https://www.scirp.org/journal/ns Natural Science, 2021, Vol. 13, (No. 9), pp: 412-424

Exploring Local Chemical Space in De Novo Molecular Generation Using Multi-Agent Deep Reinforcement Learning

Wei Hu

Department of Computer Science, Houghton College, Houghton, NY, USA

Correspondence to: Wei Hu, wei.hu@houghton.edu

Keywords: Multi-Agent Reinforcement Learning, Actor-Critic, Molecule Design, SARS-CoV-2, COVID-19

Received: August 8, 2021 **Accepted:** September 13, 2021 **Published:** September 16, 2021

ABSTRACT

Single-agent reinforcement learning (RL) is commonly used to learn how to play computer games, in which the agent makes one move before making the next in a sequential decision process. Recently single agent was also employed in the design of molecules and drugs. While a single agent is a good fit for computer games, it has limitations when used in molecule design. Its sequential learning makes it impossible to modify or improve the previous steps while working on the current step. In this paper, we proposed to apply the multi-agent RL approach to the research of molecules, which can optimize all sites of a molecule simultaneously. To elucidate the validity of our approach, we chose one chemical compound Favipiravir to explore its local chemical space. Favipiravir is a broad-spectrum inhibitor of viral RNA polymerase, and is one of the compounds that are currently being used in SARS-CoV-2 (COVID-19) clinical trials. Our experiments revealed the collaborative learning of a team of deep RL agents as well as the learning of its individual learning agent in the exploration of Favipiravir. In particular, our multi-agents not only discovered the molecules near Favipiravir in chemical space, but also the learnability of each site in the string representation of Favipiravir, critical information for us to understand the underline mechanism that supports machine learning of molecules.

1. INTRODUCTION

The COVID-19 pandemic demonstrated an urgent need for speedy methods to design drug and vaccine, and to ensure the effectiveness and safety of these products. However, achieving this goal is not an easy task as the number of drug-like molecules is estimated to be between 10^{30} and 10^{60}. In recent years, artificial intelligence (AI) has been introduced to guide more intelligent search of drug-like molecules. Re-

cent applications of machine learning techniques including supervised, unsupervised, and reinforcement learning (RL) have shown their success in this challenging area.

One unique feature of RL is that it doesn't learn from static data stored in files like supervised and unsupervised learning, instead it learns from dynamic data generated by the interaction between an agent and the environment. Many deep supervised and unsupervised learning techniques even require large datasets. This requirement can pose challenges in molecule design, and for a molecule of interest there may not be a corresponding training dataset existed. The RL approach, however, does not require predefined training datasets, instead it only needs to use a reward function. This has the potential to eventually lead to unexpected new molecules that no human has thought about so far [1]. Many current machine learning techniques are unable to effectively control the properties of the generated molecules, but RL methods can treat the design of a molecule as a computer game so an agent can learn to generate molecules with desired properties, considering desired properties as its goal in a game.

Several recent papers showed the feasibility of using reinforcement learning for molecular design [2-5]. However, most of current research used single-agent RL approach that works on a molecule at one site at a time sequentially, which does not allow for any change or improvement of the previous steps while working on the current step. In this paper, we proposed to leverage multi-agent RL for exploring chemical space, which has the advantage of optimizing all sites of a molecule concurrently by a team of RL agents.

The RNA polymerase inhibitor Favipiravir is currently in clinical trials as a treatment for infection with severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Favipiravir was first used against SARS-CoV-2 in Wuhan, China, then in other countries. In June 2020, Favipiravir received the approval in India for mild and moderate COVID-19 infections. As of the 23rd of July, 2020, there are 32 studies registered on clinicaltrials.gov to assess the effect of this drug in the treatment of COVID-19 [6]. The purpose of our work was to use multi-agent RL approach to discover the local chemical space of Favipiravir.

There are several related studies on chemical space using machine learning, including genetic algorithms [7], recurrent neural networks [8], and reinforcement learning [1]. In particular, RL agents can explore chemical space out of their own motivation, which may discover unexpected new molecules [1].

2. METHODS

Reinforcement learning as a field of machine learning adopts a trial and error learning approach. There are three important components in RL, state, action, and reward, which define the interaction between the agent and the environment. The goal of RL is for an agent to learn an optimal policy that can achieve maximal long-term reward. There are two main categories of RL, valued-based and policy-based. In the value-based, an agent learns a value function that represents long-term rewards, and then uses this function to define a policy, which means learning a policy indirectly from a value function. In the policy-based, an agent directly optimizes its policy. Actor-critic algorithms (**Figure 1**) incorporate both ideas: they have an actor to take actions (policy-based), and then use a critic to evaluate the actions (value-based) [9-11].

Our work employed a multi-agent actor-critic algorithm in which a group of actor-critic agents learn from others, and each agent needs to consider its own state as well as the states of its neighboring agents. Multi-agent learning can be divided into three major categories: fully cooperative, fully competitive, and mixed cooperative–competitive. In this paper, we used a cooperative approach, implying all the agents collaborate to maximize a common long-term reward. Our idea is simple: we not only need history to better position us for future learning, but also use our current learning outcome to change or improve our previous learning. A single-agent can only learn how to move forward based on history.

