PAPER • **OPEN ACCESS**

Curiosity in exploring chemical spaces: intrinsic rewards for molecular reinforcement learning

To cite this article: Luca A Thiede *et al* 2022 *Mach. Learn.: Sci. Technol.* **3** 035008

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 035008 https://doi.org/10.1088/2632-2153/ac7ddc

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
8 March 2022

REVISED
20 June 2022

ACCEPTED FOR PUBLICATION
1 July 2022

PUBLISHED
25 July 2022

PAPER

Curiosity in exploring chemical spaces: intrinsic rewards for molecular reinforcement learning

Luca A Thiede[1,*], **Mario Krenn**[1,2,3,5], **AkshatKumar Nigam**[1,2,6,7] **and Alán Aspuru-Guzik**[1,2,3,4]

[1] Department of Computer Science, University of Toronto, Toronto, Canada
[2] Department of Chemistry, University of Toronto, Toronto, Canada
[3] Vector Institute, Toronto, Ontario, Canada
[4] Lebovic Fellow, Canadian Institute for Advanced Research (CIFAR), 661 University Ave, Toronto, Ontario M5G, Canada
[5] Max Planck Institute for the Science of Light (MPL), Erlangen, Germany
[6] Department of Computer Science, Stanford University, Stanford, CA, United States of America
[7] Department of Genetics, Stanford University, Stanford, CA, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: luca.thiede@mail.utoronto.ca

Keywords: reinforcement learning, molecular design, intrinsic rewards

Abstract

Computer aided design of molecules has the potential to disrupt the field of drug and material discovery. Machine learning and deep learning in particular, made big strides in recent years and promises to greatly benefit computer aided methods. Reinforcement learning is a particularly promising approach since it enables de novo molecule design, that is molecular design, without providing any prior knowledge. However, the search space is vast, and therefore any reinforcement learning agent needs to perform efficient exploration. In this study, we examine three versions of intrinsic motivation to aid efficient exploration. The algorithms are adapted from intrinsic motivation in the literature that were developed in other settings, predominantly video games. We show that the *curious* agents finds better performing molecules on two of three benchmarks. This indicates an exciting new research direction for reinforcement learning agents that can explore the chemical space out of their own motivation. This has the potential to eventually lead to unexpected new molecular designs no human has thought about so far.

1. Introduction

The development of new drugs and functional materials is an important but expensive process. It can be framed as an optimization problem of desired properties over chemically stable and synthetically feasible molecules, denoted as inverse molecular design problem [1, 2]. The search space is enormous though [3] and therefore exhaustive search is not feasible. Therefore, various AI approaches exist to tackle this problem, including variational autoencoders (VAEs) [4, 5], generative adversarial networks (GANs) [6] or genetic algorithms [7–10].

These approaches are very promising, but except for the genetic algorithms they require a training dataset. However, not for every class of molecules, such a training dataset exists. Furthermore, the use of a dataset restricts the model to the given data distribution and thus makes it unlikely to find interesting molecules outside of that distribution.

One other approach is reinforcement learning which allows for de novo molecular design [11], potentially far away from any known data distribution [12–15]. Instead of a dataset, only a reward function is needed, that measures how good a generated molecule is. However, due to the vast chemical space, efficient exploration is necessary.

Here we take inspiration from the field of reinforcement learning (RL) for video games, where the idea of intrinsic rewards [16], which are loosely modeled after human curiosity [16–18], was leading to exceptional results, in some cases even without access to actual rewards from the environment [16, 19]. Inspired by this

line of research, we propose intrinsic motivation for molecular design and show that the most *curious agents* perform best on two out of three different benchmarks.

1.1. Reinforcement learning basics

Reinforcement learning is a technique used to find a policy π_θ parameterized by the parameters θ that maximizes the state-action trajectories in an environment. Formally the environment is described as a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mu_0, \gamma, R, T)$. Here, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, μ_0 is the initial state distribution, $\gamma \in (0, 1]$ is the discount factor. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and we set $r_t := R(s_t, a_t)$. And finally T is the maximal length of an episode. For every policy π we can define the expected reward as the reward the agent will collect when it is in a certain state $V^\pi(s_{t^*}) = \mathbb{E}_\pi(\sum_{t=t^*}^{T} \gamma^t R(s_t, a_t | s_{t^*}))$ and call this quantity the value of the state s_{t^*}, and analogously we define the Q value of an action in a state as $V^\pi(s_{t^*}, a_{t^*}) = \mathbb{E}_\pi(\sum_{t=t^*}^{T} \gamma^t R(s_t, a_t | s_{t^*}, a_{t^*}))$. The goal is to find a policy so that $J(\theta) = \mathbb{E}_{s_0 \sim \mu_0}(V^\pi(s_0))$ is maximized.

There are many different ways to train a policy. Throughout this paper we will use proximal policy optimization (PPO) [20], as it is known for its robust performance on many different tasks.

To help the agent explore the state space, PPO uses a so called entropy penalty [20] in its loss function. This penalty term encourages the policy to be more random, which is necessary, as a completely deterministic policy will never try out anything new.

2. Reinforcement learning for molecular design

For molecular design, we define the state s_t as the SELFIES [21] string (a string representation for molecules with 100% validity for any string) that is so far constructed. The action a_t is the next character to be appended to the string. The molecule is finished either when the max number of steps is reached, which we set to 35 throughout our experiments, or the agents use the *[STOP]* symbol.

For some property p that we wish to optimize, and by denoting the molecule at time step t as mol, the reward at every time step can be formulated in two ways. Either as

$$r_t = \begin{cases} p(\mathrm{mol}(T)), & \text{SELFIES string finished} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

or as

$$r_t = p(\mathrm{mol}(t)) - p(\mathrm{mol}(t-1)). \tag{2}$$

For both formulations the cumulative reward $\sum_{t=0}^{T} \gamma^t r_t = p(\mathrm{mol}(t))$ for $\gamma = 1$ is the property of the final molecule $p(\mathrm{mol}(T))$. The second formulation is more dense and therefore more informative, but also requires a lot more calculations of the reward function, while the first formulation is sparse, but only needs one calculation of the reward per trajectory.

The chemical space is huge, and therefore efficient exploration is necessary. In the next section we discuss a technique in reinforcement learning literature that is known to help with better exploration and adapt it to our use case.

3. Related work

The literature on reinforcement learning often distinguishes between intrinsic and extrinsic rewards. An extrinsic reward is anything that comes directly from the environment, while intrinsic rewards are any rewards that are generated by the agent itself. The general idea is to guide the exploration of an agent into unvisited regions of the state space, by augmenting the normal extrinsic reward with the intrinsic reward. The intrinsic reward is the higher, the more *novel* the visited state is. Formally, we formulate the total reward as:

$$r_{\mathrm{total}}(t) = r_{\mathrm{extrinsic}}(t) + \alpha r_{\mathrm{intrinsic}}(t). \tag{3}$$

There are many different approaches to define intrinsic rewards, which can be roughly divided into three basic categories [22]: prediction-based, count-based and memory-based.

Intrinsic rewards are known to help in so-called hard exploration problems. These are environments with either a sparse feedback (rewards are only provided in very specific rare states) or deceiving feedback (to get to good solutions, the agent first needs to take several steps in the state action space into a direction, which initially provides very low or even negative rewards).

We now give a short overview of the different approaches of intrinsic reward formulation.

3.1. Count based strategies

A canonical choice to quantify a state's novelty is to count how often the agent already visited the state. A visit count of 0 means it is entirely novel, and it becomes less novel the higher the count. However, for very high dimensional or continuous states, collecting sufficient statistics by naively counting becomes infeasible due to the curse of dimensions.

There are several approaches to combat this. In [23], the authors introduced a pseudo count function $\hat{N}_n(s)$, based on a density a learned model $\rho_n(s)$. The intrinsic reward is then defined as $(\hat{N}(s) + 0.01)^{-1/2}$ as inspired by classic count based motivation techniques [24].

Another option to use count-based exploration in continuous or high dimensional spaces is to project the states down to some lower dimensional space using locality sensitive hashing (LSH) [25] such as SimHash:

$$\phi(s) = \mathrm{sgn}(A \cdot g(s)) \tag{4}$$

where sgn is the sign function, $A \in R^{k \times D}$ is a random matrix where each entry is i.i.d. from a standard Gaussian, and g: $\mathcal{S} \to R^D$ is an optional pre-processing function (for example the encoder part of a VAE, if the state s is a video game frame). LSH are popular hashing functions that preserve distances of vectors, which means that if two vectors s_1 and s_2 are close, than $\phi(s_1)$ and $\phi(s_2)$ are also close. If k is chosen small enough, it is possible to collect sufficient statistics for empirical counts, and the intrinsic reward can again be chosen as $r_{\text{intrinsic}} = N(\phi(s))^{-1/2}$.

