AI-assisted curriculum learning

Haoping Xiao

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo July 25, 2022

Supervisor

Prof. Samuel Kaski

Advisors

M.Sc. Iiris Sundin

M.Sc. Sebastiaan De Peuter

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Haoping Xiao

Title AI-assisted curriculum learning

Degree programme Computer, Communication and Information Sciences

Major Machine Learning, Data Science and Artificial Intelligence **Code of major** SCI3044

Supervisor Prof. Samuel Kaski

Advisors M.Sc. Iiris Sundin, M.Sc. Sebastiaan De Peuter

Date July 25, 2022 **Number of pages** 40 **Language** English

Abstract

Deep reinforcement learning is widely applied in de novo molecular design to generate molecules with desired properties. This technique often has a sparse reward problem since the target properties usually exist for the minority of the generated molecules. With a sparse reward, the agent in a de novo design tool may fail to begin learning and waste much time exploring areas in the vast chemical space that are far away from the target area. A recent study successfully applied curriculum learning to mitigate the sparse reward problem. However, a chemist must hand-craft a curriculum for the generative agents, which requires domain knowledge and is time-consuming, especially as tasks grow in complexity. This thesis applies an AI assistance framework to assist in a curriculum design task by recommending actions. The AI assistant infers the private information of the chemist, including design objective function and chemist's biases. Then, the AI tries to convince the chemist to adopt its advice. The chemist is free to choose action after receiving advice. This setting presents a significant improvement in AI safety. We demonstrate this method with a simulated chemist in a de novo design task, where the generated molecules should be predicted to be active against the dopamine type 2 receptor (DRD2). Our experiments show that the AI-assisted curriculum learning achieves a pronounced improvement on the sparse property (DRD2) and significantly outperforms unassisted curriculum learning.

Keywords curriculum learning , AI-assisted design , de novo molecular design , sparse reward

Preface

I want to thank my supervisor Prof. Samuel Kaski, my advisors M.Sc. Iiris Sundin and M.Sc. Sebastiaan De Peuter for their valuable comments and helpful guidance. Without their help, my thesis and experiments would not have progressed as smoothly. When I was confused about algorithms I encountered, they helped me understand the algorithms better. In addition, I would like to thank AstraZeneca for supporting my thesis and providing me with insights into de novo molecular design. Lastly, I want to thank my families, my love and my friends for their company and encouragement throughout my thesis project.

Wish you all the best!

Otaniemi, July 25, 2022

Haoping Xiao

Contents

Symbols and abbreviations

Symbols

$\mathcal{M}$	a Markov Decision Process
$\mathcal{S}$	state space in $\mathcal{M}$
$\mathcal{A}$	action space in $\mathcal{M}$
GO	a start token in Recurrent Neural Network
EOS	an ending token in Recurrent Neural Network
ω	parameters in reward function
θ	bias parameters in user model
m	molecules
STOP	an action that stops curriculum design

Operators

exp	Exponential function
$\arg\max$	Return the indices of maximum value
$\mathcal{T}$	transition function in $\mathcal{M}$
$\mathcal{R}$	reward function in $\mathcal{M}$
π	a decision-making policy
Q	Q value function
$p(m)$	property value p of molecule m
$\sum$	sum

Abbreviations

RL	reinforcement learning
CL	Curriculum learning
SMILES	Simplified Molecular Input Line Entry Specification
RNN	Recurrent Neural Networks
MDP	Markov Decision Process
MCTS	Monte Carlo Tree Search
GHPMCP	Generalized Hidden Parameter Monte Carlo Planning
QED	Quantitative Estimate of Drug-likeness
SA	Synthetic Accessibility

1 Introduction

1.1 Motivation

Drug discovery involves proposing novel compounds with desirable pharmacological properties. Traditional drug discovery is time-consuming and cost-intensive, including an average 10-15 years for launching to market with billions to invest[27]. The prohibitive cost prompts the usage of computer-aided drug design, which accelerates drug discovery in the vast chemical space. The prominent role of computer-aided drug design is to prioritize libraries of molecules regarding selected properties toward a target of interest, thereby narrowing down drug candidates to a few clusters. However, a synthetically feasible molecular space is on the order of $10^{60} - 10^{100}$[24] and therefore a complete search in the vast space is computationally infeasible.

An alternative method for drug discovery is de novo molecular design, which is a promising technique to propose novel compounds that satisfy a desirable property profile without needing a starting template and enumerating large virtual libraries. With the advancement of deep learning, deep generative models have been introduced for de novo design to generate molecules with desirable properties by using different algorithms, such as policy-based reinforcement learning(RL), value-based RL, molecular latent vectors, tree search and genetic algorithms[15].

The policy-based reinforcement learning framework encompasses an agent optimizing its action policy to produce molecules that maximize a user-defined reward function. In other words, the framework utilizes correlations between policies and environmental rewards to reinforce and improve agent performance. Popular agent models in de novo drug design generally are deep generative models such as Recurrent Neural Network(RNN), Generative Adversarial Neural Networks(GAN) and variational autoencoders. These models are pre-trained on a large dataset to learn about producing molecules, and the user-defined reward function typically consists of multiple scoring components such as activity, selectivity and physicochemical properties. This results in a multi-parameter optimization(MPO) problem. Given sufficiently long training time, the agent models can learn to generate molecules with desired properties that satisfy a user-defined MPO objective.

However, when the user-defined MPO objective contains some target-specific scoring components like specific bio-activity and molecular docking, only a small fraction of molecules in the vast molecule space satisfy the objective. This leads to a reward sparseness problem in reinforcement learning. In cases these target-specific properties are expected in de novo design, the resulting reward function provides sparse extrinsic reward signals since by definition there are fewer rewards, and hence fewer instances provide gradients to train the agent policy. In the most problematic de novo design tasks, the agent may waste much time exploring areas in the vast chemical space that are far away from the target area and fail to begin learning at all. Therefore, leaving the computational efficiency in computing these properties aside, optimization for such reward functions is practically difficult.

Curriculum learning(CL) has been proposed as a training strategy to overcome sparse reward problems in reinforcement learning. Previous work[15] mitigates two

de novo drug design tasks with sparse reward by curriculum learning. In curriculum learning, the agent sequentially learns curriculum objectives that gradually increase in complexity. Eventually, a curriculum with gradually increasing complexity guides the agent to learn the sparse reward function. To construct curriculum for solving sparse problems, chemists need to create a set of potential curriculum objectives that correlated with the sparse reward function based on domain knowledge. From the set of curriculum objectives, a chemist sequentially chooses a curriculum objective as intermediate reward function which improves convergence on the sparse reward function. The chemist also needs to design a threshold of the curriculum objective value that the agent must achieve as a progression criterion to check sufficient learning of each curriculum objective. Once the sparse reward function converges or there are no more promising curriculum objectives, the curriculum construction is complete. Figure 1 shows a curriculum learning application where the chemist would like the de novo drug design tool to produce complex molecules that possess a complex scaffold, which results in a sparse reward objective. To use curriculum learning for sparse reward function, the chemist needs to provide multiple curriculum objectives after rigorous consideration. From all possible curriculum objectives, chemists select a few successive curriculum objectives that gradually increase in complexity. If and only if the final curriculum progression criterion is satisfied, the agent could progress into the production phase with a sparse reward function. In this thesis, we describe the selection of curriculum objectives as curriculum design.

Human experts must hand-craft a curriculum for the agents, which requires domain knowledge and is time-consuming, especially as tasks grow in complexity. In addition, humans are required to design a curriculum progression criterion, which is typically a threshold of the curriculum objective value. This criterion is hard to determine as it cannot promise success in the next curriculum objective. These problems motivate this thesis to create an AI assistant to help humans design a curriculum by providing advice.

1.2 Problem Statement

A sparse reward is difficult to incentivize the agent in a de novo design tool to produce molecules with desired properties. A practical method to mitigate sparse reward problem is curriculum learning. As in previous work[15], we divide the curriculum learning process into curriculum phase and production phase. The agent learns a sequence of curriculum objectives in the curriculum phase and then learns the sparse reward in the production phase. While design a curriculum is time-consuming and laborious, this thesis applies an AI assistant to help curriculum design.

Curriculum design can be modeled as an infinite-horizon Markov Decision Process(MDP) $E =< \mathcal{S}, \mathcal{A}, T, \mathcal{R}_\omega, \gamma, p_{0,s} >$. The states S are represented by the parameters of the agent model in a de novo drug design tool. A set of curriculum objective selections and a STOP action constitute action space $\mathcal{A}$ for the chemist. From the action space $\mathcal{A}$, the chemist sequentially takes actions to choose a curriculum objective or takes the STOP action to stop curriculum design. A curriculum objective is allowed to be revisited in curriculum learning and therefore, the agent

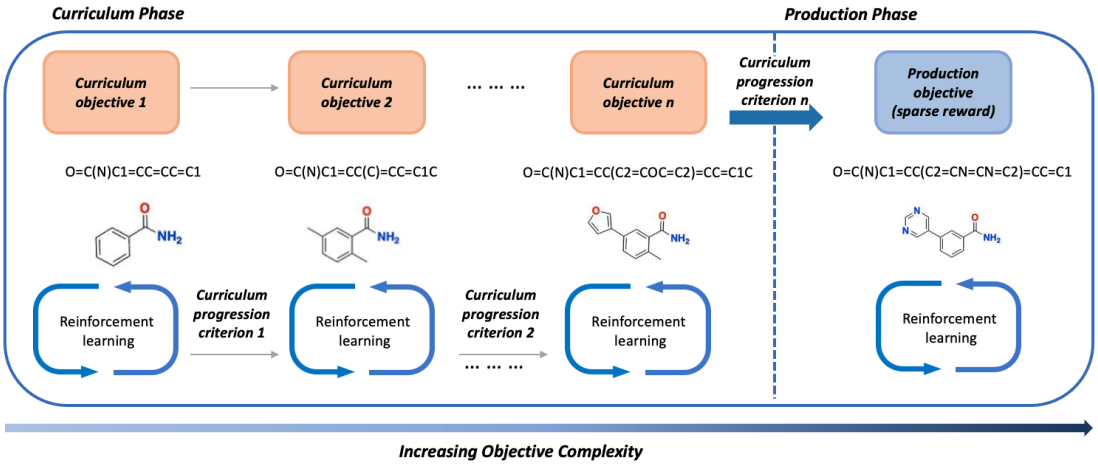

Figure 1: The figure is adapted from Guo, Jeff, et al. "Improving De Novo Molecular Design with Curriculum Learning." (2021). In the Curriculum Phase, the agent progresses through successive Curriculum Objectives that gradually increase in complexity. Curriculum progression criterion checks for sufficient learning of each Curriculum Objective based on a threshold that the agent must achieve. If and only if the final Curriculum Progression Criterion is satisfied does the agent progress to the Production Phase in which a sparse reward function is applied.

will learn the selected curriculum objective with a fixed number of epochs. This definition optimizes the process of how chemists design a curriculum. The chemist is not required to design a threshold as a curriculum progression criterion to check sufficient learning of each curriculum objective. The same curriculum objective could be selected again if the agent has not converged on that curriculum objective. The chemist action a_i (except the STOP action) at time step i will choose a curriculum objective for the agent s_i to learn, resulting in a new state s_{i+1}. For the STOP action, the agent remains in state s_i. With the definition of the action space, the chemist can only focus on selecting curriculum objectives and decides when to enter the production phase by selecting the STOP action.