There are several challenges of multi-agent learning. In single-agent learning, the agent only interacts with the environment, while in multi-agent learning, one agent needs to interacts with the environment as well as other agents. In multi-agent learning, one agent not only has to consider the state change of the environment caused by its own actions, but also those from other agents. From one agent's view, the other agents may be considered to be part of the environment, which becomes non-stationary from one agent's

Actor-Critic (episodic and single agent) [9]
Input: a differentiable policy parameterization $\pi(a|s, \Theta)$
Input: a differentiable state-value function parameterization $v(s,w)$
Parameters: step sizes $\alpha > 0$, $\alpha_w > 0$, and discount rate γ in (0,1)
Initialize policy parameter Θ and state-value weights w
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I = 1$
  Loop while S is not terminal (for each time step):
  $A \sim \pi(\bullet|S, \Theta)$
  Take action A, observe S', R
  $\delta = R + \gamma v(S',w) - v(S,w)$ (If S' is terminal, then $v(S',w) = 0$)
  $w = w + \alpha_w \delta \nabla v(S,w)$
  $\Theta = \Theta + \alpha I \delta \nabla \ln \pi(A|S, \Theta)$
  $I = \gamma I$
  $S = S'$

Figure 1. Description of actor-critic algorithm, R is an immediate reward, $v(s, w)$ is the long-term reward (value) of being in state s, and $\pi(a|s, \Theta)$ represents the policy, which means taking action a given state s for a deterministic policy. For a stochastic policy, $\pi(a|s, \Theta)$ is a probability distribution over actions given state s.

perspective. As the number of agents increases, the size of the joint action space of all agents grows exponentially. In this paper, for any one agent, we used a local joint action space of its neighboring agents.

3. RESULTS

A common task in molecule design is to search the local chemical space around known molecules or drugs. In this work, the known drug was Favipiravir. Before a molecule can be processed by computers, it needs a representation so computers can understand its chemical information. There are several ways to encode molecules including graphs and strings. Molecular graph representation uses nodes and edges in a graph to reveal the atoms and bonds that make up the molecule, whereas string presentation uses characters for the same purpose. Another representation is using chemical descriptors to create chemical fingerprints, which are vectors encoding physicochemical or structural properties. Hashed fingerprints use a hash function to hash the vectors into vectors of fixed size usually consisting of 512, 1024, or 2048 bits [12].

In this paper, we used SELFIES (Self-Referencing Embedded Strings) string representation (version 1.0.3) which is an improvement over SMILES (Simplified Molecular Input Line Entry System) strings since an arbitrary SMILES string could represent a chemically infeasible or invalid molecular structure, whereas all possible SELFIES strings represent only valid molecules. Using SELFIES strings therefore can eliminate the common post processing step in using SMILES strings, a known deficiency of SMILES strings [13].

The alphabet of SELFIES is a collection of symbols or characters used to encode chemical structures of molecules, which is made of these elements: {N-1expl, #P, S+1expl, #P+1expl, =S-1expl, =C+1expl, S, =P+1expl, #S+1expl, =O+1expl, #P-1expl, =P, Cl, =O, C, S-1expl, P, Expl=Ring1, =S+1expl, =P-1expl, O-1expl, C-1expl, Ring3, Branch1_1, #C+1expl, Branch1_3, #O+1expl, Ring1, #N, Ring2, =C-1expl, =N, =N-1expl, Branch3_1, Br, Branch2_1, Expl=Ring3, =N+1expl, #S-1expl, O, Branch2_2, Branch3_2, P-1expl, =C, =S, #S, P+1expl, Branch2_3, N+1expl, C+1expl, O+1expl, #N+1expl, #C-1expl, Branch1_2, #C, I, Expl=Ring2, Hexpl, N, F, Branch3_3}.

For example, an Favipiravir molecule is encoded as a SELFIES string of length 21: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][C][Branch1_2][C][=O][N][Ring1][Branch1_3][C][Branch1_2][C][=O][N][F] (for clarity, square bracket is used to enclose each symbol), and its SMILES repre-

sentation is C1=C(N=C(C(=O)N1)C(=O)N)F (no square bracket is used).

This section of results has two parts: multi-agents and single-agent, so we could see the difference of the two. At the same time, we employed different reward functions in each part to get better understanding of how reward functions can affect the learning of RL agents in each part.

3.1. Multi-Agents

In each episode, a team of 21 actor-critic agents were given a random SELFIES string of length 21, and each site of this string was assigned an agent to gain maximal reward based on a given reward function, which is defined in Sections 3.1 and 3.2 respectively. The SELFIES string for Favipiravir was the target representation. The learning representation was the string that the 21 agents were updating from a random SELFIES string, used as a circular linear list when defining left and right neighboring states for a given state. The state of each agent was the character at the assigned site, and the joint state of each agent was the collection of three states, left state, state, right state, which corresponded to three agents, left agent, current agent, right agent. The action space of each agent was the alphabet of SELFIES of size 61. During training each agent j (j from 1 to 21) took one action from the alphabet, and used this action (character) to update the current character at site j. One training episode consisted of 21 updates, with one update from each agent.

3.1.1. Character and Molecule Similarity Based Multi-Agents

The range of molecule similarity between learning and target molecules in **Table 1** is (0, 1) and that of character similarity is (0, 21). To visualize the correlation between character similarity and molecule similarity in the reinforcement learning process, we scaled the molecule similarity from (0, 1) to (0, 21). Three experiments were conducted and the average of the collaborative learning outcomes from the multiple actor-critic agents were collected (**Figure 2**). To smooth the collaborative learning curves, a moving average of window size 50 was applied. The molecule similarity increased as the character similarity (**Figure 2**). Each agent's cumulative rewards based on the reward function introduced in **Table 1** are shown in **Figure 3**, with a total of 21 agents. Varied learnability of different 21 sites was observed in **Figure 3** and the 21 reward curves were not clustered together at the end.