3.2. Prediction based strategies

The basic idea of prediction-based exploration strategies is to predict some function of the state the agent is in and the action it took and use the prediction error as an intrinsic reward. This rewards the agent for going to regions of the state-action space, where it has not yet understood the environment, and therefore makes large prediction errors.

Some early work by [26] encoded the state with an encoding function $\phi(s_t)$ and used this and the action taken at timestep t to predict the encoded state $\phi(s_{t+1})$. The prediction error $e_t = ||f(\phi(s_t), a_t) - \phi(s_{t+1})||_2$ is then normalized by the biggest encountered prediction error $e_t = \frac{e_t}{\max_{i<t} e_i}$ to obtain the final intrinsic reward. The authors used an autoencoder as the encoding function ϕ to reduce the dimension, as predicting pixels is too high dimensional and difficult.

Pathak *et al* [16] observed that encoding the state with an autoencoder can also encode unpredictable information that the agent does not have any power over. For example, if the agent looks at a tree whose leaves are moving in the wind, it is not reasonable to predict how the leaves will move in the next frame, given the current frame and action, since the agent does not cause the movement. To overcome this issue, the authors do not learn ϕ with an autoencoder, but with an inverse dynamics model, that makes sure only information that can be controlled in principle are used in the prediction error calculation.

3.3. Memory based strategies

Another paradigm is to explicitly store observations in memory and to use new states' dissimilarity to old states in memory as the intrinsic reward. For example, Badia *et al* [27] were embedding states with an inverse dynamics model $\phi(s)$ as described above and then used

$$r_{\text{intrinsic}}(t) = \frac{1}{\sqrt{\sum_{\phi \in N(\phi(t))} K(\phi_i, \phi(s_t))} + \epsilon} \tag{5}$$

where $K(\cdot, \cdot)$ is a kernel function measuring similarity. $K(\cdot, \cdot)$ could for example be the inverse Euclidean distance. This means, for every new state, we look in the memory of already encountered states, which ones are the most similar. We call these close states the neighborhood $N(\phi(t))$. We then take the average similarity of the new sample to its neighboring states. Thus, the more different a state is to all states in memory, the smaller $K(\cdot, \cdot)$ and therefore the bigger the intrinsic reward, leading the agent to seek out states that are unlike any in the memory.

4. Is molecular design a 'hard' exploration problem?

It is known that intrinsic rewards help in so called 'hard' exploration problems. Thus, it makes sense to first establish whether any of the commonly used benchmarks really are hard exploration problems. Since rewards are never sparse (once a molecule string is finished, the agent always gets a reward), this question boils down to whether rewards are deceiving. To determine whether a task has deceiving rewards is much harder to determine for molecular design as compared to, for example, video games. In video games, we can often see

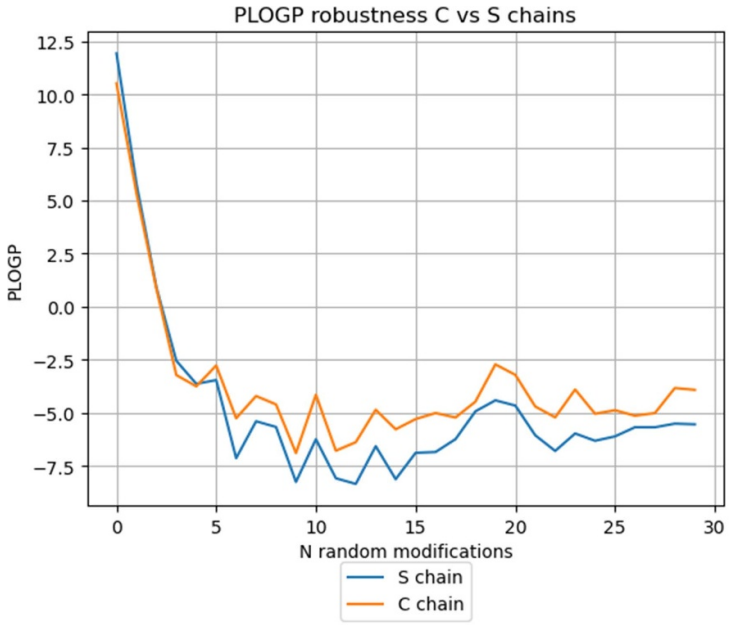

Figure 1. The robustness of the pLogP property to random replacements of N symbols in the SELFIES string of two known local optima, the sulfur, and the carbon chain. We can see, that for 0 and up to 3 mutations, the sulfur chain has the higher pLogP value. However, for more mutations, the carbon chain has the better pLogP value. This means the carbon chain is the more robust local optimum.

whether there are naive strategies that will increase the reward short term, but not long term. For example, picture a side-scrolling game like Super Mario. The reward might be to make it as far to the right as possible, as the end of the level is always on the right. At some point, there might be a door where a key is needed. To get the key, one must first go to the left, though, reducing the reward. Therefore the reward is deceiving. However, this is not as easy to see with molecular properties since we do not have such an intuitive understanding of the chemical space as we have for the state space of video games. Fortunately, in some special cases, we can show that there are indeed deceiving rewards. For this, we consider the pLogP property. We know from previous experiments [8], that there are two good local optima: the carbon chain and the sulfur chain. The full sulfur chain of length 35 has a pLogP of around 12.5, and the full carbon chain of length 35 has a pLogP of around 10. However, if we randomly perturb n character in the SELFIES string with random other SELFIES characters, the pLogP value of the sulfur chain on average falls much faster compared to the pLogP value of the carbon chain (see figure 1). Thus, when the agent is still in the training process and makes mistakes, adding carbon symbols will yield, on average, a higher final reward than adding sulfur symbols. This means pLogP has a deceiving reward structure. Unfortunately, for the other benchmarks we are using, such an analysis is not possible, and we have to rely on experiments. This phenomenon of deceiving rewards is connected to another problem of RL for molecular design. RL optimizes for the expected reward. However, we are genuinely interested only in the molecule τ^* with the maximum reward. Since the agent's policy is stochastic, optimizing for the expected reward does not in general, optimize for the reward. Formally this means:

$$\arg\max_{\tau} \mathbb{E}_{\arg\max_{\pi} \mathbb{E}_{\pi}(R(\tau'))}(R(\tau)) \neq \arg\max_{\tau} R(\tau) \tag{6}$$

for a non-deterministic policy π. Standard RL methods used in the literature optimize the left-hand side of equation (6) (corresponding to the orange curve in figure 2). However, we are interested in the right-hand side, as, in molecular design, we care only about the best or best few (corresponding to the blue curve in figure 2). The reason for equation (6) is that for some reward functions, the expected reward for a stochastic policy is higher for state action trajectories, where the agent can make more mistakes and still get a high reward, compared to the state-action trajectory that represents the optimal solution, if this optimal solution is highly sensitive to errors. In other words, if a state action trajectory has only a high reward, if a specific sequence of steps is followed (for example, a pure sulfur chain), and if the reward drops off very fast for that solution if the agents makes just few deviations, the expected reward will be lower compared to a solution with a lower reward for the locally optimal solution (a carbon chain), if the reward does not drop as dramatically for some errors in the optimal state-action trajectory.

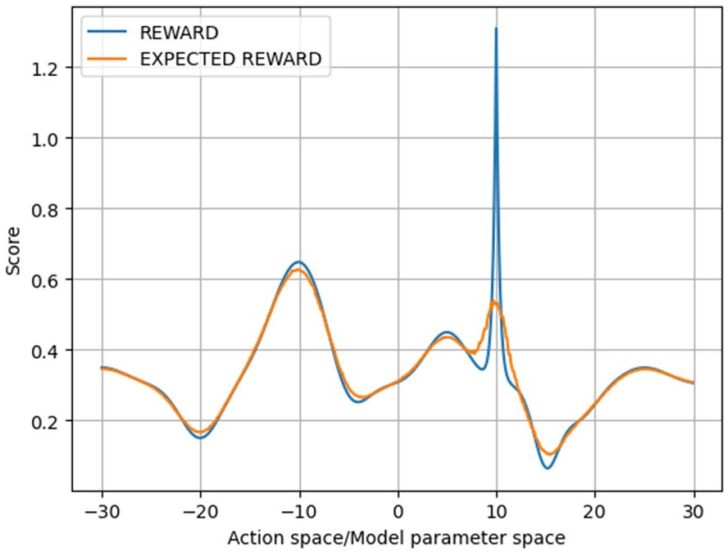

Figure 2. A one-step environment's reward surface: the model samples its actions from a Gaussian (not shown), with a fixed standard deviation. The x-axis is simultaneously a real-valued action and the mean of the Gaussian. At every point, we sample 100 actions and average their reward. This yields the orange expected reward surface. Obviously, the maximum of the expected reward is not coinciding with the maximum of the reward.