At time step i the transition function $T(s_{i+1}|s_i, a_i)$ defines a distribution of potential next states s_{i+1} given that the chemist has taken action a_i in current state s_i. The chemist has a curriculum design objective function $f_\omega(s) = \frac{\sum_i^k \omega_i * \bar{p}_i(m|s)}{\sum_i^k \omega_i}$. $\bar{p}_i(m|s)$ calculates the average value of property p_i of sampled molecules m, which are from the agent in state s in the production phase. The chemist evaluates the average property value $\bar{p}_i$ and each property p_i has a weight ω_i. We define the reward function as $\mathcal{R}_\omega(s_i, a_i, s_{i+1}) = f_\omega(s_{i+1}) - f_\omega(s_i)$, which is the improvement in the curriculum design objective value from one time step to the next. The chemist could not explicitly describe $\mathcal{R}_\omega$, but we assume the AI assistant has access to its parametric function class $\mathcal{R} = \{\mathcal{R}_{\omega'}\}_{\omega' \in \Omega}$. $\gamma \in [0, 1]$ is the discounting rate. The objective in curriculum design is to maximize the expected discounted cumulative

reward $\mathbb{E}\left[\sum_{i=0}^{\infty} r_i \gamma^i \mid T, p_{0,s}\right]$ where r_i is the reward received at the time step i. When the discounting rate equals 1, the expected discounted cumulative reward is equal to $\mathbb{E}\left[f(s_\infty)\right]$, which is the expected curriculum design objective value when the curriculum design process ends. Finally, $p_{0,s}$ is the start state distribution: $s_0 \sim p_{0,s}$.

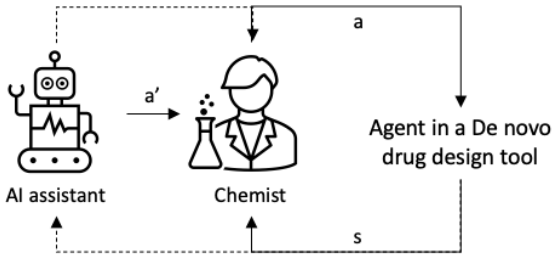

Figure 2: The assistant observes both the action a taken by the chemist and the new state s of the de novo design tool, and uses this to infer the chemist's private information. In every time step the assistant gives new advice a' , appropriate for the current state s, based on its inference of the chemist's private information. The chemist incorporates the advice into his/her own decision-making and arrives at an action a. Only the chemist directly takes action to select a curriculum objective for the agent.

This thesis applies an AI-assistance framework[11] to help a chemist design a curriculum without initially knowing the chemist's private information, including curriculum design objective function and chemist's biases. The goal of the AI assistant is to increase the quality of curriculum construction, measured by the chemist's cumulative reward. Therefore, the goal of the AI assistant is consistent with the goal of the chemist. Figure 2 shows the interaction among an AI assistant, a chemist and a de novo design tool.

1.3 Structure of the Thesis

The thesis proceeds as follows. Chapter two introduces the background knowledge of the de novo molecular design tool that we used in the thesis, sparse reward problem, Markov decision process, Monte Carlo Tree Search and Human-AI collaboration. These techniques are all related to the methods applied in this thesis.

Chapter three elaborates the AI-assisted curriculum design method. First, we abstract the AI-assistant's task as a generalized hidden parameter Markov decision process. Second, the AI assistant applies a variant of Monte Carlo Tree Search to plan advice according to the inferred reward function and chemist's biases.

Chapter four describes the experiments we conduct to demonstrate the utility of curriculum design in simulated example cases. Chapter five summarizes the thesis.

2 Background

This thesis aims to solve the sparse reward problem in de novo drug design by applying an AI-assistance framework. This chapter will introduce the background of the thesis, including a de novo design tool, sparse reward remedies, Markov decision process, Monte Carlo tree search and human-AI collaboration.

2.1 REINVENT: de novo molecular design tool

This thesis focuses on a de novo molecular design tool REINVENT[5], implemented by the Molecular AI group at AstraZeneca. REINVENT aims to efficiently explore the chemical space and produce promising compounds with targeted properties. REINVENT uses a policy-based reinforcement technique to fine-tune a generative model (RNNs), which has been pre-trained to produce valid molecules. The fine-tuning enables the generative model to produce molecules with desired properties. Users are required to define a scoring function for REINVENT, consisting of multiple scoring components such as activity, selectivity and physicochemical properties. A diversity filter, the pre-trained generative model and the user-defined scoring function will be combined to generate rewards for the agent's action. The rewards incentivize the agent model to produce valid and diverse molecules with desired properties. More details will be introduced in the following sections.

2.1.1 SMILES

Molecules in REINVENT are represented in Simplified Molecular Input Line Entry Specification (SMILES)[33] notation. Based on molecular graph theory, SMILES rigorously represents molecules by a sequence of characters corresponding to atoms and special chemical structures. In the SMILES representation system, atoms are represented by their atomic symbols, and special symbols $-$, $=$, $\#$, $:$ are used to denote single, double, triple, and aromatic bonds respectively. Branches in a compound are specified by enclosures in parentheses. The SMILES notation also consider rules to designate cyclic structures, disconnected structures and aromatic structures. In brief, SMILES establishes a natural grammar to represent molecules, and is well suited for high-speed machine processing. Canonicalization algorithms[34] can be used to generate consistent and unique encodings for molecules.

2.1.2 Generative Model

As a de novo molecular design tool, REINVENT requires a generative model to produce valid molecules with desired properties. A generative model approximates a distribution of training data, so it can sample data from the same distribution. REINVENT employs Recurrent Neural Networks (RNNs) as a generative model. RNNs are particularly well-suited to the sequential structure of SMILES strings. Recent success[28] demonstrates that RNNs could not only reproduce molecules in the training dataset, but also generate novel and valid SMILE compounds.

RNNs are a class of neural networks that incorporate the dependency of previous sequential data. They have a typical network architecture as Figure 3. For every time step t, the previous hidden state h_{t-1} and the current input x_t are encoded into a new hidden state h_t, which determine both the output y_t at the current step as well as influence the next hidden state. This special network structure enables the network to memorize previous information, which contributes to the next prediction. However, this structure also results in a vanishing and exploding gradients [3] due to the gradient-based back-propagation through time (BPTT) technique in training RNNs. Therefore, RNNs in REINVENT adopted LSTM[17]/GRU[9] cells, which are designed to address the vanishing/exploding gradient problems by learning to determine which previous information to retain.

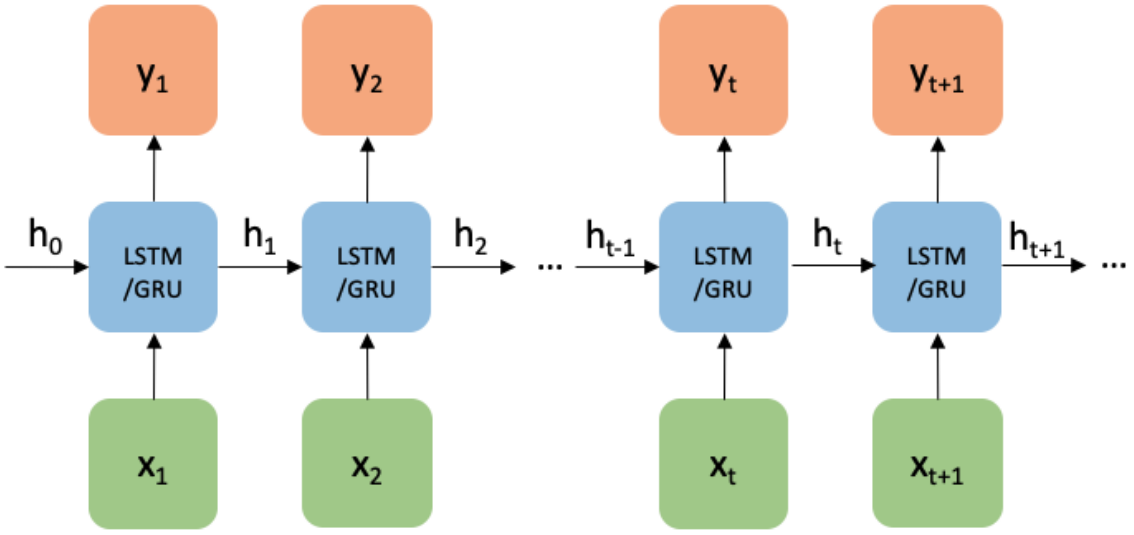

Figure 3: The architecture of RNN

The RNN model in REINVENT is trained to produce valid and novel molecules on ChEMBL[14]. The training process of RNN applies teacher forcing technique[37]. Given a SMILES sequence$(y_1, y_2, \cdots, y_n)$, the parameters of the RNN model are updated by maximizing the estimated likelihood $P(y_t|x_1, \cdots, x_t)$, where $x_1 = GO$ (a start token) and $x_t = y_{t-1}$ (for $t > 1$). Since the hidden state h_{t-1} incorporates the previous information, the likelihood $P(y_t|x_1, \cdots, x_t)$ can be written as $P(y_t|h_{t-1}, x_t)$. In tuition, the teacher forcing technique aims to correct the prediction at every time step given a preceding sequence by updating the parameters.

When RNNs generate molecules in SMILES notation, a special start token GO is fed into the network as x_1 with an initial default hidden state h^0. This will update the next hidden state and predict the next token y using the parameters that have been learned. This process iterates until the ending token EOS is predicted.

In brief, the RNN model in REINVENT can produce valid and novel molecules in SMILES notation.

2.1.3 Reinforcement learning in REINVENT

REINVENT encompasses a reinforcement learning tool to fine-tune generative models such that generated molecules possess desired molecules. A reinforcement learning system typically incorporates an agent and an environment in which the agent takes actions and receives corresponding rewards for its actions. The reinforcement learning system in REINVENT is shown as Figure 4[1].