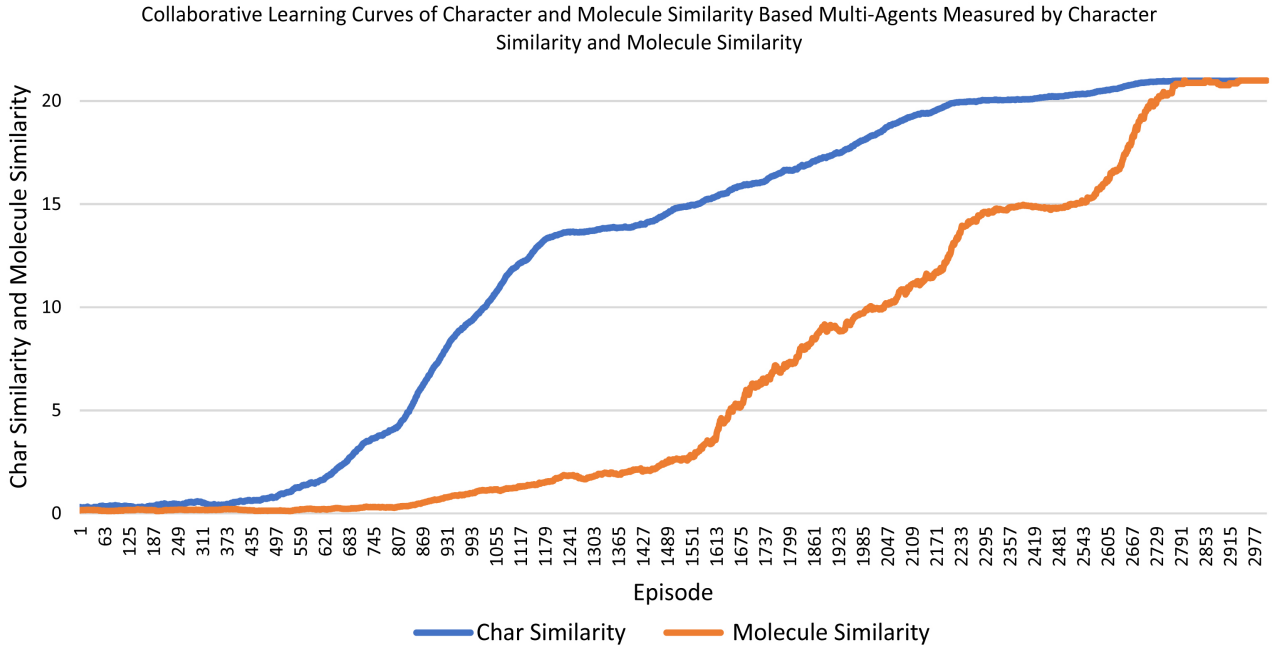

Figure 2. Collaborative learning curves of character and molecule similarity based multi-agents.

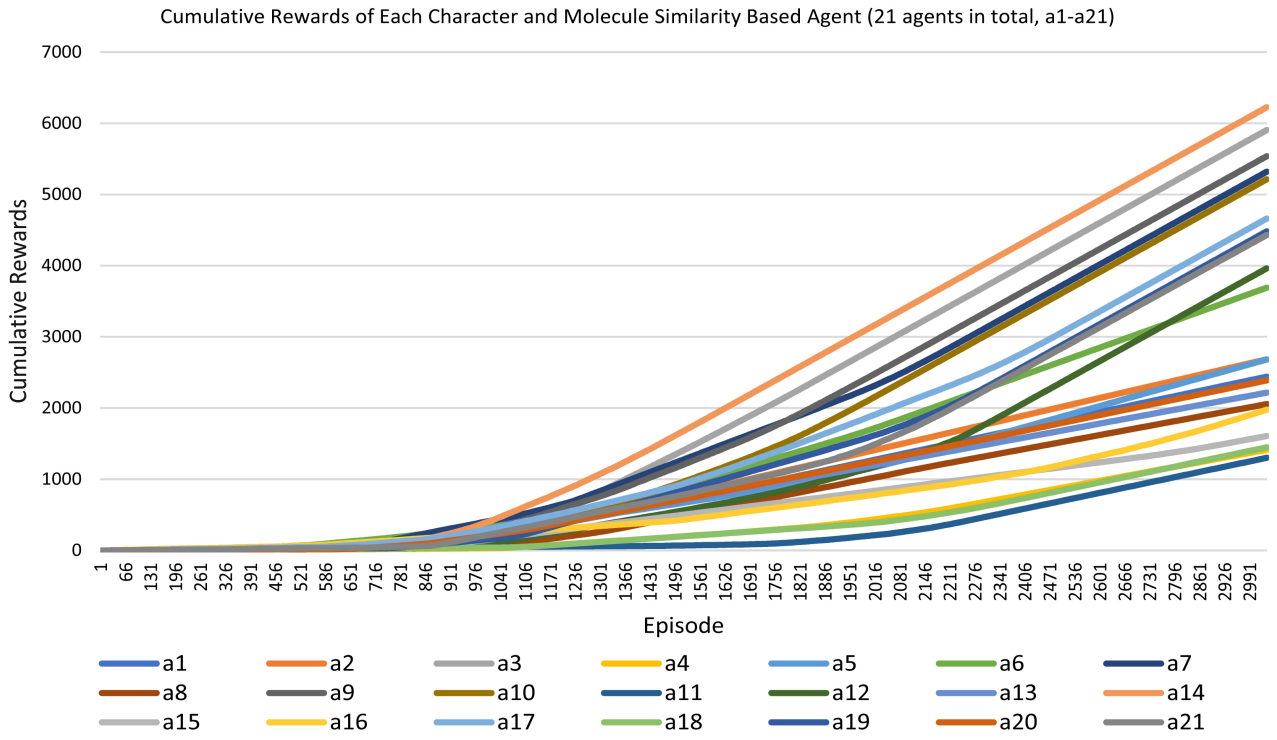

Figure 3. Cumulative rewards of each character and molecule similarity based agent trained with reward function in Table 1.

