{target}), & \text{else} \end{cases} \tag{12}$$

where c is a positive constant, t is the time step, d is the distance function, and s are the four key parameters mentioned previously. The rewards value is regulated by the time step to differentiate the early and late achievement of the ending condition. A high value is given for early convergence, which encourages the RL agent to guide the system into the desired conditions as early as possible. The states of the RL controller include the four key parameters, and some other measurements such as pressure, molecular weights, etc. Additionally, distances from the target value of the four key parameters, as well as the available EO and PO amounts are incorporated as states of the RL controller. Thus, a total of 13 state measurements is used in the RL controller.

5. RESULTS

This section presents the results of applying RL to the simulated polyol process. The training of the SAC controller follows the procedure in Figure 3. The evaluation curve shown in Figure 4 indicates that the RL agent can learn the control policy in about five thousand episodes, where positive averaged rewards can be observed. To avoid the drop in rewards after five thousand episodes, the optimal RL policy is saved at the episode (4700) with the highest rewards value.

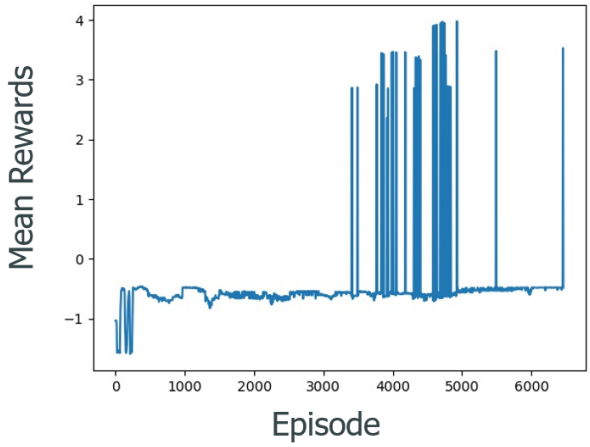
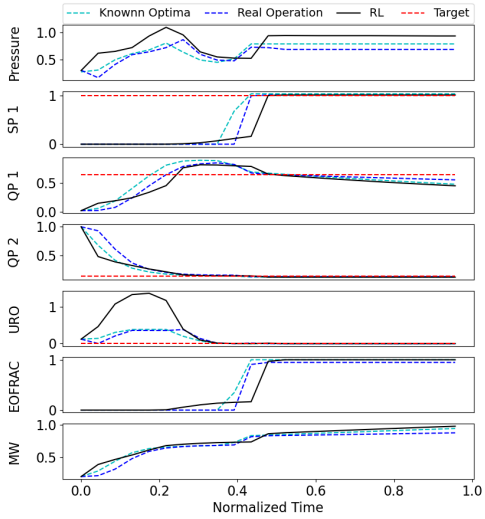


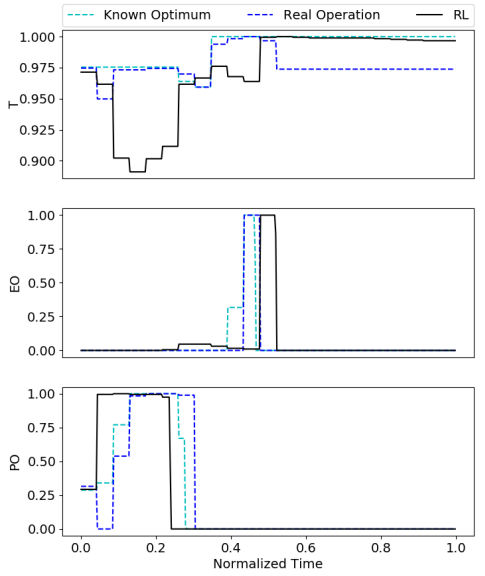
Fig. 4. Training curve of RL on the polyol process

Figure 5 illustrates the learned result, which is also compared with the computed optimal recipe from Nie et al. (2013)'s work and historical operation data from plants. The RL controller has a consistent optimum as the others, which provides similar control trajectories. In Figure 5, SP is the abbreviation of safety parameter, QP is the abbreviation of quality parameter, URO represents unreacted oxide, EOFRAC is the EO fraction, and MW is the molecular weight. The red dashed lines in the figure represent the target condition of key parameters. Achieving all of them indicates a preferred product grade and the end of the reaction. Although the temperature profile is slightly different in the beginning, the EO and PO flows are highly similar to the known optimum as well as the operation profile from the plant. The responses of key parameters shown in Figure 5a also indicate this similarity, where key parameters from RL achieve the target condition at the same time as the others.

In the current reaction system, it seems that the existing recipe has already achieved the optimal performance. Therefore, to find a possible improvement for future reaction design, the upper bounds of three manipulated variables are extended from the original design. The optimized results obtained by RL are summarized in Figure 6. The improvement of each key parameter is indicated in Figure 6, where the ending time is determined by the last key parameter (QP 1) to meet the ending condition. In this test, the reaction time to achieve the target condition is reduced by 18% compared to the original setup. The strategy used by the RL agent is to increase feed rates of reactants EO and PO into the reactor with a necessary digestion time between each feed. In this sense, the result obtained by RL suggests a possible improvement direction that the batch time of the polyol process can be reduced by increasing inlet flow rates of reactants.