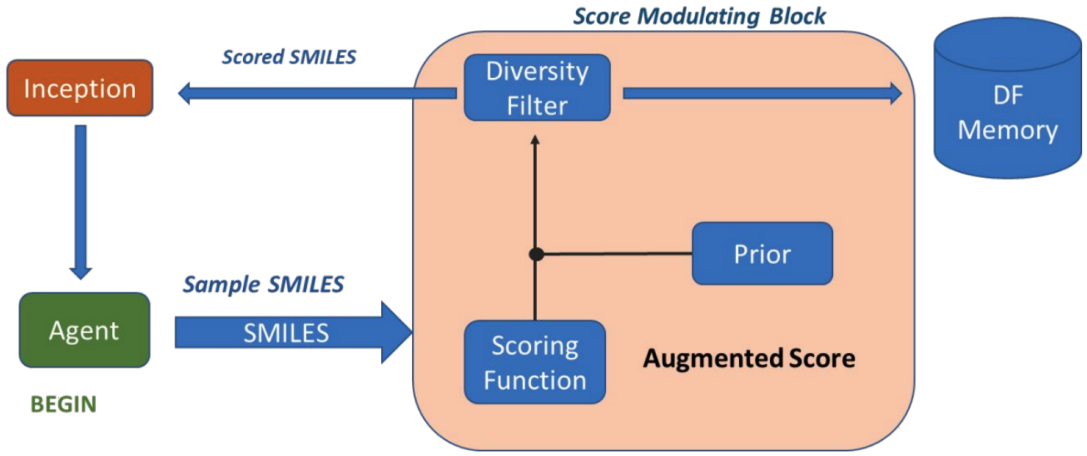

Figure 4: RL cycle in REINVENT: The agent produces molecules in SMILES notation to maximizing the reward from the score modulating block. The scoring function incentivizes the agent to produce molecules with desired properties. The prior model check the validity of generated molecules. The diversity filter encourages diverse generation and collects generated scaffolds in the DF memory. The inception module stores SMILES that previously receive high rewards for experience replay.

In REINVENT, since the agent is a pre-trained RNN, its action is to predict the next token given a sequence of generated tokens until the ending token occurs. A sequence of actions produces a sequence of tokens, which can be translated into SMILES. The environment for the agent is a score modulating block that consists of a user-defined scoring function, "prior" and diversity filter. A user-defined scoring function combines multiple scoring components. Each scoring component is responsible for one target property. The combination could be either a weighted sum (eq 1) or a weighted product (eq 2), where each component p_i has a weight w_i. The user-defined scoring function outputs a score to evaluate how good generated molecules m are.

$$f(m) = \frac{\sum_i w_i \times p_i(m)}{\sum_i w_i} \tag{1}$$

[1]The figure comes from the support information of REINVENT

$$f(m) = \left[\prod_i p_i(m)^{w_i}\right]^{1/\sum_i w_i} \tag{2}$$

The "prior" is a generative model that is the same as the agent at the beginning of the RL. The prior does not undergo any optimization while the agent is updated through interaction with the environment. Even without any optimization, the prior has great generative capacity since it is a pre-trained RNN. The prior likelihood prediction constrains the exploration of the agent so that the produced molecules have valid SMILES syntax during RL. The last element in the score modulating block in Figure 4, the diversity filter, determines if the generated compounds already exist or if there are too many compounds of the same scaffold. It may reset the user-defined score to 0 for the purpose of diverse generation, even if the generated molecules satisfy a desired property. The diversity filter encourages the agent to explore other part of the chemical space. In summary, the environment provides feedback to help the agent balance exploitation in the desired chemical space and exploration in the promising chemical space.

The last component of the RL system is inception, which essentially acts as a form of experience replay. It keeps track of previously high-score molecules and randomly exposes a subset of them to direct the learning process[5]. The reinforcement learning(RL) algorithm that is employed in REINVENT is policy-based. It explicitly learns an optimal stochastic policy[30]. While the policy in REINVENT is a parametric model, a popular way to optimize the policy is through policy gradient descent[23]. The gradient method in REINVENT starts by sampling a batch of molecules and then calculates the corresponding loss from environment feedback. The parameters of the policy are updated through back-propagation with the loss. This optimization process stops when a satisfied policy is found. Using this gradient descent method, a pre-trained generative agent model can be fine-tuned to produce molecules with desired properties by interacting with the environment.

2.2 Sparse reward remedies

In de novo drug design with reinforcement learning, the chemist's goal is translated to the scoring function, which could consist of multiple molecular properties. While some molecular properties, such as molecular weights, are possessed by every molecule, there are other properties only exist for a small fraction of molecules, such as specific bio-activity. The reward function is a sparse function when it is dominated by sparse properties, since there are fewer generated molecules can receive rewards. This sparse reward problem will lead to long training time and a suboptimal policy in reinforcement learning.

In recent years, advanced techniques have been proposed to mitigate sparse reward problems. Hand-crafted shaping rewards have shown great success in accelerating learning [21]. Recent work has shown this technique can even help AI learn to play the advanced game Dota2 at an expert level[4]. Reward shaping enhances the original reward from the environment with additional localized rewards that encourage behavior consistent with prior knowledge. The additional reward should

be justified for the intended behavior and thus appropriately reward or punish the agent's actions, filling the gap of the original sparse reward. In REINVENT, the diversity filter plays a similar role in shaping rewards. When the REINVENT agent overexploits and generates molecules with the same scaffold, the diversity filter will penalize it to encourage exploration. This technique can alleviate mode collapse, in which the agent gets stuck into a pathological local minimum in the objective function, and converges to generating a few instances with high reward[18]. However, the technique in REINVENT does not fill the gap of the original sparse reward, and the agent may fail to learn early on.

Curiosity-driven methods have been shown to be effective in sparse reward problems[8]. Curiosity is a type of intrinsic reward function which uses prediction error as reward signal. By rewarding the agent when it visits states on which it has high prediction error, we incentivize it to collect those experiences that will improve its predictions most. The agent is intrinsically motivated to explore its environment out of curiosity. The curiosity encourages the agent to explore unseen states in the environment, so that the agent may eventually help solve the sparse reward task. However, this approach may not be applicable since many challenging but irrelevant tasks could potentially distract the exploration[25]. In a de novo design problem, the chemical space is extremely huge, and a curiosity-driven method may result in endless exploration.

Transfer learning is another potential technique. Transfer learning uses a large prior dataset to bootstrap the AI model so that it possesses prior knowledge in a new task. Transfer learning may direct the agent to an area that is close to the target area, giving the agent a better chance of finding the high rewards around the target area. Therefore, transfer learning could alleviate sparse reward problems in principle. The agent in REINVENT is pre-trained using a transfer learning technique. But the purpose of pre-training is to help the agent produce valid molecules in the vast chemical space. To deal with sparse reward, a specific dataset appropriate for the de novo drug design goal is required, which is practically infeasible. Chemists resort to de novo design tools to generate promising molecules because there are few known molecules that satisfy their goal. A workaround in REINVENT is inception, which is a modified version of experience replay. Once well-scored molecules are found, they are stored in memory and inception will randomly expose a subset of them to the agent to direct the learning process. Nevertheless, the agent may struggle to find well-scored molecules in a sparse reward environment at the beginning and thus inception would not work.

Curriculum learning has been shown to successfully mitigate sparse reward problems. Human education is usually organized through a curriculum which introduces concepts from easy to difficult and from concrete to abstract. Curriculum learning is inspired by human education, helping AI models master challenging concepts by learning easier concepts first. Curriculum learning can be seen as a continuation method, which can help find better local minima for a non-convex function[2]. Continuation methods are a well-known non-convex optimization approach. They first optimize a simple(smoother) objective function to reveal the global picture of the smooth problem. Then, the objective function is gradually transformed into less smooth

version until the original non-convex objective function is recovered. Applying a continuation method in optimization involves a sequence of training criteria, starting from an easier objective function and ending with the training criterion of interest.

Curriculum learning can be implemented at the data-level, model-level and task-level[29]. Data-level curriculum learning presents samples to the AI model from easy to hard and thus needs a difficulty criterion for data. On the other hand, model-level curriculum learning does not require a difficulty criterion but instead, it gradually increases the AI model capacity. Task-level curriculum learning gradually increases the difficulty of the tasks. An important element in curriculum learning is the curriculum progression criterion, which determines when to start learning in the next level. In practice, training is stopped when the task performance has converged. Otherwise, training with a fixed number of epochs may be sufficient, since a curriculum can be revisited later.

Curriculum learning has been applied in de novo design to mitigate sparse rewards[15]. For de novo design, data-level curriculum learning is infeasible because of a lack of data in specific tasks. As the generative model is usually pre-trained on huge dataset to possess great generative capacity, updating the model architecture through model-level curriculum is not feasible. Previous studies applied task level curriculum learning to decompose complex reward function into simpler constituent objectives, guiding training towards successful convergence of the final objective. Chemists have to design a set of curriculum objectives, from which they select a subset to form a curriculum. Although the method achieve great improvement, it requires human expertise and is time-consuming. Therefore, this thesis aims to create an AI assistant to help chemists choose a sequence of curriculum objectives to construct a curriculum.

2.3 Markov decision process

This thesis defines curriculum design as a Markov Decision Process. A Markov Decision Process (MDP) is a mathematical framework that models decision-making problems of agents where the results are partly stochastic and partly controllable. MDPs contain a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $\mathcal{T}$, a reward function $\mathcal{R}$, a discounting rate $\gamma \in [0, 1]$ and an initial state distribution $p_{0,s}$ if applicable. The state characterizes the current state of the environment, with which the agent interacts. The actions allow the agent to change its environment. $\mathcal{T}(s_{i+1}|s_i, a_i)$ is the transition function that describes the distribution of the next state s_{i+1} given that the agent takes action a_i in the current state s_i. The decision-making process is a Markov process if the new state s_{i+1} only depends on the current state s_i. In other words, $\mathcal{T}(s_{i+1}|s_i, a_i, s_{i-1}, a_{i-1}, \cdots) = \mathcal{T}(s_{i+1}|s_i, a_i)$. Under Markovian dynamics, the current state s is sufficient for making an optimal decision. The reward function specifies reward for being in a state s_i $\mathcal{R}(s_i)$, for taking an action a in state s_i $\mathcal{R}(s_i, a)$, or taking an action a in state s_i and ending up in state s_{i+1} $\mathcal{R}(s_i, a, s_{i+1})$. In summary, an MDP is formalized as a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, p_{0,s} >$.

The agent in a MDP $< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, p_{0,s} >$ has a policy $\pi(a|s) = P(\mathcal{A} = a_i|\mathcal{S} = s_i)$, which is defined as the probability distribution over actions for every possible

state. The goal of the agent in a MDP is to maximize the expected discounted cumulative reward $\mathbb{E}\left[\sum_{i=0}^{\infty} r_i \gamma^i \mid T, p_{0,s}, \pi(a|s)\right]$ that it receives. For an infinite MDP, the discounted cumulative reward models the infinite cumulative rewards better by giving greater weights to immediate rewards than delayed rewards. Therefore, an optimal policy for the agent in a MDP is the policy that can maximize the expected discounted cumulative reward.