Table 1. Reward function based on character similarity and molecule similarity. The Tanimoto similarity of ECFP4 fingerprints between two molecules (we called it molecule similarity in this paper) is calculated using software RDKit with fingerprint length = 2048, and fingerprint radius = 3.

```
reward(learning_representation, target_representation, j):
    score = TanimotoSimilarity(learning_representation[left,j,right], target_representation[left,j,right],)
    score = 2.0*score
    if learning_representation[j] == target_representation[j]
        score += 1
    return score
```

3.1.2. Character Similarity Based Multi-Agents

To understand the contributions of character similarity and molecule similarity respectively to the overall collaborative learning of multi-agents, we trained 21 agents with a reward function that was solely based on character similarity (Table 2). Three experiments were conducted and the average of the collaborative learning outcomes from the multi-agents were collected (Figure 4). The correlation between the increase of character similarity and molecule similarity in Figure 4 was not as strong as that in Figure 2, which implied adding the molecule similarity in the reward function boosted the increase of molecule similarity. In other words, the multi-agents trained with character similarity and molecule similarity using the reward function in Table 1 could find molecules that were closer to the target molecule in chemical structures, a desired property of machine learning for molecules. The learnability of different 21 sites was similar as displayed in Figure 5 and the 21 reward curves were clustered together at the end, a clear contrast to the behaviors of the curves in Figure 3.

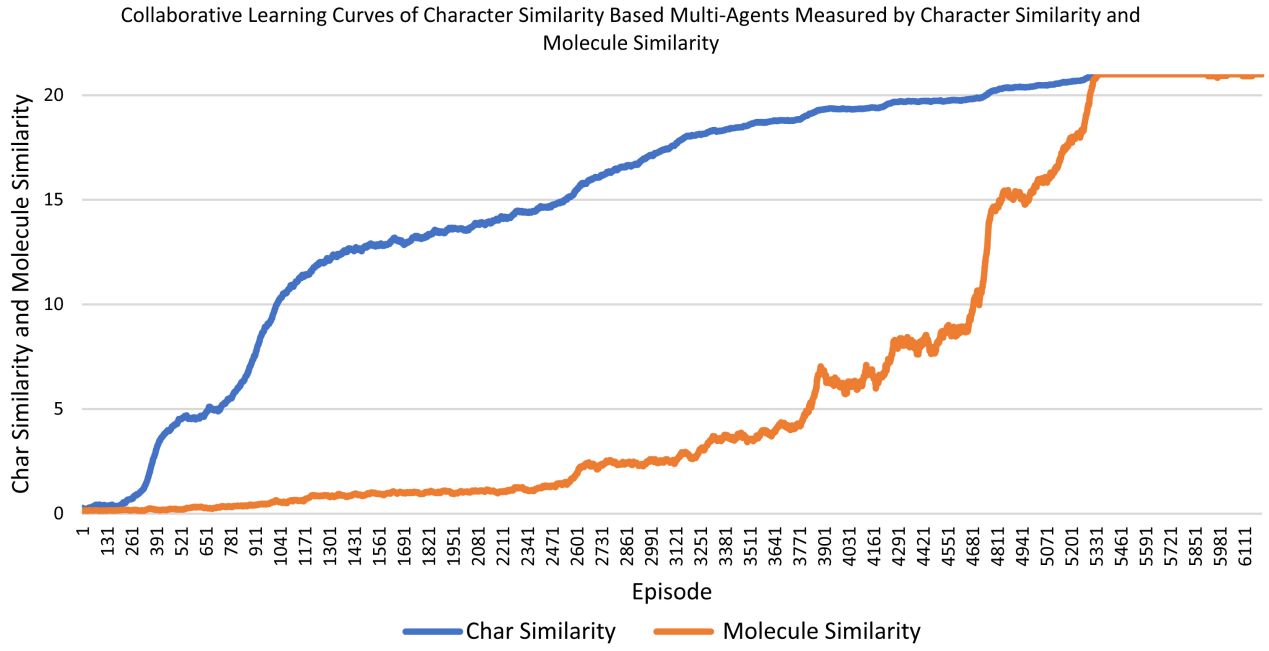

Figure 4. Collaborative learning curves of character similarity based multi-agents.