Figure 2 illustrates this problem for a one-dimensional continuous environment. Here, a model samples its actions from a Gaussian with a fixed standard deviation. Note that the model parameter (the mean of the Gaussian) and the sample are in the same domain on the x-axis. For each position of the Gaussian, the model is sampled 100 times, and the rewards of the samples are averaged. This yields the expected reward for a given model parameter. Obviously, the peak of the reward is not at the same location, where the peak of the expected reward is. Therefore, even if a model finds the best global maximum of the expected reward, it is desirable to keep exploring even if the average reward goes down, as there still might be higher maximum rewards. In the experiment section, we present evidence that this is precisely what is happening for the pLogP task when we are using intrinsic motivation.

5. Intrinsic rewards for molecular design

Inspired by these previous works, we propose three variants for intrinsic rewards: a count-based, a prediction-based, and a memory-based version.

5.1. Prediction based

For the prediction-based curiosity, we use a network that predicts the property of the next molecule. We then use the prediction error [28] as the intrinsic reward (see figure 3). To test different variations of this idea, we formulate it as:

$$r_{\text{intrinsic}}(t) = \text{dist}\left(\hat{p}\left(\text{mol}\left(t,\theta\right),\eta\right), p\left(\text{mol}\left(t,\theta\right)\right)\right). \tag{7}$$

Here $\text{mol}(t,\theta)$ is the molecule the agent parameterized by θ generates at time step t. $\hat{p}(\cdot,\eta)$ is the prediction network parameterized by η, that tries to predict the value of the considered property p of the molecule $\text{mol}(t,\theta)$. Finally $\text{dist}(\cdot,\cdot)$ is a distance metric, for example, L_1 or L_2. We consider two options for training the predictor network: either update the network each time the agent generates new molecules or collect the data in a buffer and train on the whole buffer.

Unlike Pathak *et al* [16], we do not predict the next state. The reason is that given the current state (the string so far) and the next action (the character to append), predicting the next state (the string so far with the new character appended) does not require to learn anything about the chemical space.

5.2. Count based

For the count-based version we follow closely the work in [25], as described in section 3.1. We encode at every timestep the current molecule via Morgan fingerprints (MFs) to get a feature vector describing the molecule. We then uses LSH to project the fingerprint down to a lower dimension to be able to collect sufficient statistics, as described in section 3.1. While the agent explores the chemical space, we then keep a list of all unique encountered hash strings, and keep a counter for each entry in the list. Whenever we

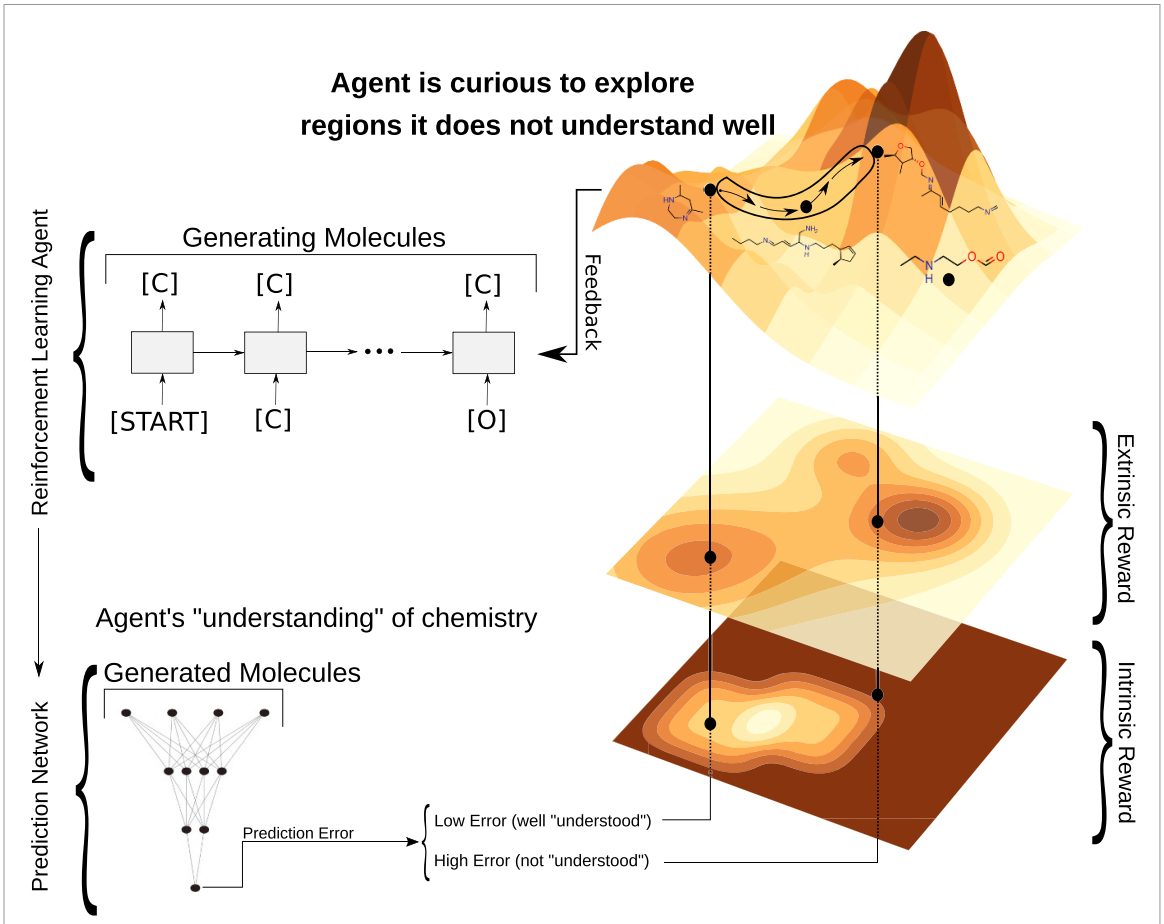

Figure 3. An illustration of curiosity: the agent generates molecules and gives them to the property prediction network. Initially, the predictions are wrong everywhere in the chemical space, but over time the network learns to predict the properties of molecules it has already encountered. Providing the prediction error as an intrinsic reward, the agent is incentivized to move to regions it has not yet explored since the prediction network should make more mistakes there. The combination of the intrinsic and extrinsic rewards is then used as feedback to train the agent. The same principle can be applied for the count-based and memory-based intrinsic rewards by substituting the prediction network with the location-sensitive hashing function and a counting table or memory and a distance function.

encounter a molecule that has a hash string that is already in the list, we count up once. For each molecule, the intrinsic reward is then defined as the square root of the inverse of the counter of its hash sting:

$$r_{\text{intrinsic}}(t) = \frac{1}{\sqrt{\text{count}(\text{LSH}(\text{MorganFingerprint}(\text{mol}_t)))} + \epsilon}. \tag{8}$$

Thus, if a molecule or a structurally similar molecule is encountered over and over again, the counter goes up, and therefore the intrinsic reward goes down, motivating the agent to go to unexplored regions of the state space, since these molecules will still have low count values.

5.3. Memory based

Finally, we consider a memory-based alternative, strongly based on the memory-based strategy introduced in section 3.3. Conceptionally, we want to penalize the agent for generating structurally similar molecules to the ones he has already generated in the past. At every time step, we explicitly store the agent's molecules into a memory of size N. If the memory overflows, we follow a first-in-first-out policy that deletes the oldest molecules first. We also filter molecules out that are already in the memory. We then calculate the pairwise Tanimoto similarity (TS) of the MFs [29] of the new molecules to every molecule in the memory and search for the maximum similarity. In the framework introduced in section 3.3 this corresponds to a neighborhood of size 1. Instead of using a positive intrinsic reward, though, we use a negative one, penalizing already visited areas of the state space instead of rewarding unvisited ones. We do this simply because it was more convenient in the implementation. However, the effect is functionally the same: it motivates the agent to explore new regions. Formally, the equation for our memory-based intrinsic reward can be written as

$$r_{\text{intrinsic, alternative}}(t) = -\max_{i=1,\ldots,N} \text{TS}(\text{MF}(\text{mol}(t)), \text{MF}(\text{mol}_i)). \tag{9}$$

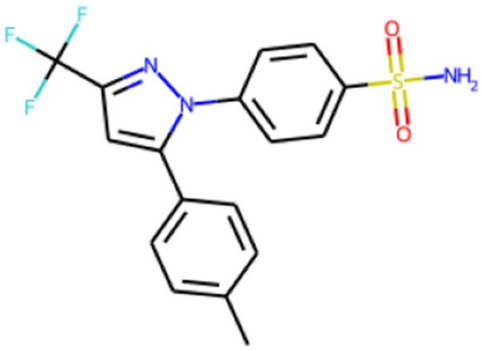

Figure 4. The molecule celecoxib which the agent needs to rediscover.