2.4 Monte Carlo Tree Search

This thesis formulates curriculum design as a Markov Decision Process and applies a method based on Monte Carlo Tree Search to find a solution. Monte Carlo Tree Search(MCTS) is a search algorithm for finding optimal decisions by taking actions in the environment and building a search tree according to the accumulated rewards.

"The basic MCTS process is conceptually very simple... A tree is built in an incremental and asymmetric manner. For each iteration of the algorithm, a tree policy is used to find the most urgent node of the current tree. The tree policy attempts to balance considerations of exploration (look in areas that have not been well sampled yet) and exploitation (look in areas which appear to be promising). A simulation is then run from the selected node and the search tree updated according to the result. This involves the addition of a child node corresponding to the action taken from the selected node, and an update of the statistics of its ancestors. Moves are made during this simulation according to some default policy." [7]

MCTS builds a tree of visited state-action pairs starting from the current state (Figure 5). MCTS uses Monte Carlo simulation to accumulate value estimates to guide toward highly rewarding trajectories in the search tree. It consists of four steps: selection, expansion, simulation and backup. These four steps are repeated multiple times and the action with the highest value will be selected as the most promising action. In summary, MCTS uses Monte Carlo simulation to accumulate value estimates to guide toward highly rewarding trajectories in the search tree.

- Selection: start from the root node R and select successive child nodes until a leaf node L is reached. A leaf node is a node that has no explored child nodes. This selection needs to balance exploration and exploitation. Typically, the upper confidence bound(UCB) for tree (UCT) policy is applied to select an action. Specifically, each node in a tree has an associated UCB value and the child node with the highest UCB value will be chosen. The UCB equation is shown in Equation 3, where s' represents a child node of a parent node s after taking action a, $Q(s,a)$ is the value of the child node, $N(s)$ is a count of visits to state s and $N(s,a)$ is a count of visits to the state-action pair (s,a). As we can see, the first term $Q(s,a)$ controls exploitation while the second term considers exploration. The more a child node s' is visited (higher $N(s,a)$), the lower the second term becomes, hence decreasing its probability of being selected again. UCB algorithm has exploration and exploitation inherently built-in.

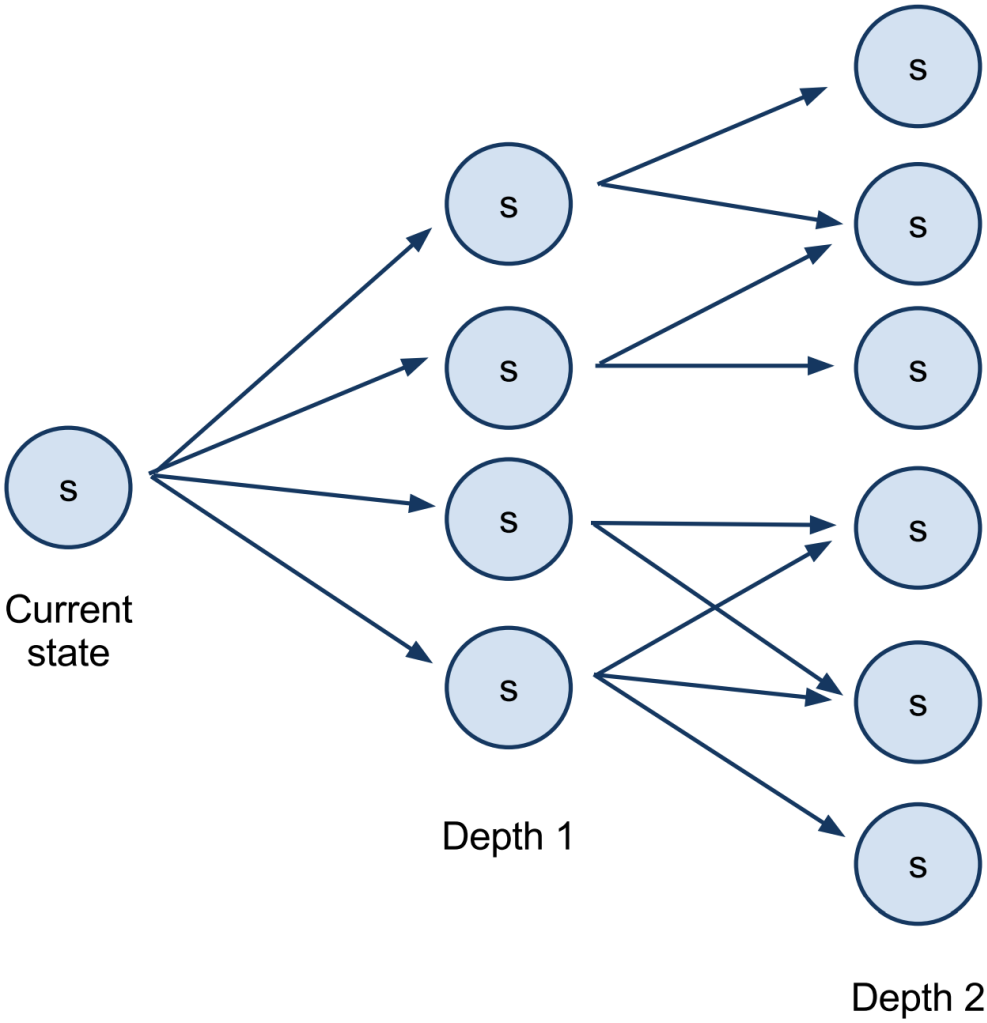

Figure 5: This figure is created by Wiering et al. [35]. It shows a Monte Carlo Tree Search roll-out from the agent's current state out to a depth of 2. The MCTS methods simulate a trajectory(a sequence of state-action pairs $(s_0, a_0, s_1, a_1, \cdots)$) forward from the agent's current state. At each state, a policy will select an action, leading to the next state one level deeper in the tree. After rolling out to a terminal state or a maximum depth, the values of the selected action are updated towards the rewards received following it on that trajectory.

$$UCB(s') = Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \tag{3}$$

- Expansion: randomly create a child node C of a leaf node L that the selection stage reached, unless the leaf node L is a terminal state or maximum depth is reached.

- Simulation: Roll out one or multiple simulations from the expanded node C with reward accumulated for each simulation. A rollout may be as simple as choosing uniform random actions until the terminal state or maximum rollout length is reached. The simulation process evaluates the value of the expanded node C.
- Backup: Use the accumulated rewards in the simulations to update the value of nodes on the path from the expanded node C to the root node R.

2.5 Human-AI collaboration

This thesis proposes an AI assistant that provides advice to the chemist for curriculum design. Only the chemist can take actions in the de novo design molecular tool. This AI assistance framework fits within Human-AI collaboration methods where the AI and human are solving the same problem. Dimitrakakis et al. [12] propose an AI assistant to cooperate with human in a decision process. The AI assistant primarily controls the process, but the human can override the AI's actions. However, their method assumes the AI knows the reward function of the human in advance before cooperation. Hadfield Menell et al. [16] introduce cooperative inverse reinforcement learning, where the AI and the human simultaneously take actions to solve a task. The AI assistant does not initially know the reward function of the task, but it learns the reward function by inverse reinforcement learning when the human demonstrates a policy. In shared autonomy [26], the AI assistant receives a human action command and selects high-value action that are closest to the human command. The AI assistant assists a user without access to the user's reward function, implicitly inferring it through human action commands. In these automation-based methods, the AI assistant is allowed to access to the environment and take actions autonomously. In contrast, the assistant helper action MDP (HAMDP) [13] assists an agent through action recommendations. However, HAMDP assumes the agent will always accept a recommendation for the optimal action, and does not take into account the agent's biases. De Peuter et al. [11] proposed an AI assistant which recommends actions based on the user that it encounters, taking into account the user biases and the private reward function. The study shows that their AI assistant framework significantly outperforms automation methods based on inverse reinforcement learning and preference learning. Therefore, this thesis adopts this AI assistant framework to help chemist design curriculum to solve sparse reward in de novo molecular design.

3 Methods

We describe the method introduced in [11] in this section, and explain how it maps to the AI-assisted curriculum design process. As shown in Figure 2, the chemist takes action based on advice from the assistant. In our problem settings, this means that the chemist chooses a curriculum objective based on a suggested curriculum objective. The assistant helps the chemist solve the curriculum design problem E described in Section 1.2. The goal of the assistant is to maximize the cumulative discounted reward obtained by the chemist through its advice. Therefore, the assistant aims to solve a decision problem with reward function $R(s_i, a_i, s_{i+a}; \omega)$ by its actions(advice).

To create such an assistant, we assume a user model $\hat{\pi}(a \mid s, a'; \theta, \omega)$ is available, which simulates how a chemist will act when receiving advice a'. The model depends on two sets of unobserved parameters: $\omega \in \Omega$ and $\theta \in \Theta$. Ω is the parameter space of the chemist's preference in the reward function and Θ is the parameter space for the possible biases that the chemist has in action selection. We call θ the bias parameters.

3.1 Advice as a decision problem

The assistant's decision problem can be model as A generalized hidden parameter MDP(GHP-MDP)[22]. A GHP-MDP is an MDP in which the transition and reward function are parameterized with unobserved parameters. In this thesis, these parameters are ω and θ, the two unobserved parameters which determine a chemist's reward function and biases. Let $\mathcal{M} = \langle \mathcal{S}, \Omega, \Theta, \mathcal{A}, \mathcal{T}, \hat{\pi}, \mathcal{R}, \gamma, p_{0,s}, p_{0,\omega}, p_{0,\theta} \rangle$. $\mathcal{S}, \mathcal{A}, \gamma$ and $p_{0,s}$ are the state space, chemist action space, discounting rate and the initial state distribution from the curriculum design process E, which is described in problem statement 1.2. The assistant should provide suggested action $a' \in \mathcal{A}$ from the same action space $\mathcal{A}$ as the chemist's, and the state for the assistance problem would be the agent state in de novo design tool. The advice $a'_i \in \mathcal{A}$ provided by the assistant should not restrict the chemist from taking action in E. Only the chemist's action a_i leads the agent to a new state based on the transition function of E. Therefore, the assistant only indirectly influences the change of state, by using advice to induce a different policy from the chemist. The user model $\hat{\pi}(a \mid s, a'; \theta, \omega)$ predicts the chemist's action induced by advice. $\mathcal{T}$ is the transition function class $\mathcal{T} = \{\mathcal{T}_{\omega,\theta}\}_{\omega \in \Omega, \theta \in \Theta}$. The transition function $\mathcal{T}_{\omega,\theta}(s_{i+1} \mid s_i, a'_i)$ defines a distribution of potential next states s_{i+1} given that the chemist received advice a' in the current state s_i. For given reward and bias parameters ω and θ, the transition function is $\mathcal{T}_{\omega,\theta}(s_{i+1} \mid s_i, a'_i) = \sum_{a_i \in \mathcal{A}} \hat{\pi}(a_i \mid s_i, a'_i; \theta, \omega) T'(s_{i+1} \mid s_i, a_i)$. It encodes the interaction between the assistant and the chemist and between the chemist and the agent in de novo design tool from Figure 2. $T'(s_{i+1} \mid s_i, a_i)$ is used to approximate the transition function $T(s_{i+1} \mid s_i, a_i)$ in curriculum design process E. The computation of the next state s_{i+1} requires training the agent with a fixed number of epochs, which is usually sufficient for convergence. For simulation efficiency in the assistant's decision problem $\mathcal{M}$ when planning advice, we use $T'(s_{i+1} \mid s_i, a_i)$ to estimate the next state s_{i+1} by training the agent with fewer epochs.