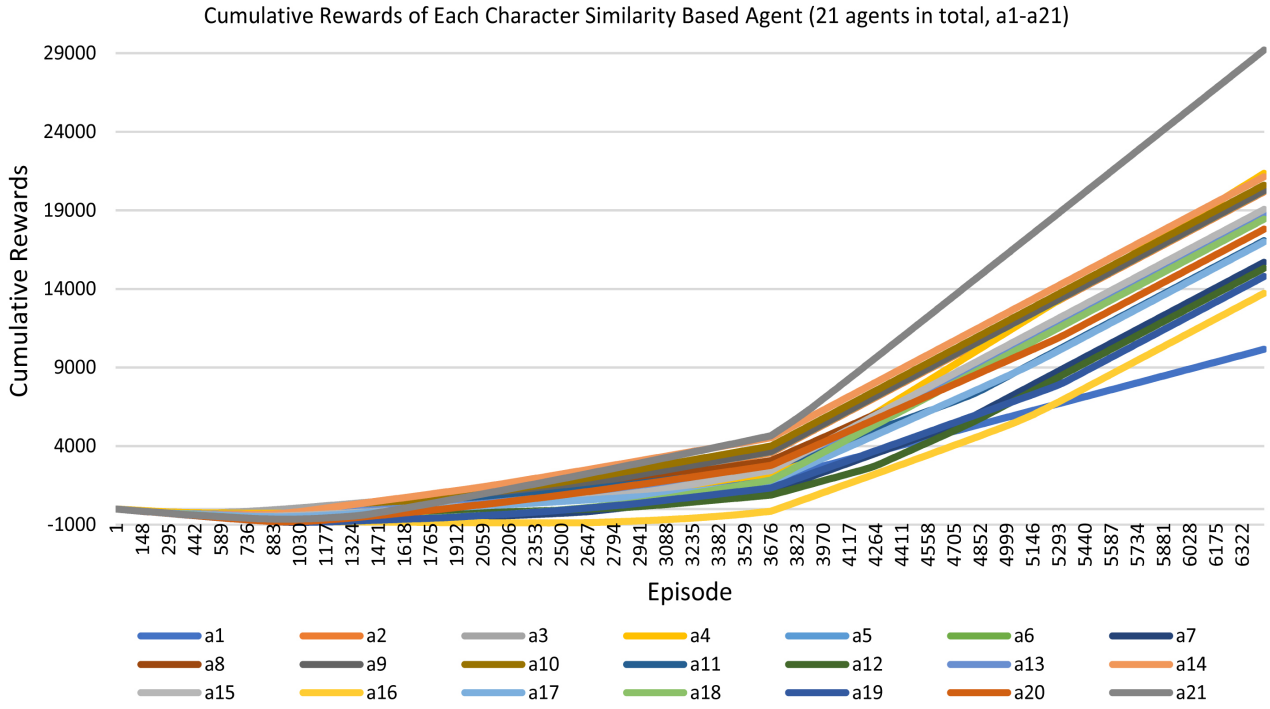

Figure 5. Cumulative rewards of each character similarity based agent trained with reward function in Table 2.

To visualize the learnability of different sites by multi-agents, we scaled the last cumulative rewards in Figure 3 and Figure 5 to the range of (0, 10) (Figure 6). In general, different sites were similarly learnable by character based agents in this section but they showed much varied learnability by character and molecule based agents in Section 3.1.1. This could be interpreted as characters in SELFIES string representation

Table 2. Reward function based on character similarity.

```
reward(learning_representation, target_representation, j):
    score = −1
    if learning_representation[j] == target_representation[j]
        score = 1
    if learning_representation[left] == target_representation[left]
        score += 1
    if learning_representation[right] == target_representation[right]
        score += 1
    return score
```

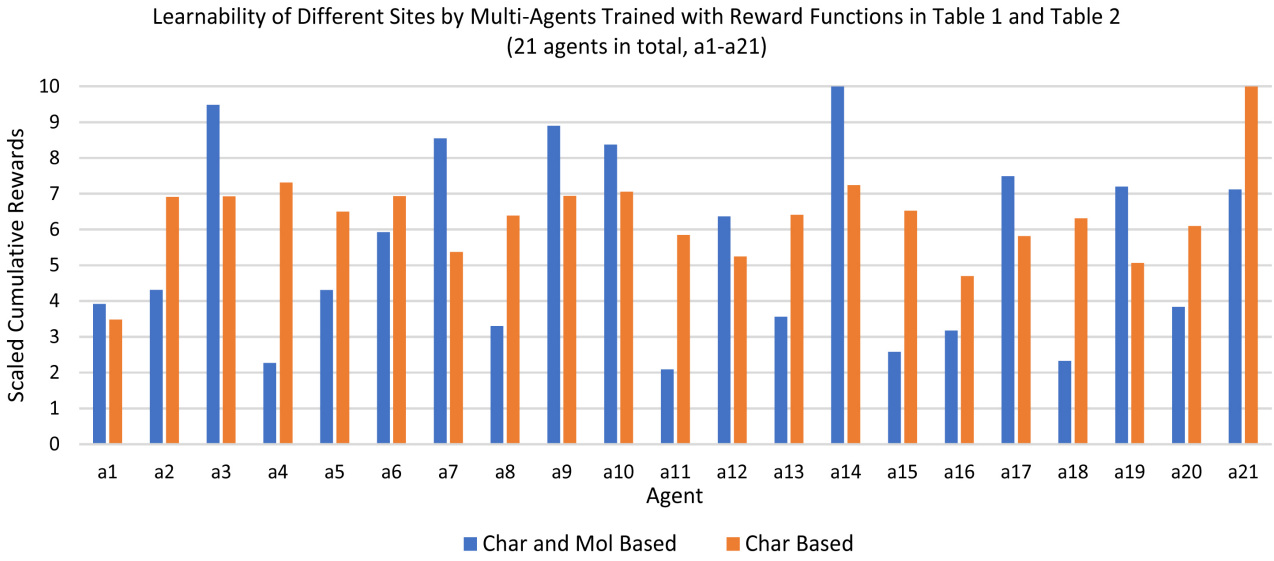

Figure 6. Learnability of different sites by multi-agents trained with reward functions in Table 1 and Table 2.

play similar roles in the learning of character based agents whereas character and molecule based agents need to learn extra chemical information so the learning is harder and more varied.

From the documents for SELFIES https://selfies.readthedocs.io/en/latest/tutorial.html, we could analyze the varied learnability of different sites by the multi-agents in Section 3.1.1 as shown in Figure 6. Let's recall that the Favipiravir SELFIES string is [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][C][Branch1_2][C][=O][N][Ring1][Branch1_3][C][Branch1_2][C][=O][N][F]. The character [P] at site 4 is after [Branch1_1] so it is used as a symbol (not as an atom) to specify the length of the branch. [P] has a Q value of 15, and therefore the length of the branch = Q + 1 = 16 that covers the 16 SELFIES characters from first [N] to the second [N] in Favipiravir, which correspond to the SMILES branch (N=C(C(=O)N1)C(=O)N) in the SMILES representation of Favipiravir C1=C(N=C(C(=O)N1)C(=O)N)F. Because [P] can have dual meanings so it is harder to learn. The same reason could be given to [Branch2_1] at site 8 in [Branch1_1][Branch2_1], because [Branch2_1] has a Q value of 6, so the length of the branch [Branch1_1] = Q + 1 = 7, which covers [C][Branch1_2][C][=O][N][Ring1][Branch1_3] that corresponds to the SMILES branch C(C(=O)N1). [Branch1_3] at site 15 has a Q value of 5, so [Ring1] has a length of Q + 1 that connects the current atom [N] with the 6th preceding atom through a single bond (Q + 1 = 6), which is C1 in the SMILES ring C1=C(N=C(C(=O)N1. In summary, these results suggested

that the symbols after branch in SELFIES were harder to learn because of their dual meanings when the 21 agents were trained with a reward function that included both character similarity and molecule similarity.