6. Experiments

We test our methods on three different tasks: optimizing for penalized logP [4] (pLogP), quantitative estimate of druglikeness [30] (QED), and similarity (in terms of TS of MFs) to the molecule celecoxib (see figure 4). The latter two tasks are from the Guacamol benchmark [31]. In the experiments, we will start training agents for pLogP to build up an intuition for the hyperparameter ranges that makes sense and analyzes the algorithms' behaviors. We choose pLogP to do so, as we have the best understanding of the reward (see section 4), and know the optimal solutions. Particularly, we know that pLogP has a deceiving reward structure and therefore expect the intrinsic rewards to help. After that, we train agents for the remaining tasks with the insights won from the pLogP experiments. For all experiments, we will use the PPO algorithm to train our RL agents. As described in section 2, we define a state s_t as the so far constructed SELFIES string and encode each of the symbols as a one-hot encoded array. To encode this sequence, for all following experiments, we use an architecture consisting of an LSTM [32] to encode the state and a linear (feed-forward) network to predict the actions based on the LSTM encoded state representation. The only hyperparameter that is linked to exploration is the entropy penalty. Therefore, for every task we search over it starting from the default 0.01 [20]. The remaining hyperparameters are not associated with exploration, and we therefore choose them to be fixed for all agents. We list them in the appendix in table A1. Except for the first experiments where we are making sure we have an optimized entropy penalty, the remaining experiments run for 500 epochs with a batch size of 64, which gives a total of 32 000 property evaluations.

7. Building intuition—pLogP

As we have the best understanding of the pLogP task, we use it to investigate the different methods more closely, build up an intuition about the methods' workings, and find sensible hyperparameter ranges. In section 4 we have shown that pLogP has a deceiving reward structure, with the carbon chain being the deceiving local minimum and the sulfur chain being a better solution. We, therefore, expect an RL agent without an intrinsic reward to find the local minimum and then get stuck in it. We test this by training an RL agent using different hyperparameters for the weight on the entropy penalty to establish a strong baseline. The weights we tried were 0.01, 0.02, and 0.03, and each agent trains three times. In table 1 we see the results for the best molecule the agent found averaged over the three runs. As expected, for each of the hyperparameter, the best molecule is the carbon chain (see figure 5(a)) with a pLogP value of 10.5. In figure 6 we see the average reward over time for the different agents. Clearly, the agents find the carbon chain very fast. They then get stuck in the local optimum, though, generating almost exclusively carbon chains without further exploration. This is also evident from the small variance of the average reward in figure 6.

7.1. Counting based

Next, we train an agent with the counting-based intrinsic reward. We start with a weight $\alpha = 1$, an MF with 256 fingerprint bits, 32 bits for the LSH, and an entropy weight of 0.02. Without any further hyperparameter experimentation, the first attempt found the sulfur chain and even better molecules (sulfur chains with some phosphor and carbon impurities, see figure 5(b)). We then tried different weights α and found that the results are quite robust. We could go down to a weight of 0.5 and still found molecules better than the carbon chain. For even lower α, the agent went back to only finding the carbon chains. Table 2 shows the best result

Table 1. Table of the pLogP values of the best-generated molecules for four different values of the entropy weight. The results are an average of three runs. All agents have found the carbon chain. The agent with a weight of 0.01 had one bad training run, where it did not find carbon chain, leading to a lower average pLogP value. Bold values indicate the best results throughout the tables.

Entropy weight	Best pLogP
0.01	7.13
0.02	**10.52**
0.03	**10.52**

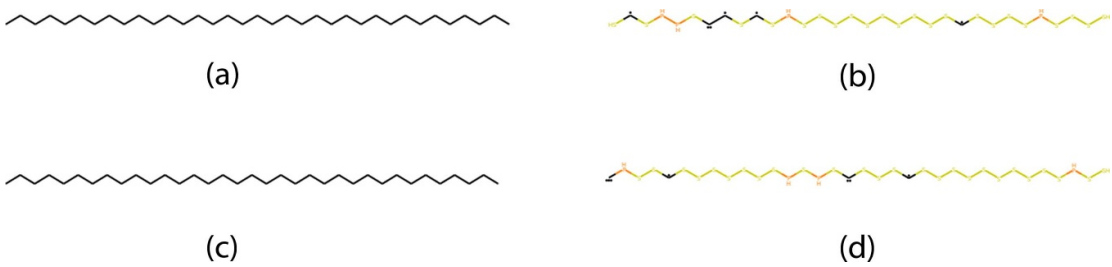

Figure 5. (a) The molecule with the highest pLogP value generated by an agent without any intrinsic rewards. (b) The molecule with the highest pLogP value generated by an agent with the counting based intrinsic rewards. (c) The molecule with the highest pLogP value generated by an agent with the prediction based intrinsic rewards. (d) The molecule with the highest pLogP value generated by an agent with the memory based intrinsic rewards.

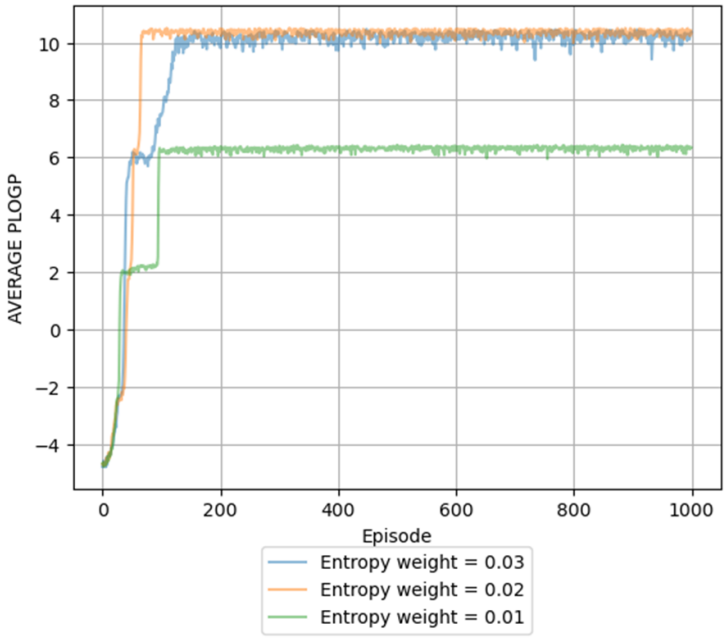

Figure 6. The average pLogP value of the generated molecules in one batch for different entropy weights. It is clear, that the agent converged and would not find better molecules later on.

Table 2. Table of the best-achieved values for the different intrinsic rewards, averaged over three runs, for the best weight α respectively.

Intrinsic motivation	Best pLogP
None	10.52
Count based	13.00
Memory based	**14.47**
Prediction based	10.52

found by an agent with a counting-based intrinsic reward. To further analyze the training, we plot the average intrinsic reward, average pLogP value and the best pLogP value for the agent with $\alpha = 1$ and for an agent with $\alpha = 0$ and entropy weight 0.02 in figures 7(a)–(c). We can see several interesting points. First, the