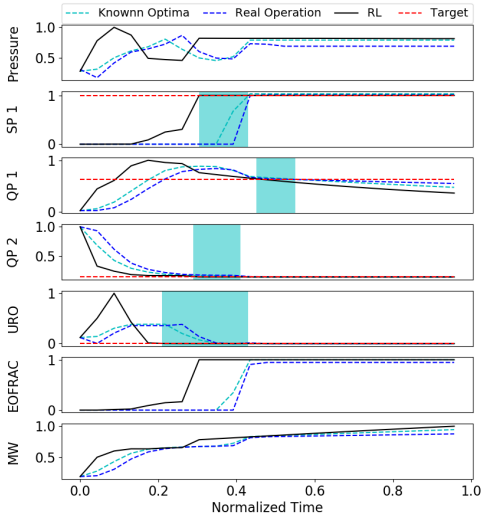


(a)

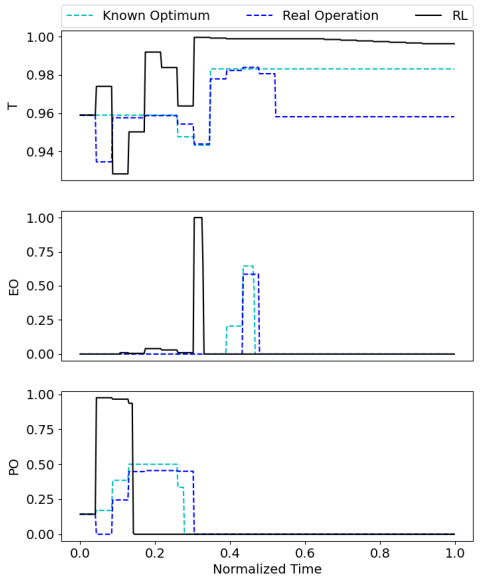


(b)

Fig. 5. Performance comparison on polyol process among RL with known optima and real plant operation in normalized scales



(a)



(b)

Fig. 6. Optimization results of RL using extended upper bounds on manipulated variables and its comparison with other two methods

6. CONCLUSION

This work applied RL in the optimal control of an industrial polyol process. The final product of the polyol process is determined by four key parameters, and meanwhile the reaction time of each batch should be reduced to obtain better economic benefits. The soft actor-critic algorithm was adopted in the polyol process to achieve such control objectives. Results in this work illustrate that RL can achieve similar performance as the traditional optimization techniques. RL also helps to understand how certain control parameters can affect the overall process through the changes in rewards. A scenario of increased feed rates is verified in RL, and the overall result is consistent with our prior knowledge, which gives us the confidence to apply it further in other processes. In this study, RL shows a promising performance in solving complex industrial control problems, and we expect to see further applications of RL in the chemical industry.

REFERENCES

Badgwell, T.A., Liu, K.H., Subrahmanya, N.A., Kovalski, M.H., et al. (2019). Adaptive pid controller tuning via deep reinforcement learning. US Patent App. 16/218,650.

Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, 679–684.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731), 34–37.

Bertsekas, D.P., Bertsekas, D.P., Bertsekas, D.P., and Bertsekas, D.P. (1995). *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

Brys, T., Harutyunyan, A., Suay, H.B., Chernova, S., Taylor, M.E., and Nowé, A. (2015). Reinforcement learning from demonstration through shaping. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

Heess, N., Hunt, J.J., Lillicrap, T.P., and Silver, D. (2015). Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*.

Li, H., Collins, C.R., Ribelli, T.G., Matyjaszewski, K., Gordon, G.J., Kowalewski, T., and Yaron, D.J. (2018). Tuning the molecular weight distribution from atom transfer radical polymerization using deep reinforcement learning. *Molecular Systems Design & Engineering*, 3(3), 496–508.

Li, K. and Malik, J. (2017). Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*.

Ma, Y., Noreña-Caro, D.A., Adams, A.J., Brentzel, T.B., Romagnoli, J.A., and Benton, M.G. (2020). Machine-learning-based simulation and fed-batch control of cyanobacterial-phycocyanin production in plectonema by artificial neural network and deep reinforcement learning. *Computers & Chemical Engineering*, 142, 107016.

Ma, Y., Zhu, W., Benton, M.G., and Romagnoli, J. (2019). Continuous control of a polymerization system with deep reinforcement learning. *Journal of Process Control*, 75, 40–47.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nie, Y., Biegler, L.T., Villa, C.M., and Wassick, J.M. (2013). Reactor modeling and recipe optimization of polyether polyol processes: Polypropylene glycol. *AIChE Journal*, 59(7), 2515–2529.

Pan, X., You, Y., Wang, Z., and Lu, C. (2017). Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*.

Pathak, D., Agrawal, P., Efros, A.A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 16–17.

Peng, X.B., Andrychowicz, M., Zaremba, W., and Abbeel, P. (2018). Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, 1–8. IEEE.

Petsagkourakis, P., Sandoval, I.O., Bradford, E., Zhang, D., and del Rio-Chanona, E.A. (2020). Reinforcement learning for batch bioprocess optimization. *Computers & Chemical Engineering*, 133, 106649.

Russo, D. and Van Roy, B. (2014). Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems*, 1583–1591.

Spielberg, S., Tulsyan, A., Lawrence, N.P., Loewen, P.D., and Bhushan Gopaluni, R. (2019). Toward self-driving processes: A deep reinforcement learning approach to control. *AIChE Journal*, 65(10), e16689.

Sutton, R.S. (1985). *Temporal Credit Assignment in Reinforcement Learning*. Ph.D. thesis.

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, 3(1), 9–44.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R.S., McAllester, D.A., Singh, S.P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Williams, R.J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4), 229–256.

Zhou, Z., Li, X., and Zare, R.N. (2017). Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12), 1337–1344.