3.1.3. Chemical Structures of the Molecules Discovered by Multi-Agents

This section illustrates the molecules found by our multi-agents near the target Favipiravir in chemical space. We first display the chemical structure of the target molecule Favipiravir (Figure 7) and then several molecules discovered by our multi-agents (Figures 8-13). As expected, the molecule similarity of the learned molecule and the target decreased as the character similarity decreased. In this context, the similarity of the two molecules serves as a distance, which measures the closeness between the learned molecule and the target in chemical space.

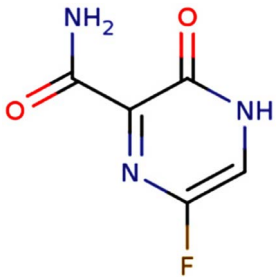

Figure 7. Target molecule: Favipiravir, SMILES: C1=C(N=C(C(=O)N1)C(=O)N)F, SELFILES: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][C][Branch1_2][C][=O][N][Ring1][Branch1_3][C][Branch1_2][C][=O][N][F].

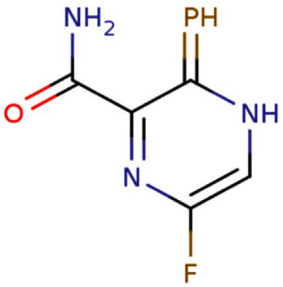

Figure 8. SMILES: C1=C(N=C(C(=P)N1)C(=O)N)F, SELFIES: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][C][Branch1_2][C][=P][N][Ring1][Branch1_3][C][Branch1_2][C][=O][N][F], Char similarity: 20, Mol similarity: 0.6571428571428571.

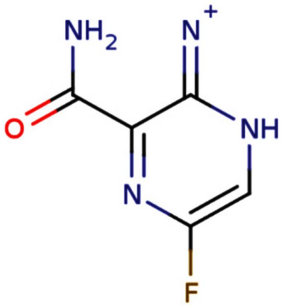

Figure 9. SMILES: C1=C(N=C(C(=[N+1])N1)C(=O)N)F, SELFIES: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][C][Branch1_2][#N+1expl][#N+1expl][N][Ring1][Branch1_3][C][Branch1_2][C][=O][N][F], Char similarity: 19, Mol similarity: 0.6571428571428571.

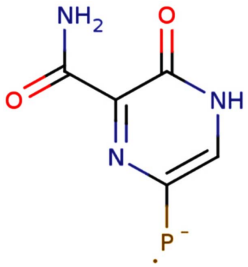

Figure 10. SMILES: C1=C(N=C(C(=O)N1)C(=O)N) [P-1], SELFIES: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][#C][Branch1_2][=O][=O][N][Ring1][Branch1_3][C][Branch1_2][C][=O][N][P-1expl], Char similarity: 18, Mol similarity: 0.6571428571428571.

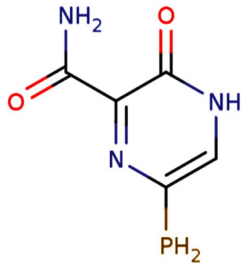

Figure 11. SMILES: C1=C(N=C(C(=O)N1)C(=O)N)P, SELFIES: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][#C][Branch1_2][#P][=O][N][Ring1][Branch1_3][C][Branch1_2][=O+1expl][=O][N][P], Char similarity: 17, Mol similarity: 0.6571428571428571.

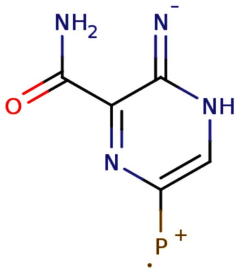

Figure 12. SMILES: C1=C(N=C(C(=[N-1])N1)C(=O)N) [P+1], SELFIES: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][#C][Branch1_2][=S][=N-1expl][N][Ring1][Branch1_3][C][Branch1_2][=O+1expl][=O][N][=P+1expl], Char similarity: 16, Mol similarity: 0.4146341463414634.

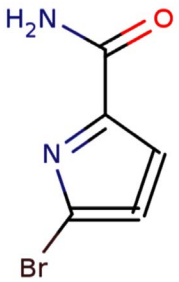

Figure 13. SMILES: C=1=C(N=C(C=1)C(=O)N)Br, SELFIES: [C][=C][Branch1_1][P][N][=C][Branch1_1][Branch2_1][=C][Ring3][Branch2_3][C][N][Ring1][Branch1_3][C][Branch1_2][#P+1expl][=O][N][Br], Char Similarity: 15, Mol Similarity: 0.23255813953488372.

3.2. Single-Agent

In this single-agent setting, at the start of training the agent was given a random SELFIES string of length 21, and the state at site j (j from 1 to 21) was the character at this site. The agent learned to take one action from the SELFIES alphabet of size 61 and used this action (character) to update the current character at j. One training episode consisted of 21 updates that went through 21 sites (from 1 to 21) of the string sequentially by the same agent.

3.2.1. Character and Molecule Similarity Based Single-Agent with Joint State Reward Function

To make a fair comparison between multi-agent and single-agent methods, the same reward function (Table 3) is used for the single-agent here as in Section 3.1 for the multi-agent experiments (Table 1). In this case, the input state of the single agent was a joint state made of three states, left state, state, right state. The SELFIES string was used as a circular linear list when defining left state and right state of the current state. Three experiments were conducted and the average of the learning outcomes from a single agent was collected. The learning curves based on the training with the joint state reward function in Table 3 are in Figure 14. As in the sections on multi-agents, a moving average of window size = 50 was used in Figure 14, and we scaled the molecule similarity from (0, 1) to (0, 21).