Since the REINVENT agent state would be only changed by the chemist's action, the reward for the assistant's advice depends on the chemist' action induced by its advice. In other words, its reward is equivalent to the reward that chemist received after taking his/her induced action a, which would be predicted by the user model. Therefore, the reward function should be identical to the reward function $\mathcal{R}$ in curriculum design process E. Lastly, Ω and Θ are the parameter spaces for the reward function and biases under prior distributions $p_{0,\omega}$ and $p_{0,\theta}$ respectively.

3.2 Advice generation

The MDP $\mathcal{M}$ describes the assistant's task in curriculum design. $\mathcal{M}$ essentially defines a space of MDPs $\{\mathcal{M}_{\omega,\theta}\}_{\theta\in\Theta,\omega\in\Omega}$. Each MDP $\mathcal{M}_{\omega,\theta}$ has the unknown transition function $\mathcal{T}_{\omega,\theta}$ and the unknown reward function $\mathcal{R}_{\omega}(s_i, a_i, s_{i+1})$, resulted from two sets of unknown parameters ω and θ. In other words, the assistant does not know what kind of chemist it is advising. However, the assistant can maintain posterior beliefs over unknown parameters ω and θ based on observed transitions (s_i, a'_i, s_{i+1}) and the priors $p_{0,\omega}$ and $p_{0,\theta}$. After providing advice a'_i to the chemist, the assistant observes the transitions by observing what action a_i the chemist takes in state s_i (resulting in state s_{i+1}) in response to advice a'_i. The AI assistant uses Bayesian inference to estimate the true value of the unknown parameters. It calculates the likelihood of the observed transitions under different parameter values in $\Omega \times \Theta$ to maintain a posterior distribution over the chemist's biases and reward function.

This thesis employs *Generalized Hidden Parameter Monte Carlo Planning* (GHPMCP) [11] for planning over the assistant's decision problem $\mathcal{M}$. The GHPMCP algorithm is based on MCTS[7] and enables planning over MDPs where the transition function $\mathcal{T}_{\omega,\theta}$ and the reward function $\mathcal{R}_{\omega}$ are unknown. Though these two function are unknown, the AI assistant applies the maintained beliefs over the unknown parameters Ω and Θ in the GHPMCP algorithm. The GHPMCP algorithm uses the MCTS tree as a sort of particle filter and implicitly represents this belief distribution at all future states present in the tree. This allows it to gauge the information value of actions for future policies.

The GHPMCP algorithm is shown in Algorithm 1. GHPMCP plans multiple time over sampled $\mathcal{M}_{\omega,\theta}$ and provide the action with the highest Q-value. When GHPMCP plans in the AI assistant's decision problem $\mathcal{M}$, it samples ω and θ from a maintained belief distributions over reward parameters p_{ω} and bias parameters p_{θ}. Then, it simulates $\mathcal{M}_{\omega,\theta}$ down the tree up to a maximum depth max_depth, following an Upper Confidence bound for Trees(UCT)[7] policy, which balances the exploration and the exploitation. "The variable h is applied to identify tree nodes by their path to the root and d to represent the current depth. As the simulation progresses down the tree, the state and action node visit counts $N(h, s)$ and $N(h, s, a)$ and Q-values $Q(h, s, a)$ are updated. Simulation proceeds recursively until a leaf node is reach, either because the maximum depth is reached or because a new node is encountered." [11]

Algorithm 1 GHPMCP

function PLAN(s, p_ω, p_θ)
  for $i = 1, \cdots, n\_iterations$ **do**
    $\omega \sim p_\omega$
    $\theta \sim p_\theta$
    $h \leftarrow s$
    SIMULATE$(h, s, \mathcal{M}_{\omega,\theta}, 1)$
  end for
  $h \leftarrow s$
  return $\arg\max_a Q(h, s, a)$
end function
function SIMULATE$(h, s, \mathcal{M}_{\omega,\theta}, d)$
  if $N(h, s) = 0$ **then**
    for all $a \in \mathcal{A}$ **do**
      $N(h, s, a) \leftarrow 0$
      $Q(h, s, a) \leftarrow 0.0$
    end for
  end if
  $a' \leftarrow \arg\max_{a \in \mathcal{A}} Q(h, s, a) + c\sqrt{\frac{\log N(h,s)}{N(h,s,a)}}$ ▷ UCT policy
  $s' \sim \mathcal{T}_{\omega,\theta}(\cdot \mid s, a')$ ▷ A user model involved
  $r \leftarrow \mathcal{R}_\omega(s, a', s')$
  $h' \leftarrow ha's'$
  $d' \leftarrow d + 1$
  if $N(h, s, a) = 0$ **or** $d = max\_depth$ **then**
    $q \leftarrow r$
  else
    $q \leftarrow r + \gamma$SIMULATE$(h', s', M_{\omega,\theta}, d')$
  end if
  $N(h, s) \leftarrow N(h, s) + 1$
  $N(h, s, a) \leftarrow N(h, s, a) + 1$
  $Q(h, s, a) = Q(h, s, a) + \frac{q - Q(h,s,a)}{N(h,s,a)}$
  return q
end function

3.3 User model

The assistant plans the next action to maximize the chemist's cumulative reward. In the simulated decision problem $\mathcal{M}_{\omega,\theta}$, the assistant consider how the chemist will act after receiving its advice a' by the transition function $\mathcal{T}_{\omega,\theta}$. This consideration requires a user model $\hat{\pi}(a \mid s, a'; \theta, \omega)$. Since the user model simulated the chemist's behavior, this section uses user model and chemist interchangeably.

The user model in this thesis is also inspired by De Peuter et al.[11]. The process of how humans make decisions is well-studied and these established theories lead to our user model. The user model simulates how a human make choices by equation 4.

$$p(a \mid u) = \frac{p(a)\exp(\beta u(a))}{\sum_{\hat{a}\in\mathcal{A}} p(\hat{a})\exp(\beta u(\hat{a}))} \tag{4}$$

where $a \in \mathcal{A}$ is an action, u is a utility function that evaluates the action a, $p(a)$ is the prior distribution over actions, and the last parameter $\beta \in \theta$ is a temperature parameter. Equation 4 describes that the probability of taking action a depends on the action value $u(a)$ and human prior preference $p(a)$. β balances the exploration and exploitation on actions. A high temperature ($\beta \to \infty$) results in a greedy exploitative strategy and makes fully ration choices while a low temperature ($\beta \to 0$) would tend to make every action equiprobable (i.e. pure exploration). This choice rule is known as a softmax policy, which is widely applied in RL as an action selection rule. In addition, the softmax choice rule is a compelling model of human cognition that frequently used to model human behavior[20, 1, 31, 10, 6].

Since the chemist selects a curriculum objective to maximize his/her cumulative reward, we apply the chemist's prior knowledge $R_\omega(s_0, a, s_1)$ as the utility function. $R_\omega(s_0, a, s_1)$ defines the improvement in curriculum design objective value for taking action a in the initial state s_0 and ending up in s_1. $R_\omega(s_0, a, s_1)$ provides a score evaluating sampled molecules from the agent s_1 in the production phase. We assume the chemist has prior knowledge $R_\omega(s_0, a, s_1)$ after (s)he creates a set of promising curriculum objectives. In other words, given an prior agent s_0 in de novo design tool, the chemist understands how helpful the action a is to solve the sparse reward in the production phase. Although the agent state will update during curriculum learning, the chemist will continue to use its prior knowledge as the utility function. It is too computationally intensive and time-consuming for chemists to consider $R_\omega(s_i, a, s_{i+1})$ for every possible actions and state pair. In tuition, chemists should prefer the curriculum objective that will cause an agent to produce desired molecules.

With the prior knowledge, we employ the softmax choice rule instead of greedy selection for the following reasons. The greedy action does not necessarily result in an optimal order of curriculum objectives in curriculum design because the prior knowledge is biased as it allows a fixed state for the agent. However, the softmax choice rule allows exploration to overcome the limitation of biased prior knowledge. The greedy action is still most likely to be selected because it has the highest selection probability, but all the others are ranked and weighted according to their value estimates.

When the chemist designs a curriculum, we assume the chemist would select the best curriculum objective they can think of by using the choice rule. Therefore, it is a stochastic action choice which is denoted by random variable A_1. The distribution of A_1 is given in the equation 5. $\mathcal{R}_\omega(a)$, $p(A_1 = a)$ are a simplified notation of $R_\omega(s_0, a, s_1)$, $p(A_1 = a|s_i; \theta, \omega)$ respectively, because the state and parameters are constant in this context.

$$p\left(A_1 = a\right) = \frac{p(a)\exp\left(\beta_1 \mathcal{R}_\omega(a)\right)}{\sum_{\hat{a}\in\mathcal{A}} p(\hat{a})\exp\left(\beta_1 \mathcal{R}_\omega(\hat{a})\right)} \tag{5}$$

After using prior knowledge to select an action choice $a \in A_1$, the chemist chooses whether to switch to the assistant's suggested action a' or insist on his/her original

choice. Since there are only two actions for the chemist to choose from, it is computationally feasible to sample molecules in the production phase to evaluate $R_\omega(s_i, a, s_{i+1})$ and $R_\omega(s_i, a', s'_{i+1})$ to compare the two actions. For the STOP action in the action space, the agent remains in the previous state and thus $R_\omega(s_i, STOP, s_{i+1}) = 0$. If neither the chemist's prior choice a nor the assistant's advice a' improves the curriculum design objective value ($R_\omega(s_i, a, s_{i+1}) < 0$ and $R_\omega(s_i, a', s'_{i+1}) < 0$), then the chemist will choose the STOP action instead to terminate the curriculum design process.