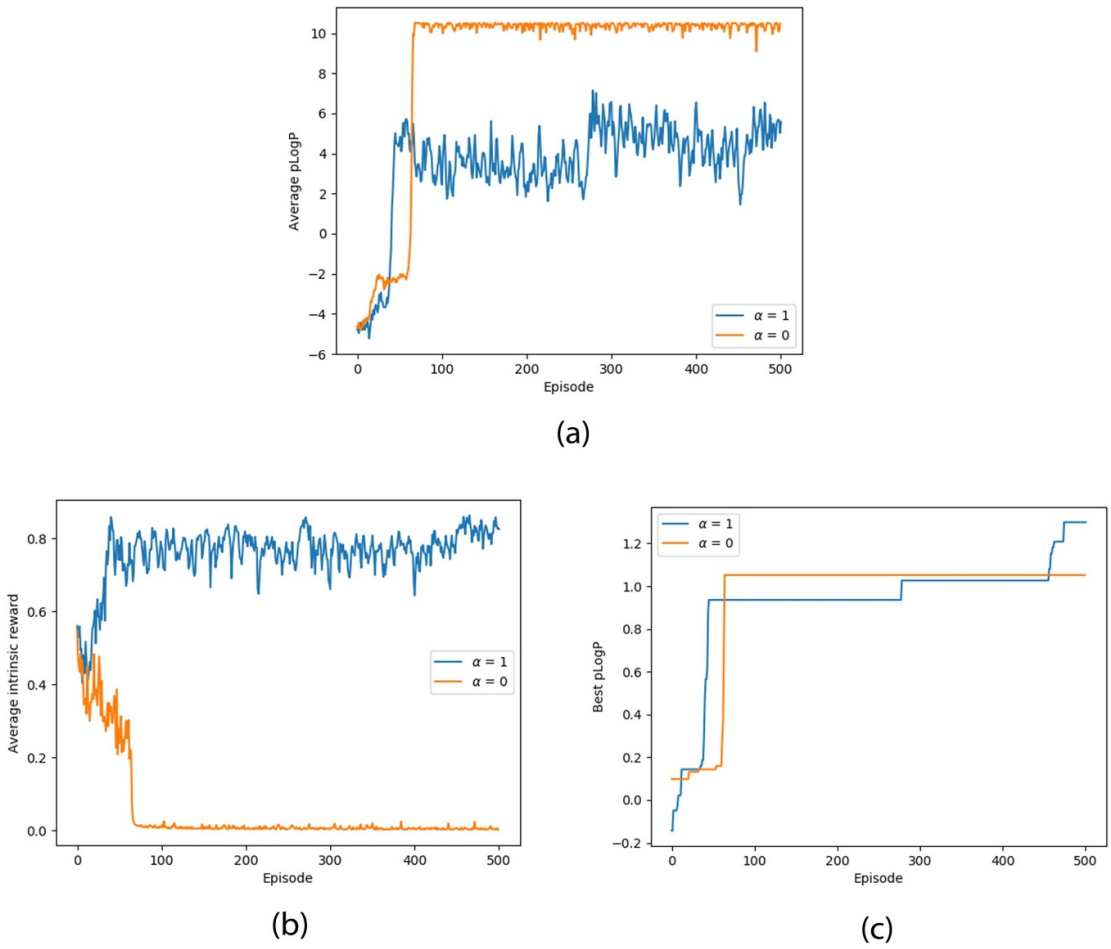

Figure 7. (a) The average pLogP value of the molecules generated per batch over the training episodes, for an agent with counting based intrinsic motivation ($\alpha = 1$) and without intrinsic motivation ($\alpha = 0$). Interestingly, the intrinsically motivated agent's average reward is lower than the vanilla agent's, but the top performing molecules pLogP value is higher. This is consistent with our observations from section 4. (b) The average intrinsic count-based reward of the molecules generated per batch over the training episodes for an agent with $\alpha = 1$ and $\alpha = 0$. (Note that we can calculate and plot the intrinsic reward even if we did not use it for the agent with $\alpha = 0$.) We can see that not explicitly encouraging the agent to keep exploring after it has found a good local optimum leads the agent to continuously generate the same molecules, leading to a count-based reward of 0, which explains why it does never find better molecules than the carbon chain. (c) The pLogP value of the best generated molecules in the training run over episodes. The agent with intrinsic motivation ($\alpha = 1$) initially finds a carbon chain with some double bonds. This is why its performance is slightly below the performance of the vanilla agent. But after some time, it finds better solutions such as the sulfur chain and even better molecules than the sulfur chain.

pLogP of the best-generated molecule starts lower for the agent that uses the intrinsic reward ($\alpha = 1$) but then rises higher. When we were looking at samples from the generated molecules, we saw that the agent was generating carbon chains in the time frame from episode 100 to 300. However, the carbon chains had some double bonds and other 'impurities', which explains why the top pLogP value is lower than for the agent without the intrinsic reward. It seems like the count-based reward motivated the agent to search through different areas of the state space before it could fully exploit the area around the carbon chain. This explains why it did not find the pure carbon chain. For our runs with lower α, it started to find the pure carbon chains, validating this conjecture.

Next, we can see that the intrinsic reward for the agent with $\alpha = 0$ goes to zero (the intrinsic reward was calculated but not used for training), as we expect from an agent that keeps only generating carbon chains. On the other hand, for $\alpha = 1$ the intrinsic reward stays high, which means that the agent constantly generates unique new molecules, resulting in better exploration.

We furthermore see that the average pLogP tends to go downwards after its initial jump. It then makes another jump at around episode 300 when it finds the new best solution (a mixed chain of sulfur and carbon), then tends to go downwards again, till it again finds a better solution and jumps. The reason for these downward trends is that the intrinsic reward pushes the agent to explore new regions of the state space over time, albeit these states having a lower extrinsic reward. This is precisely what we expect an intrinsic reward to work. It also illustrates again why an agent without intrinsic rewards cannot find the sulfur chain:

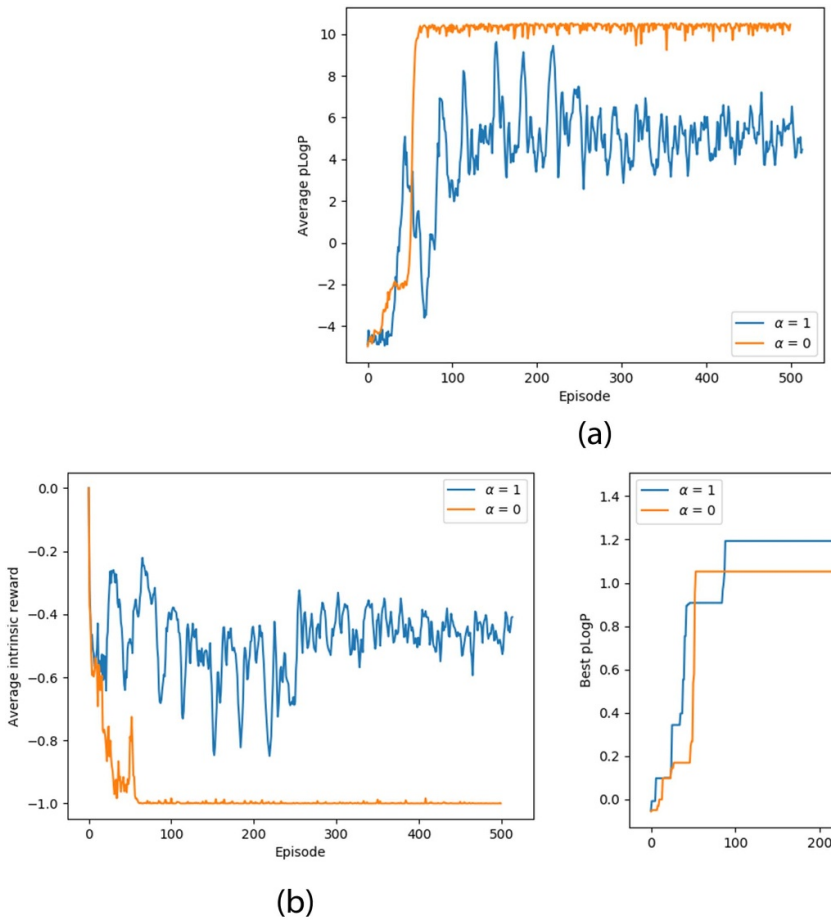

Figure 8. (a) The average pLogP value of the molecules generated per batch over the training episodes, for an agent with memory-based intrinsic motivation ($\alpha = 1$) and without intrinsic motivation ($\alpha = 0$). Like the counting-based intrinsic reward, the intrinsically motivated agent's average reward is higher than that of the vanilla agent. Still, the intrinsically motivated agent finds better top-performing molecules. This is consistent with our observations from section 4. (b) The average intrinsic memory-based reward of the molecules generated per batch over the training episodes for an agent with $\alpha = 1$ and with $\alpha = 0$. We can see that not explicitly encouraging the agent to keep exploring after it has found a good local optimum, causes the agent's intrinsic reward to go to -1. This means that the generated molecules are identical (similarity of 1) to those in the memory. This explains why the agent does never find better molecules than the carbon chain. (c) The pLogP value of the best generated molecules in the training over episodes. The agent with memory-based intrinsic motivation ($\alpha = 1$) initially finds a carbon chain with some double bonds similar to the counting-based intrinsically motivated agent. The agent then very quickly finds better solutions such as the sulfur chain and even better molecules than the sulfur chain.

the agent greedily tries to maximize its extrinsic reward and therefore does not have any incentive to visit those areas of lower extrinsic reward. Thus, the agent gets stuck in local optima.

Finally, we can see that the average pLogP for the intrinsically motivated agent is lower than the not intrinsically motivated agent, even after finding the sulfur chains. Despite this, the intrinsically motivated agent finds better molecules. This validates the point made in section 4, that normal RL optimizes for the average expected reward, not the maximally encountered reward, as we would like for molecular design. It also shows that intrinsic motivation can help with this problem as predicted, by motivating the agent to keep searching, even if the average reward goes down.