3.2.2. Character and Molecule Similarity Based Single-Agent with Single State Reward Function

In this section, we chose a reward function that is only meaningful in the single-agent setting (Table 4). In this case, the character at site j (j from 1 to 21) was a state, which was used as input state to the single agent. This input state was a single state as compared to the joint state in Table 3. However, the molecule similarity was measured from the first character to the current character j (Table 4). Three experiments were conducted and the average of the learning outcomes from a single agent was collected. The learning curves of the agents trained with the single state reward function in Table 4 were in Figure 15, and again a moving average of window size = 50 was used in Figure 15, and we scaled the molecule similarity from (0, 1) to (0, 21). Compared to the results of multi-agents in Section 3.1, the performance of single-agent was not as good as that of multi-agents. Furthermore, in this single-agent setting, the joint state reward function (Table 3) seemed to be better than the single state reward function (Table 4), in the learning of character similarity and molecule similarity (Figure 14 and Figure 15).

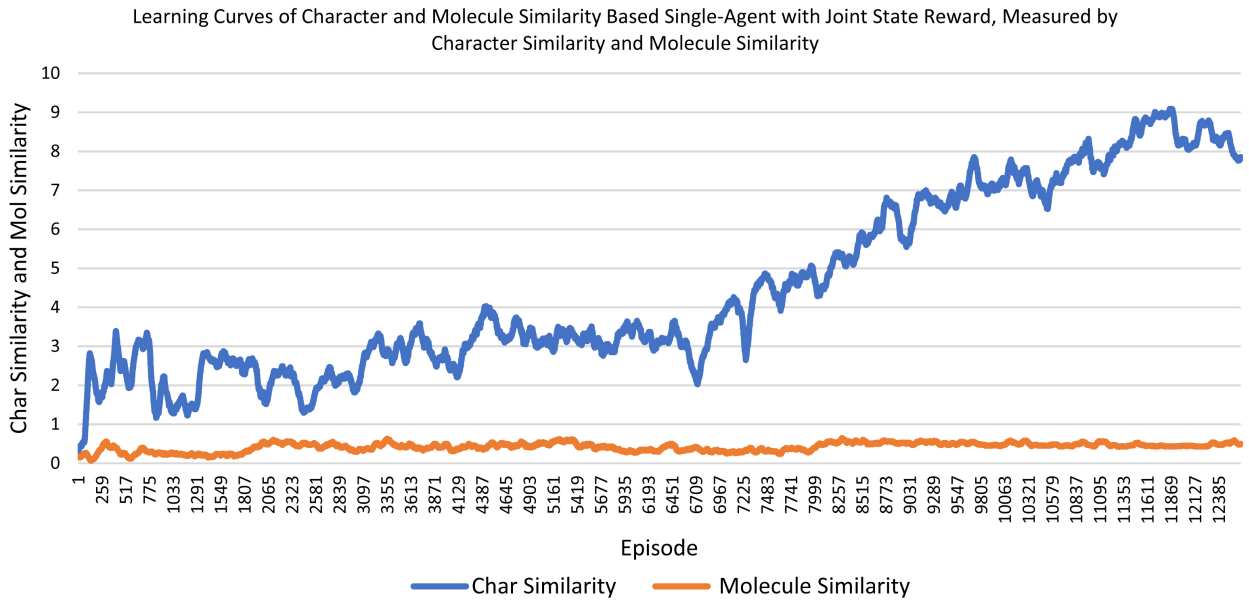

Figure 14. Learning curves of character and molecule similarity based single-agent with joint state reward function in Table 3.

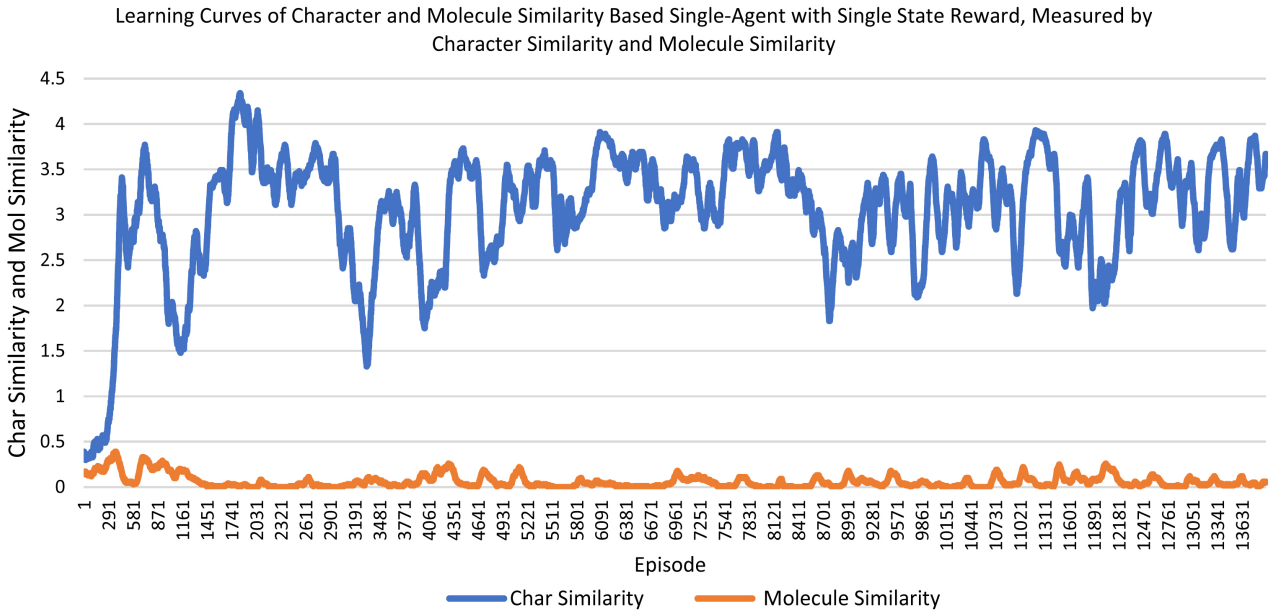

Figure 15. **Learning curves of character and molecule similarity based single-agent with single state reward function in Table 4.**