In summary, the user model $\hat{\pi}(a \mid s, a'; \theta, \omega)$ in this GHPMCP algorithm simulates the chemist's behaviors. First, it will apply prior knowledge $\mathcal{R}_\omega(a)$ and the softmax choice rule to select a curriculum objective and then compare that with the assistant's advice using sampled molecules. The user model $\hat{\pi}(a \mid s, a'; \theta, \omega)$ makes a sequence of choices to construct a curriculum and will only settle if the assistant and itself cannot find a curriculum objective that improves the agent further in the production phase.

3.4 Summary

In this section, we have defined the assistant's decision problem as a GHP-MDP. The assistant provides advice to help the chemist take actions, but only the chemist's action can change the state of the agent in the de novo drug design tool. When the assistant is planning the advice, it has to consider the possible reward function and chemist's bias in action selection, which is implemented by GHPMCP. GHPMCP samples possible reward parameters and bias parameters and applies MCTS to estimate the Q-values of the assistant's actions. In MCTS, the state transition considers how the chemist will act, which is simulated by the user model. After adequate simulation, the assistant will choose the action with the highest Q-values.

4 Experiments

4.1 DRD2 activity

Curriculum design aims to solve the sparse reward problem in reinforcement learning. In this thesis, we select DRD2 activity as the property to simulate a sparse reward. The objective of the agent in the de novo drug design tool is to generate molecules that are predicted to be active against a biological target, which is the dopamine type 2 receptor DRD2.

To create a DRD2 activity prediction model, we use the corresponding bioactivity data from REINVENT[2]. The training dataset includes 24080 active molecules and 251688 inactive molecules while the test dataset has 1104 active molecules and 67840 inactive molecules. To create a predictive model that results in a sparse reward problem, we randomly select 100 molecules from the training dataset including 10 active molecules and 90 inactive molecules. We use this tiny imbalanced dataset to train a random forest classifier and use this classifier to simulate sparse rewards in the de novo design tool.

We use the random forest classifier from Scikit-learn, a machine learning toolkit in Python. We extract molecules' Morgan fingerprint, which is one of the supported molecular features in REINVENT. The random forest classifier has 100 trees in the forest and the maximum depth of each tree is 2. The resulting classifier can not recall any active molecules in the test dataset (see Table 1) and can only recall 2 active molecules in the training dataset (see Table 2). The weak DRD2 activity prediction model tends to predict molecules to be inactive, which is desired to produce sparse rewards. We use this model to create a drug design task in which the generated molecules should be predicted to be active. This task has sparse rewards since few molecules can be predicted to be active.

metric	accuracy	recall
value	0.987	0

Table 1: The performance of a weak DRD2 activity classifier on test dataset

metric	accuracy	recall
value	0.92	0.2

Table 2: The performance of a weak DRD2 activity classifier on training dataset

Then, we apply this weak DRD2 activity prediction model in REINVENT to check the performance when converged. REINVENT uses the predictive probability from the model as a score to train the agent. We call this predictive probability DRD2 activity score for the rest of the thesis. As we can see in Figure 6, the generated molecules from the prior agent only get around 0.09 average DRD2 activity score and after 1000 epochs, the average DRD2 activity score converges at around 0.38.

[2]The DRD2 data comes from REINVENT community

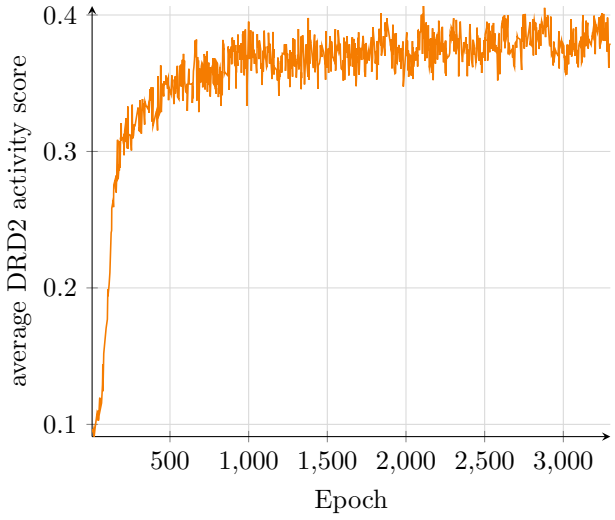

Figure 6: Without curriculum learning, the average score of weak DRD2 activity converges at around 0.38.

4.2 Simulated Chemist

We demonstrate our AI assist curriculum design method with a simulated chemist. We create a simulated chemist which has the same action policy as the user model $\hat{\pi}(a \mid s, a'; \theta, \omega)$. However, the simulated chemist has reward parameters ω and bias parameters θ which they cannot explicitly describe. The AI has to maintain posterior belief over these parameters to estimate θ and ω.

The simulated chemist aims to use a de novo drug design tool to generate molecules that are predicted to be active by the weak DRD2 activity model. The higher the predictive probability the generated molecules achieve, the closer the simulated chemist gets to its goal. Since DRD2 activity is a sparse property in our setting, the simulated chemist uses curriculum learning to mitigate the sparse reward problem. In curriculum design, the objective function $f_\omega(s)$ of the simulated chemist evaluates sampled molecules m of the agent s in the production phase. We assume the chemist would take into account the DRD2 activity, Quantitative Estimate of Drug-likeness(QED) and synthetic accessibility(SA) in molecular evaluation. Thus, the curriculum design objective function $f_\omega(s) = \frac{\sum_i^k \omega_i * \bar{p}_i(m|s)}{\sum_i^k \omega_i}$ consists of these three properties. The reward parameters ω consist of 3 parameters representing the simulated chemist's interests in the three properties and the bias parameter θ is β in the softmax choice rule (equation 4).

The chemists we simulate in the experiments are instantiated with parameter values sampled from the priors $p_{0,\omega}$ and $p_{0,\theta}$. All parameters have independent prior distributions, which are listed in table 3. We assume the DRD2 activity has a higher weight because it is the main goal in curriculum learning and the other two properties have lower weights since they are necessary for the chemist to evaluate molecules. These priors are also used in the beliefs maintained by the AI assistant.

	name	prior distribution	explanation
Ω	ω_1	$\mathcal{N}(0.8, 0.1)$	the weight of DRD2 activity in reward function
	ω_2	$\mathcal{N}(0.1, 0.05)$	the weight of QED in reward function
	ω_3	$\mathcal{N}(0.1, 0.05)$	the weight of SA in reward function
Θ	β	*Unif* $(1, 4)$	bias parameter of the choice in eq. 4

Table 3: Prior distributions for the parameters of the simulated chemist. The left-most column indicates whether the parameter is part of the reward parameters Ω or the bias parameters Θ

4.3 Action Space

To solve the sparse DRD2 activity reward in de novo design, the simulated chemist resorts to curriculum learning. We create a set of 25 curriculum objectives for the simulated chemist and the AI assistant to design the curriculum. The supported scoring components for REINVENT include physicochemical properties, predictive models and similarity measures to some specific structures [5]. For physicochemical properties, we select the number of rotatable bonds, the number of hydrogen bond donors, the number of hydrogen bond acceptors, the number of rings, molecular weights, topological polar surface area, the length of graph, slogp and the number of atomic stereo centers. The definition of these scoring components can be found in the supporting information[3] of REINVENT.

We analyze the distributions of these properties among the active molecules in the dataset and use their 95% confidence interval as criteria to create a set of curriculum objectives. The generated molecules from the de novo design tool get rewards if their property values fall in the 95% confidence interval. For similarity measures, we apply Tanimoto similarity, matching substructure and custom alert. The provided targets in the similarity measures are selected from active molecules in the dataset and prior knowledge. For predictive models, we have the weak DRD2 activity model, Quantitative Estimate of Drug-likeness(QED) from the Rdkit package [19] and a synthetic accessibility model from REINVENT community.

The simulated chemist is given 25 curriculum objectives and must create a curriculum to solve the sparse reward problem. Therefore, the simulated chemist has 26 actions including a STOP action. Once the STOP action is selected, the curriculum design process is terminated. The STOP action will not change the agent state, and thus the cumulative reward remains the same as the one in the previous state. We assume a uniform prior distribution over these 26 actions. In other words, the softmax choice rule in the user model can be simplified to:

$$p\left(A_1 = a\right) = \frac{\exp\left(\beta_1 \mathcal{R}_\omega(a)\right)}{\sum_{\hat{a}\in\mathcal{A}} \exp\left(\beta_1 \mathcal{R}_\omega(\hat{a})\right)} \tag{6}$$

[3] the support information of REINVENT

4.4 Hyperparameters

The number of epochs The transition function $T(s_{i+1}|s_i, a_i)$ of the agent state in the curriculum design problem E is achieved by applying action a_i(a curriculum objective) to the agent s_i and training it for a fixed number of epochs. In our experiments, the number of epochs is 300, as it is usually sufficient for convergence in REINVENT. Since a curriculum objective can be revisited, the a_i would be re-considered if the REINVENT agent has not converged on the curriculum objective. In addition, the sparse reward objective in the production phase will be also be trained with 300 epochs, which would typically be insufficient for convergence without curriculum learning. Therefore, obtaining reward $\mathcal{R}_\omega(s_i, a_i, s_{i+1})$ requires 600 epochs for one action-state (s_i, a_i, s_{i+1}) pair: 300 epochs for the action and 300 epochs for the production phase. After 600 epochs, the chemist samples molecules m for the agent and receives reward $\mathcal{R}_\omega(s_i, a_i, s_{i+1})$. Since calculating the reward $\mathcal{R}_\omega(s_i, a_i, s_{i+1})$ is computational intensive, the chemist will make their prior choice based on their prior knowledge $\mathcal{R}_\omega(a)$ at first and use $\mathcal{R}_\omega(s_i, a_i, s_{i+1})$ to select between their choice and advice. In the assistant's decision problem, the transition function $T\left(s_{i+1} \mid s_i, a_i\right)$ in the curriculum design problem E is approximated by $T'\left(s_{i+1} \mid s_i, a_i\right)$, which is achieved by training the agent with fewer epochs(200) in the production phase.

Planning for Assistance The AI assistant learns the policy of the simulated chemist and provides advice to the simulated chemist. We use the following parameter values in Table 4 for GHPMCP.

name	GHPMCP	explanation
γ	1	discounting rate used while planning
max_depth	1	maximum depth of the tree
n_iterations	1000	number of planning iterations
c	$\frac{1}{\sqrt{2}}$	exploration constant used by the UCT policy

Table 4: Parameters used in GHPMCP. The names of the parameters correspond to those used in Algorithm 1

4.5 Evaluation

4.5.1 Evaluation metrics

AI-assisted curriculum design aims at helping a chemist design a curriculum for curriculum learning, which will be beneficial for optimizing sparse reward in de novo molecular design. Therefore, an appropriate evaluation should include evaluation in the curriculum design stage and performance in optimization.