7.2. Memory based

We are now testing the same for the memory-based intrinsic reward as described in section 5.3. As before, we started with a weight of $\alpha = 1$. The memory contains the last 1000 unique molecules. The MF was calculated with 256 fingerprint bits. The best result can be found in table 2. Clearly, the memory-based intrinsic reward works well, outperforming the other approaches with a top score of 14.4. The top performing molecule is plotted in figure 5. To investigate the behavior further, we again plot the average extrinsic and intrinsic reward, and the value of the best performing molecule (see figures 8(a)–(c)). The figures look quite similar to the counting-based reward: for one, the average pLogP is lower for the curious agent than the non-curious

Table 3. The best achieved results on the QED benchmark for an agent without intrinsic rewards for different entropy penalties.

Entropy weight	Best QED
0.01	0.878
0.02	**0.882**
0.03	0.881
0.04	0.878

Table 4. Table of the best QED values averaged over three runs for the different intrinsic rewards and hyperparameter. The entropy penalty is fixed at 0.02.

Intrinsic motivation	α	Best QED
None	0	0.882
Count based	0.75	0.900
	1	0.895
	1.25	0.865
Memory based	0.75	**0.916**
	1	0.902
	1.25	0.897
Prediction based	1	0.890
	1.25	0.901
	1.5	0.900

one, but the best-encountered value is higher. However, the best pLogP value jumps up a lot quicker for the memory-based agent compared to the count-based agent. This is because as soon as the agent encounters a new molecule, this molecule is put into the memory, and generating it again immediately yields a high negative intrinsic reward. In comparison, the count-based intrinsic reward would only decrease slightly when the same molecule is encountered the next time, as the counter only counts up once. This can be an advantage but also a disadvantage. It allows the count-based agent to exploit promising regions of the state space for longer before getting pushed to new regions. On the other side, the process is slower. There we expect that it depends on the specific task, whether counting-based or memory-based rewards perform better. Another potential issue with the memory-based reward is, that it seems to train less stable. We can see that the average pLogP oscillates a lot, especially in the beginning. The reason for this is likely due to the restricted size of the memory.

7.3. Prediction based

Finally, we test the prediction based intrinsic reward. As before, we start with an intrinsic reward weight of $\alpha = 1$, the L_1 distance metric, and no greedy curiosity. We update the prediction network after every episode with the newly generated molecules. Unfortunately, the agent does not find anything better than the carbon chain. We tried many different combinations of α, distance metrics and update schemes, but nothing worked consistently. We note that we did find the sulfur chain during some rare runs with the formulation that does not use a buffer, without greedy curiosity and with the L_1 distance metric. However, this was rare and not reliably reproducible, which is why we do not report it as a main result. However, due to this, we will use the non-buffer formulation for the other benchmarks.

8. QED

Next, we are testing the three intrinsic rewards on the QED benchmark. We start again by training agents for different entropy penalties without an intrinsic reward. The results, averaged over three runs, are shown in table 3. We can see that all agents perform very similarly, with a slight advantage for the agent with an entropy penalty of 0.02. Therefore, for all following experiments regarding QED, we are using the entropy penalty of 0.02.

We test the different intrinsic rewards by training three agents with different weights α for every intrinsic reward formulation. Learning from our experiments with pLogP, we choose $\alpha = \{0.75, 1, 1.25\}$ for the counting and memory-based intrinsic rewards, and $\alpha = \{1, 1.25, 1.5\}$ for the prediction based intrinsic rewards. The results for the different hyperparameter, averaged over three runs, are shown in table 4. We can see that the agents with the counting-based and prediction-based intrinsic rewards generate slightly better top-performing molecules with QED values of 0.9 compared to 0.88 for a regular RL agent. The

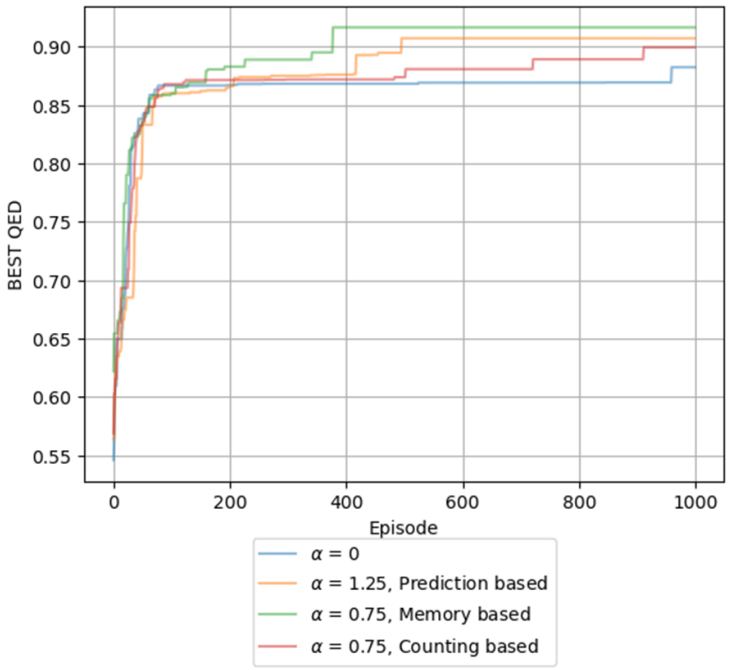

Figure 9. The best QED values encountered during training over time for agents with different intrinsic motivations and the respective best performing hyperparameter weights α, averaged over three runs. All three agents with intrinsic motivations are performing better than the one without intrinsic motivation.

memory-based agent's performance is even stronger with top QED values of 0.91, which is in accordance with the pLogP task, where the agent with memory-based rewards was also performing best. Note that this is still a bit off the theoretical optimum of 0.948 and worse than the models in Guacamol. However, in contrast to most of the models in Guacamol which started from a dataset that already included molecules with a QED of 0.948 (even the genetic algorithm which does not train on the dataset, initialized its population from the dataset), our goal was to not use any prior information, therefore a direct comparison is not sensible. Figure 9 shows a plot of the best QED values over time for the best hyperparameter settings of each version of the intrinsic reward. In figures 10(a)–(d) we plotted the best performing molecules for each version of the intrinsic rewards. Note that neither QED nor the rediscovery score consider synthetic accessibility or stability, which is why the molecules can include radicals (the dots in the plots).

9. Similarity

Finally, we are testing the algorithms on the similarity task. As before, we start by establishing the baseline without an intrinsic reward. The results are shown in table 5. This time, 0.01 is the best performing hyperparameter by a big margin. We also notice that the best-achieved performance is much smaller than for the other two tasks—0.234 vs 0.882 for QED and 10.5 (1.05 since it was divided by 10 before given to the agent) for pLogP. Therefore, we were choosing about $5\times$ smaller weights α for this task. Specifically, we use $\alpha = \{0.1, 0.2, 0.3\}$ for counting and memory-based rewards, and $\alpha = \{0.2, 0.3, 0.4\}$ for the prediction-based rewards. Table 6 shows the results of these experiments. As we can see, the agent without intrinsic rewards is the best this time. All three intrinsic rewards impede performance. This can be the case if the optimization's bottleneck is not a deceiving or sparse reward structure of the problem, but something else. In this specific example, it seems like the feedback is just too uninformative, leading to slow convergence, so that the agent did not fall in any local optimum yet. This also explains why all the agents are still far away from the theoretical optimum of 1. Figure 11 shows a comparison of the best encountered similarity values over time for the different intrinsic rewards with the corresponding best weights α. In figures 12(a)–(d) we also plotted the best performing molecules for the different techniques.