Table 3. **Reward function based on character similarity and molecule similarity (joint state).**

```
reward(learning_representation, target_representation, j):
    score = TanimotoSimilarity(learning_representation[left,j,right], target_representation[left,j,right],)
    score = 2.0*score
    if learning_representation[j] == target_representation[j]
        score += 1
    return score
```

Table 4. **Reward function based on character similarity and molecule similarity (single state).**

```
reward(learning_representation, target_representation, j):
    score = TanimotoSimilarity(learning_representation[0 to j], target_representation[0 to j],)
    score = 2.0*score
    if learning_representation[j] == target_representation[j]
        score += 1
    return score
```

Our findings reported in this section suggested that multi-agent RL approach increased learning significantly compared to single-agent RL, which could be attributed to the synergy of team work of a group of agents and a more focused individual responsibility of each agent in the group.

4. CONCLUSION

In drug discovery, study of the structural neighborhood of known molecules is critical for local optimization, and machine learning can be employed for this task to enhance structure-based molecule and

drug design. By viewing molecule design as a game, RL can be applied to de nova drug design of molecules with desired properties without using a large training dataset. The RL agents can also search chemical space without any prior knowledge, which may lead to discovery of molecules unknown before. We proposed to use multi-agent RL to study the local chemical space of Favipiravir in this work. To assess the validity of our idea, we ran our algorithm on Favipiravir molecule with a group of RL agents. Our experiments confirmed multi-agents outperform single-agent in exploring local chemical space, and showed the collaborative learning of this team of agents as well as the individual learning of each agent therein. Multi-agents exhibit the advantage of concurrent learning of all sites of a molecule, compared to the single-agent approach that can only work on one site at a time and cannot change or improve any previous sites while working on the current site. Essentially, a single-agent approach can only learn to move forward, but cannot backward. But in a real learning task, we typically need to use our current learning outcome to better inform us of how to change or improve our previous learning, in addition to using history to help us move forward.

CONFLICTS OF INTEREST

The author declares no conflicts of interest regarding the publication of this paper.

REFERENCES

1. Thiede, L.A., Krenn, M., Nigam, A.K. and Aspuru-Guzik, A. (2020) Curiosity in Exploring Chemical Space: Intrinsic Rewards for Deep Molecular Reinforcement Learning. arXiv:2012.11293 [cs.LG]
2. Neil, D., Segler, M., Guasch, L., Ahmed, M., Plumbley, D., Sellwood, M. and Brown, N. (2018) Exploring Deep Recurrent Models with Reinforcement Learning for Molecule Design. *ICLR* 2018 *Conference*, Vancouver, BC, 30 April-3 May 2018.
3. Jeon, W. and Kim, D. (2020) Autonomous Molecule Generation Using Reinforcement Learning and Docking to Develop Potential Novel Inhibitors. *Scientific Reports*, **10**, 22104. https://doi.org/10.1038/s41598-020-78537-2
4. Popova, M., Isayev, O. and Tropsha, A. (2018) Deep Reinforcement Learning for De Novo Drug Design. *Science Advances*, **4**, eaap7885. https://doi.org/10.1126/sciadv.aap7885
5. Pereira, T., Abbasi, M., Ribeiro, B. and Arrais, J.P. (2021) Diversity Oriented Deep Reinforcement Learning for Targeted Molecule Generation. *Journal of Cheminformatics*, **13**, Article Number: 21. https://doi.org/10.1186/s13321-021-00498-z
6. Agrawal, U., Raju, R. and Udwadiac, Z.F. (2020) Favipiravir: A New and Emerging Antiviral Option in COVID-19. *Medical Journal Armed Forces India*, **76**, 370-376. https://doi.org/10.1016/j.mjafi.2020.08.004
7. Nigam, A.K., Pollice, R., Krenn, M., Gomes, G. dos P. and Aspuru-Guzik, A. (2021) Beyond Generative Models: Superfast Traversal, Optimization, Novelty, Exploration and Discovery (STONED) Algorithm for Molecules using SELFIES. *Chemical Science*, **12**, 7079-7090. https://doi.org/10.1039/D1SC00231G
8. Li, X., Xu, Y., Yao, H. and Lin, K. (2020) Chemical Space Exploration Based on Recurrent Neural Networks: Applications in Discovering Kinase Inhibitors. *Journal of Cheminformatics*, **12**, 42. https://doi.org/10.1186/s13321-020-00446-3
9. Sutton, R.S. and Barto, A.G. (2018) Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning) (Adaptive Computation and Machine Learning Series). 2nd Edition, MIT Press, Cambridge, MA.
10. Duryea, E., Ganger, M. and Hu, W. (2016) Exploring Deep Reinforcement Learning with Multi Q-Learning. *Intelligent Control and Automation*, **7**, 129-144. https://doi.org/10.4236/ica.2016.74012
11. Hu, W. and Hu, J. (2020) Distributional Reinforcement Learning with Quantum Neural Networks. *Intelligent Control and Automation*, **10**, Article ID: 91668, 16 p. https://doi.org/10.4236/ica.2019.102004

12. David, L., Thakkar, A., Mercado, R. and Engkvist, O. (2020) Molecular Representations in AI-Driven Drug Discovery: A Review and Practical Guide. *Journal of Cheminformatics*, **12**, 56. https://doi.org/10.1186/s13321-020-00460-5

13. Krenn, M., Haese, F., Nigam, A.K., Friederich, P. and Aspuru-Guzik, A. (2020) Self-Referencing Embedded Strings (SELFIES): A 100% Robust Molecular String Representation. *Machine Learning: Science and Technology*, **1**, Article ID: 045024. https://doi.org/10.1088/2632-2153/aba947