Here, we use the following two metrics:

1. average DRD2 activity score: We use the average DRD2 activity score as a metric to evaluate an agent in a specific state. Given an agent in de novo molecular tool, we sample 1000 molecules and use the weak DRD2 activity

classifier to predict their activity. The predictive probability is referred as DRD2 activity score.

2. curriculum design objective value: $f_\omega(s) = \frac{\sum_i^k \omega_i * \bar{p}_i(m|s)}{\sum_i^k \omega_i}$ consists of three properties, which are DRD2 activity predicted by the weak DRD2 activity classifier, Quantitative Estimate of Drug-likeness(QED) and synthetic accessibility(SA). The reward parameters ω consist of 3 parameters representing the simulated chemist's interests in the three properties. For a design problem, the curriculum design objective value $f_\omega(s)$ is the most appropriate measure of comparison. It represents the chemist satisfaction with the resulting agent state after taking action in the curriculum design process.

For curriculum design, our measure of interest is the curriculum design objective value achieved by a given time step, where time steps are counted as actions of the simulated chemist. For the sparse reward problem, our measure of interest is the average DRD2 score achieved by a given time step, where time steps are counted as actions of the simulated chemist.

4.5.2 Hypotheses

In this section, we propose the following four hypotheses that we would like to test in our experiments.

H1 *Applying the curriculum design objective function in de novo design tool instead of the original sparse reward can not improve the sparse property (DRD2 activity) of generated molecules.*

H2 *For curriculum design, AI-assisted curriculum design achieve higher curriculum design objective values than unassisted curriculum design and curriculum design with random advice.*

H3 *For the sparse reward problem, curriculum learning would improve the average DRD2 activity scores.*

H4 *AI-assisted curriculum learning achieve higher average DRD2 activity scores than unassisted curriculum learning and random-assisted curriculum learning.*

In our earlier discussion on sparse reward remedies (Section 2.2), we introduce a reward shaping method to mitigate sparse reward problem. Reward shaping enhances the original reward with additional rewards that can fill the gap in the original sparse reward. In our curriculum design, the objective function incorporates the sparse reward and other two properties, which originate from chemist's biases in molecular evaluation and enhance the sparse reward. While the curriculum design objective function has additional rewards, we would like to propose the first hypothesis **H1**, because the drug-likeness and synthetic accessibility are weakly correlated to the sparse DRD2 activity. The weak correlation fails to incentivize the agent to produce DRD2 active molecules. Since the chemist cannot describe their curriculum

design objective function directly, we must learn an approximate objective function through human interaction and then subsequently apply it in the de novo drug design tool. However, as we use a simulated chemist, we know the exact curriculum design objective function once we sample a simulated chemist. We apply the exact curriculum design objective function in de novo design tool to overestimate the result of using an approximate objective function. The result is measured by the average DRD2 activity score of generated molecules, which is in range $[0, 1]$.

Curriculum design helps chemists design a curriculum for curriculum learning. To test the second hypothesis **H2**, we compare our AI-assisted curriculum design method with the result of an unassisted simulated chemist and assisted with random advice. In our experiments, the unassisted simulated chemist selects curriculum objectives based on its prior knowledge.

The goal of curriculum learning is mitigating the sparse reward problem. We test the following hypotheses **H3** and **H4** to show the utility of AI-assisted curriculum design.

To test these hypotheses, we need five experiments including AI-assisted curriculum learning, unassisted curriculum learning, random-assisted curriculum learning, reinforcement learning with sparse reward and reinforcement learning with curriculum design objective function. All experiments except reinforcement learning with sparse reward require a simulated chemist. These comparative experiments are run ten times. Every run of the experiments uses a new randomly sampled simulated chemist with different reward function and biases. In every run we apply four experiments once.

We use a two-sided paired Wilcoxon signed rank test [36] to verify significance. We define the independent paired samples as the measured values at the same time step for two different comparative experiments but the same simulated chemist. As a result, we can collect the same number of paired samples as the number of simulated chemists we sampled. In our implementation, we use the Wilcoxon signed-rank test from SciPy statistics package [32].

4.6 Results

For curriculum design, we compare AI-assisted curriculum design to unassisted curriculum design and curriculum design with random advice. These three comparative experiments require a simulated chemist. Different simulated chemists and different curriculum design methods will result in a different curriculum, including the number of curriculum objectives and the type of curriculum objectives. We extend the measurement for every experiment until the max time step that one of the experiments achieved by filling the measured values at the final time step in every experiment. For example, assuming a AI-assisted curriculum design takes the maximum action (6 actions or time steps), the measured values of every design experiment will be extended to 6 time steps. This is done by adding STOP actions to the trajectory until it reaches the desired length. STOP actions have no effect on the evaluation, as they do not change the agent state and have zero reward. Although we extend the measured values, we will list the average actions for each design method takes to

compare the time costs.

4.6.1 Results of Curriculum Design

We provide evidence towards **H2** through three methods, including AI-assisted curriculum design, unassisted curriculum design and curriculum design with random advice. The curriculum design objective value achieved by all methods is shown in Figure 7 and detailed results with significance tests can be found in Table 5.

In time step 0, the prior REINVENT agent is presented as a low curriculum design objective value since it has not started curriculum learning. We observe that the AI-assisted curriculum design significantly (p-value<0.01) outperforms the unassisted curriculum design and curriculum design with random advice after time step 1. The unassisted simulated chemists rarely improve their curriculum design objective value after the first curriculum objective is selected. In addition, we see that the curriculum design with random advice is only slightly better than the unassisted approach, since it offers an alternative action for the simulated chemist to consider. Even though the action space has only 26 actions, random recommendations do not prove tremendously helpful. On the other hand, the AI assistant uses the GHPMCP algorithm to plan advice based on a user model and offers the advice with the highest Q-values after multiple simulations. As we can see, advice with careful planning enables AI-assisted curriculum design to achieve higher curriculum design objective value than the other two methods, which proves the hypothesis **H2**.

Note that at each time step, the simulated chemist chooses a curriculum objective for curriculum learning or takes the STOP action because of the extension. In our 10 repetitive experiments for each method, the simulated chemist in AI-assisted curriculum design takes more actions (4.6 time steps) than the other two methods (2.6 time steps is the least one). Note, however, that AI-assisted curriculum design significantly outperforms the other two methods in the first three time step. The advantage of AI-assisted curriculum design is revealed from the second time step and the curriculum design objective values keep improving as time step increases.

time step	1	2	3	4	5	6	average time step
unassisted	0.371±0.015	0.388 ± 0.015	0.402 ± 0.021	0.406 ± 0.021	0.409 ± 0.020	0.409 ± 0.020	2.6
random	0.371± 0.013	0.405 ± 0.024	0.416± 0.023	0.416± 0.022	0.417± 0.022	0.417 ± 0.022	3.6
assisted	0.372 ± 0.014	**0.436± 0.028**	**0.453± 0.029**	**0.460± 0.028**	**0.464±0.030**	**0.477 ±0.037**	4.6

Table 5: Mean curriculum design objective value ± standard error as a function of time for by three different design methods. Bold shows significant improvement of assisted curriculum design (p-value<0.01) over the alternatives within the same time step. The average time step indicates the average number of actions taken by the simulated chemist in the curriculum design.

4.6.2 Improvement on Sparse reward problem

We translate curriculum design to curriculum learning when the context switches from design problem to sparse reward problem. Selecting a curriculum objective

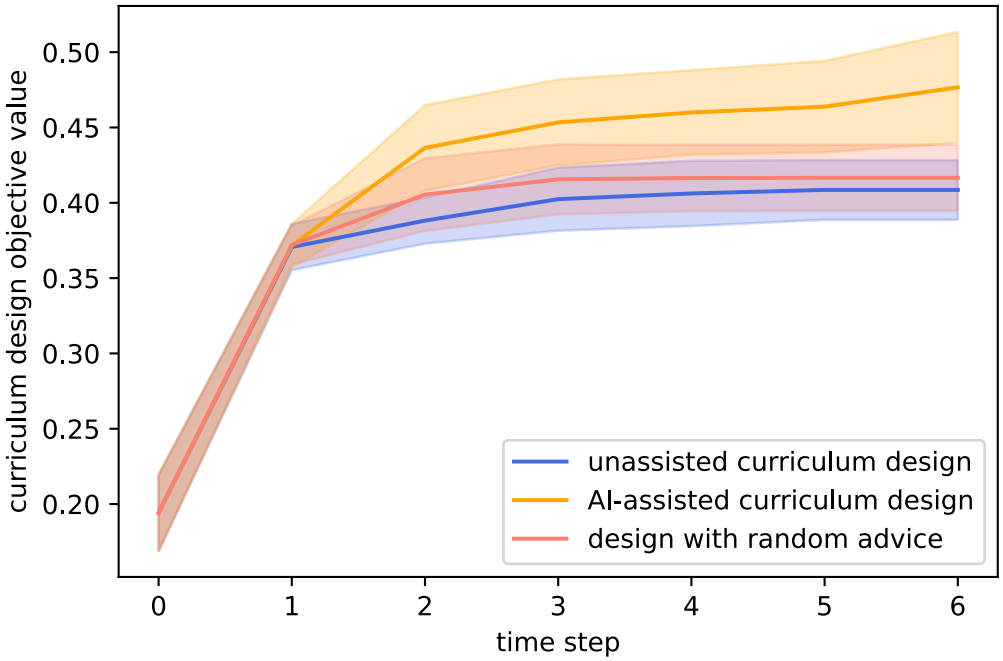

Figure 7: Curriculum design objective value achieved

in curriculum design corresponds to learning a curriculum objective in curriculum learning. For the sparse reward problem, we have three hypotheses: **H1**, **H3** and **H4**. We ran five different experiments to provide evidence towards these hypotheses, including AI-assisted curriculum learning, unassisted curriculum learning, random-assisted curriculum learning, reinforcement learning with curriculum design objective function as reward and reinforcement learning with DRD2 activity as reward.

As we can see the average DRD2 activity score in Figure 8 and detailed results with significance tests in Table 6, the reinforcement learning with the original reward significantly outperforms (p-value$<$0.01) the one with the curriculum design objective function, in which the additional reward is weakly correlated to the sparse reward. Thus, we verify the hypothesis **H1**. Reward shaping requires the additional reward appropriately rewards or punishes the agent's actions, filling the gap of the original sparse reward. The results indicate that the curriculum design objective function can not be simply used as a reward shaping technique in reinforcement learning. Conversely, the curriculum design objective function introduces the other two properties because of the chemist's biases on molecular evaluation.