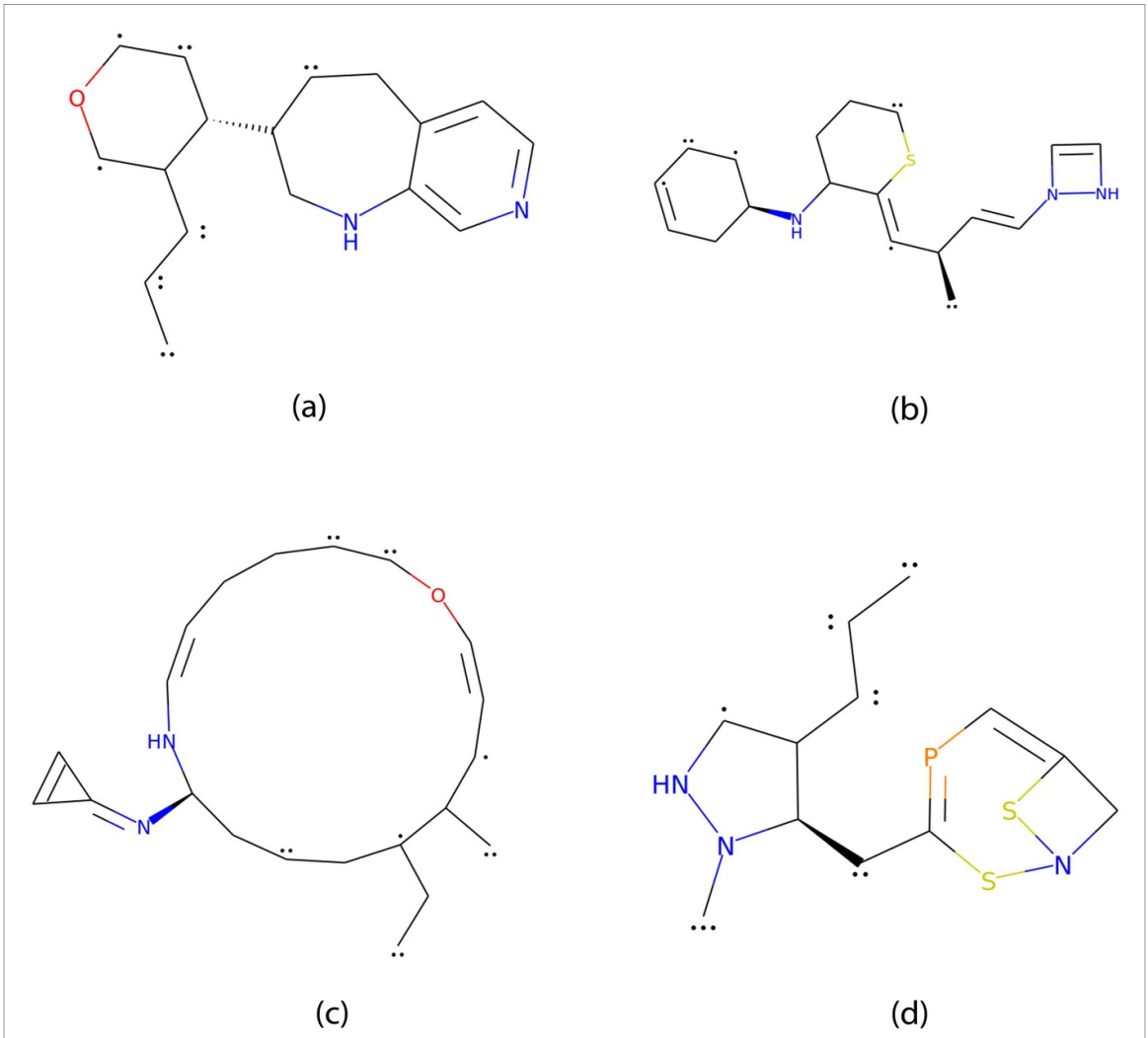

Figure 10. (a) The molecule with the highest QED value generated by an agent without any intrinsic rewards. (b) The molecule with the highest QED value generated by an agent with the counting based intrinsic rewards. (c) The molecule with the highest QED value generated by an agent with the prediction based intrinsic rewards. (d) The molecule with the highest QED value generated by an agent with the memory based intrinsic rewards.

Table 5. The best achieved results on the similarity benchmark for an agent without intrinsic rewards for different entropy penalties.

Entropy weight	Best similarity
0.01	**0.234**
0.02	0.174
0.03	0.1778
0.04	0.119

Table 6. Table of the best similarity values averaged over three runs for the different intrinsic rewards and hyperparameter. The entropy penalty is fixed at 0.01.

Intrinsic motivation	α	Best similarity
None	0	**0.234**
Count based	0.1	0.201
	0.2	0.111
	0.3	0.103
Memory based	0.1	0.198
	0.2	0.179
	0.3	1.822
Prediction based	0.2	0.217
	0.3	0.213
	0.4	0.219

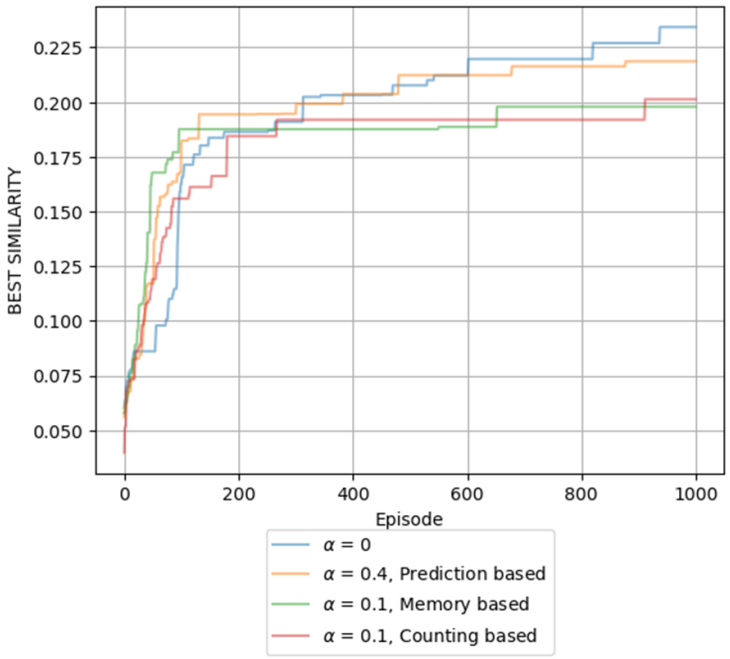

Figure 11. The best similarity values encountered during training over time for agents with different intrinsic motivations and the respective best performing hyperparameter weights α, averaged over three runs. All three agents with intrinsic motivations are performing better than the one without intrinsic motivation.

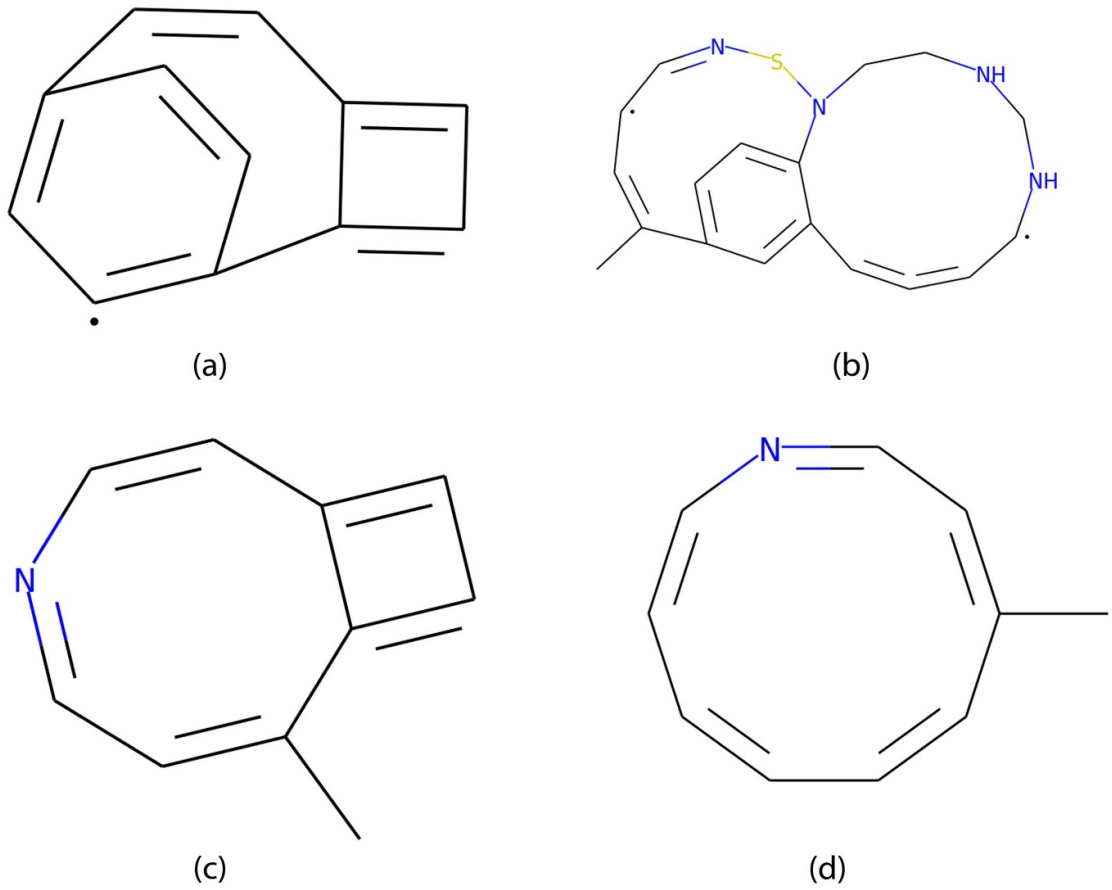

Figure 12. (a) The molecule with the highest similarity value generated by an agent without any intrinsic rewards. (b) The molecule with the highest similarity value generated by an agent with the counting based intrinsic rewards. (c) The molecule with the highest similarity value generated by an agent with the prediction based intrinsic rewards. (d) The molecule with the highest similarity value generated by an agent with the memory based intrinsic rewards.

10. Conclusion

In this work, we showed in a case study for the first time that molecular design can be a 'hard exploration' problem. We then developed *curious agents* based on approaches to intrinsic motivation from other domains. We demonstrated that they outperform their lesser curious competitors in two of three distinct molecular design tasks. In one task, pLogP, we discovered molecules that are better than the previously known best compounds. Our results point towards an efficient RL-based exploration strategy for identifying new high-performance molecules and compounds. We believe that this is an important contribution along the way to make computer systems that can discover completely new unknown molecules, no human has ever thought about. This has enormous potential, for example, in computer-aided drug design and material discovery. In the future, more experiments for larger molecules and more complex objectives will lead to better understanding of the practical applicability of curiosity-based rewards. We are optimistic that this is the case, though, since intuitively, the more complex a target is, the more likely it is a *hard exploration problem*. Importantly, while this work only answered the question whether intrinsic rewards generally can help an agent optimize its rewards better, for practical viability synthetic accessibility in particular needs to be considered, for example by including an synthetic accessibility score [33] in the reward or by designing an action space that only allows known chemical reactions [34]. Other interesting directions is the adaptation and extension of intrinsic rewards—for example by exploiting different similarity measures for memory-based intrinsic rewards. For example, we could use the edit-distances on SELFIES strings or some distance metric comparing the three-dimensional structure of molecules. An interesting question is also how intrinsic rewards affect the performance of different generative models, such as genetic algorithms, GANs or variational autoencoders. For example, curiosity used in conjunction with GANs could help to combat mode collapse, the phenomenon that the generator focuses on only one mode, however it could also hurt the training by pushing the agent too far away from the data distribution. Similarly, curiosity could help search the latent space of VAEs.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/aspuru-guzik-group/curiosity.

Acknowledgments

The authors thank Theophile Gaudin and Si Yue Guo for interesting discussions. L T acknowledges support from Mitacs via Project FR44234. M K acknowledges support from the Austrian Science Fund (FWF) through the Erwin Schrödinger Fellowship No. J4309. G P G gratefully acknowledges the Natural Sciences and Engineering Research Council of Canada (NSERC) for the Banting Postdoctoral Fellowship. A A-G thanks Anders G Frøseth for his generous support. A A-G acknowledges the generous support of Natural Resources Canada and the Canada 150 Research Chairs program. Computations were performed on the Béluga supercomputer situated at the École de technologie supérieure in Montreal. In addition, we acknowledge support provided by Compute Ontario and Compute Canada.

Appendix. Hyperparameter

For all experiments we used the PPO algorithm with an LSTM network followed by a single feed forward layer. The hyperparameter are in table A1.

Table A1. The hyperparameter that were fixed over all runs and in our experiments.

Hyperparameter	Value
γ	1
Learning rate	10^{-3}
PPO clipping value	0.2
PPO epochs	4
Batch size	64
SELFIES length	35
LSTM neurons	64

ORCID iDs

Luca A Thiede https://orcid.org/0000-0003-1202-6809
Mario Krenn https://orcid.org/0000-0003-1620-9207

References

[1] Sanchez-Lengeling B and Aspuru-Guzik A 2018 Inverse molecular design using machine learning: generative models for matter engineering *Science* **361** 360–5
[2] Gromski P S, Henson A B, Granda J M and Cronin L 2019 How to explore chemical space using algorithms and automation *Nat. Rev. Chem.* **3** 119–28
[3] Polishchuk P G, Madzhidov T I and Varnek A 2013 Estimation of the size of drug-like chemical space based on GDB-17 data *J. Comput.-Aided Mol. Des.* **27** 675–9
[4] Gómez-Bombarelli R, Wei J N, Duvenaud D, Hernández-Lobato J M, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel T D, Adams R P and Aspuru-Guzik A 2018 Automatic chemical design using a data-driven continuous representation of molecules *ACS Cent. Sci.* **4** 268–76
[5] Jin W, Barzilay R and Jaakkola T 2018 Junction tree variational autoencoder for molecular graph generation (arXiv:1802.04364)
[6] Guimaraes G L, Sanchez-Lengeling B, Outeiral C, Farias P L C and Aspuru-Guzik A 2017 Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models (arXiv:1705.10843)
[7] Nigam A, Pollice R and Aspuru-Guzik A 2021 JANUS: parallel tempered genetic algorithm guided by deep neural networks for inverse molecular design (arXiv:2106.04011)
[8] Nigam A, Friederich P, Krenn M and Aspuru-Guzik A 2019 Augmenting genetic algorithms with deep neural networks for exploring the chemical space (arXiv:1909.11655)
[9] Jensen J H 2019 A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space *Chem. Sci.* **10** 3567–72
[10] Henault E S, Rasmussen M H and Jensen J H 2020 Chemical space exploration: how genetic algorithms find the needle in the haystack *PeerJ Phys. Chem.* **2** e11
[11] Gaudin T 2019 Exploring the chemical space without bias: data-free molecule generation with DQN and SELFIES *Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019)* (*Vancouver, Canada, 2019*)
[12] Bjerrum E J and Threlfall R 2017 Molecular generation with recurrent neural networks (RNNs) (arXiv:1705.04612)
[13] Segler M H S, Kogej T, Tyrchan C and Waller M P 2018 Generating focused molecule libraries for drug discovery with recurrent neural networks *ACS Cent. Sci.* **4** 120–31
[14] Ertl P, Lewis R, Martin E and Polyakov V 2017 In silico generation of novel, drug-like chemical matter using the LSTM neural network (arXiv:1712.07449)
[15] Olivecrona M, Blaschke T, Engkvist O and Chen H 2017 Molecular de-novo design through deep reinforcement learning *J. Cheminform.* **9** 48
[16] Pathak D, Agrawal P, Efros A A and Darrell T 2017 Curiosity-driven exploration by self-supervised prediction *2017 IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)* pp 488–9
[17] Aubret A, Matignon L and Hassas S 2019 A survey on intrinsic motivation in reinforcement learning (arXiv:1908.06976)
[18] Schmidhuber J 2010 Formal theory of creativity, fun and intrinsic motivation (1990–2010) *IEEE Trans. Auton. Ment. Dev.* **2** 230–47
[19] Burda Y, Edwards H, Pathak D, Storkey A, Darrell T and Efros A A 2018 Large-scale study of curiosity-driven learning (arXiv:1808.04355)
[20] Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O 2017 Proximal policy optimization algorithms (arXiv:1707.06347)
[21] Krenn M, Hase F, Nigam A, Friederich P and Aspuru-Guzik A 2020 Self-referencing embedded strings (SELFIES): a 100% robust molecular string representation *Mach. Learn.: Sci. Technol.* **1** 045024
[22] Weng L 2020 Exploration strategies in deep reinforcement learning (available at: https://lilianweng.github.io/lil-log/2020/06/07/exploration-strategies-in-deep-reinforcement-learning.html) (Accessed 30 December 2020)
[23] Bellemare M G, Srinivasan S, Ostrovski G, Schaul T, Saxton D and Munos R 2016 Unifying count-based exploration and intrinsic motivation (arXiv:1606.01868)
[24] Strehl A L and Littman M L 2008 An analysis of model-based interval estimation for Markov decision processes *J. Comput. Syst. Sci.* **74** 1309–31
[25] Tang H, Houthooft R, Foote D, Stooke A, Chen Xi, Duan Y, Schulman J, de Turck F and Abbeel P 2017 #Exploration: a study of count-based exploration for deep reinforcement learning (arXiv:1611.04717)
[26] Stadie B C, Levine S and Abbeel P 2015 Incentivizing exploration in reinforcement learning with deep predictive models (arXiv:1507.00814)
[27] Badia A P *et al* 2020 Never give up: learning directed exploration strategies (arXiv:2002.06038)
[28] Nigam A, Pollice R, Hurley M F D, Hickman R J, Aldeghi M, Yoshikawa N, Chithrananda S, Voelz V A and Aspuru-Guzik A 2021 Assigning confidence to molecular property prediction *Expert Opin. Drug Discovery* **16** 1009–23

[29] Bajusz D, Rácz A and Héberger K 2015 Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *J. Cheminform.* **7** 20
[30] Richard Bickerton G, Paolini G V, Besnard J, Muresan S and Hopkins A L 2012 Quantifying the chemical beauty of drugs *Nat. Chem.* **4** 90–98
[31] Brown N, Fiscato M, Segler M H S and Vaucher A C 2019 GuacaMol: benchmarking models for de novo molecular design *J. Chem. Inf. Model.* **59** 1096–108
[32] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80
[33] Ertl P and Schuffenhauer A 2009 Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions *J. Cheminform.* **1** 8
[34] Gao W, Mercado R and Coley C W 2021 Amortized tree generation for bottom-up synthesis planning and synthesizable molecular design (arXiv:2110.06389)