Next, we see that the average DRD2 activity score of curriculum learning is better than the result of reinforcement learning, which verifies the third hypothesis **H3**. The unassisted curriculum learning is only slightly better than reinforcement learning with sparse reward, since the simulated chemist uses prior knowledge to create the curriculum. With random advice, the simulated chemist is offered an alternative

to consider and thus the result is better than the unassisted curriculum learning. Again, although the action space has only 26 actions, random-assisted curriculum learning does not offer as good advice as AI assisted curriculum learning. Lastly, we see the AI-assisted curriculum learning significantly outperform the other methods from time step 2 and the average DRD2 activity score eventually reach about 0.52. This result proves **H4**.

In reinforcement learning with DRD2 activity, the average DRD2 score converges at around 0.38 and it rarely improves even after training with 3300 epochs, as shown in Figure 6. With AI assisted curriculum learning, the average DRD2 score reaches about 0.52 at time step 6. We see that the AI-assisted curriculum learning significantly improve the DRD2 activity score with less training time.

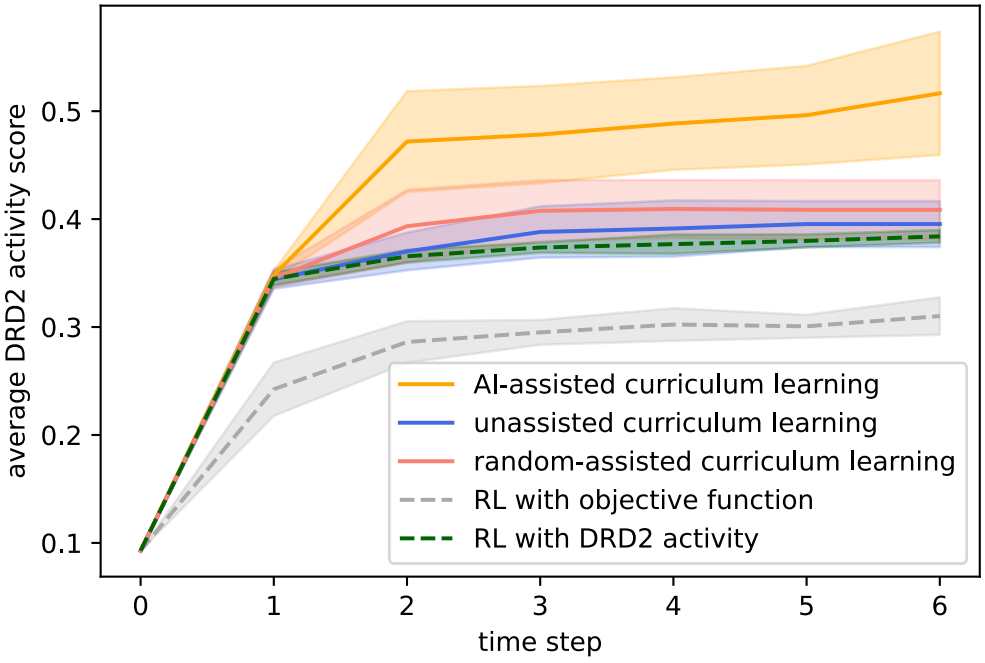

Figure 8: Average DRD2 activity score achieved

4.6.3 Summary

In summary, through our comparative experiments and results, we verify four hypotheses (**H1**-**H4**) and get the following conclusions.

1. Applying the curriculum design objective function in de novo design tool instead of the original sparse reward can not improve the sparse property (DRD2 activity) of generated molecules.

time step	1	2	3	4	5	6	average time step
objective function	0.242± 0.024	0.286 ± 0.019	0.295± 0.011	0.302±0.015	0.301±0.011	0.310 ±0.017	-
DRD2 activity	0.345± 0.006	0.366 ± 0.006	0.374± 0.005	0.377±0.009	0.380±0.006	0.384 ±0.006	-
unassisted	0.344±0.009	0.370 ±0.017	0.388±0.024	0.391±0.026	0.395±0.021	0.395 ±0.021	2.6
random	0.345± 0.008	0.393 ± 0.034	0.408± 0.028	0.409±0.027	0.409±0.027	0.409 ±0.027	3.6
assisted	0.348 ± 0.004	**0.472± 0.047**	**0.478± 0.045**	**0.488± 0.043**	**0.496±0.046**	**0.517 ±0.057**	4.6

Table 6: Mean of average DRD2 activity score ± standard error at different time steps by different methods. Bold shows significant improvement of assisted curriculum learning (p-value<0.01) over the alternatives within the same time step. The average time step indicates the average number of curriculum objectives learned in the curriculum learning. For reinforcement learning, there is no extension and each time step is equivalent to 300 epochs.

2. For curriculum design, AI-assisted curriculum design achieve higher curriculum design objective values than unassisted curriculum design and curriculum design with random advice.

3. For the sparse reward problem, curriculum learning would improve the average DRD2 activity scores.

4. AI-assisted curriculum learning achieve higher average DRD2 activity scores than unassisted curriculum learning and random-assisted curriculum learning.

5 Summary

This thesis aims to mitigate the sparse reward problem in de novo molecular design. Sparse rewards are an inevitable problem in drug design when specific biochemical properties are expected. The synthetically feasible molecular space is on the order of $10^{60} - 10^{100}$ [24] while only a small fraction of molecules possess specific biochemical properties. Among sparse reward remedies, curriculum learning is a promising method to mitigate the sparse reward problem and it has been successfully applied in de novo molecular design[15]. Prior approaches required a chemist to hand-craft a curriculum for the agent, which requires domain knowledge and is time-consuming, especially as tasks grow in complexity. In addition, the chemist is required to design a curriculum progression criterion, which is typically hard to do as it cannot promise success in the next curriculum objective.

This thesis defines the curriculum design process as an infinite-horizon Markov decision process, in which the chemist focuses on selecting curriculum objectives and decides when to enter the production phase by selecting the STOP action. We applied an AI-assistance framework[11] to help this decision process. This AI-assistance framework assumes an available user model, which simulates how the chemist takes action after receiving the assistant's advice. The AI assistant infers what kind of the chemist it is advising through the interaction between the chemist and AI and the interaction between the chemist and the de novo drug design tool. The AI assistant estimates the Q-values of actions based on the possible chemists that it could encounter and recommend the action with the best Q-value. The assistant only provides advice to the chemist and only the chemist can take action in the de novo design tool. The assistant must convince the chemist to adopt its advice. Advice also reduces the negative effects of value misalignment in the assistant. The chemist can simply reject advice from a misaligned assistant. These elements present a significant improvement in AI safety. [11]

We demonstrate the AI-assistance framework with a simulated chemist in a de novo design task, where the generated molecules should be predicted by a weak DRD2 activity classifier. Through our experiments, we show that AI-assisted curriculum learning significantly outperforms random-assisted curriculum learning, unassisted curriculum learning, reinforcement learning with DRD2 activity and reinforcement learning with the curriculum design objective function. We show that curriculum learning can mitigate the sparse reward problem and that the AI-assistance framework help to improve the curriculum design. Therefore, AI-assisted curriculum learning significantly mitigates the sparse reward problem.

6 Future works

First, we assume the user model makes a prior choice based on prior knowledge and then compares it with the advice. The user model is a naive version and it could be further improved. Second, the AI assistant approximates the transitions of the agent state by running REINVENT with few epochs. In the future, a state transition estimator could provide a better approximation. Lastly, this thesis uses a weak DRD2 activity classifier as a sparse reward and creates only 25 promising curriculum objectives by statistically analyzing the distribution of each molecular property. More sparse reward applications and methods to create curriculum objectives could be explored in the future.

References

[1] Chris L Baker, Rebecca Saxe, and Joshua B Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[4] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[5] Thomas Blaschke, Josep Arús-Pous, Hongming Chen, Christian Margreitter, Christian Tyrchan, Ola Engkvist, Kostas Papadopoulos, and Atanas Patronov. Reinvent 2.0: an ai tool for de novo drug design. *Journal of Chemical Information and Modeling*, 60(12):5918–5922, 2020.

[6] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[7] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[8] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[10] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[11] Sebastiaan De Peuter and Samuel Kaski. Zero-shot assistance in novel decision problems. *arXiv preprint arXiv:2202.07364*, 2022.

[12] Christos Dimitrakakis, David C Parkes, Goran Radanovic, and Paul Tylkin. Multi-view decision processes: the helper-ai problem. *Advances in neural information processing systems*, 30, 2017.

[13] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. A decision-theoretic model of assistance. *Journal of Artificial Intelligence Research*, 50:71–104, 2014.

[14] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

[15] Jeff Guo, Vendy Fialková, Juan Diego Arango, Christian Margreitter, Jon Paul Janet, Kostas Papadopoulos, Ola Engkvist, and Atanas Patronov. Improving de novo molecular design with curriculum learning. 2021.

[16] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[18] Maria Korshunova, Niles Huang, Stephen Capuzzi, Dmytro S Radchenko, Olena Savych, Yuriy S Moroz, Carrow Wells, Timothy M Willson, Alexander Tropsha, and Olexandr Isayev. A bag of tricks for automated de novo design of molecules with the desired properties: application to egfr inhibitor discovery. 2021.

[19] Greg Landrum. Rdkit documentation. *Release*, 1(1-79):4, 2013.

[20] Christopher Lucas, Thomas Griffiths, Fei Xu, and Christine Fawcett. A rational model of preference learning and choice prediction by children. *Advances in neural information processing systems*, 21, 2008.

[21] Maja J Mataric. Reward functions for accelerated learning. In *Machine learning proceedings 1994*, pages 181–189. Elsevier, 1994.

[22] Christian Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized hidden parameter mdps: Transferable model-based rl in a handful of trials. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5403–5411, 2020.

[23] Jan Peters and J Andrew Bagnell. Policy gradient methods. *Scholarpedia*, 5(11):3698, 2010.

[24] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679, 2013.

[25] Sebastien Racaniere, Andrew K Lampinen, Adam Santoro, David P Reichert, Vlad Firoiu, and Timothy P Lillicrap. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*, 2019.

[26] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Shared autonomy via deep reinforcement learning. *arXiv preprint arXiv:1802.01744*, 2018.

[27] Nalini Schaduangrat, Samuel Lampa, Saw Simeon, Matthew Paul Gleeson, Ola Spjuth, and Chanin Nantasenamat. Towards reproducible computational drug discovery. *Journal of cheminformatics*, 12(1):1–30, 2020.

[28] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2018.

[29] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, pages 1–40, 2022.

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[31] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. *Advances in neural information processing systems*, 23, 2010.

[32] Pauli Virtanen, Ralf Gommers, Evgeni Burovski, Travis E Oliphant, Warren Weckesser, David Cournapeau, Pearu Peterson, Tyler Reddy, Josh Wilson, Matt Haberland, et al. scipy/scipy: Scipy 1.5. 3. *Zenodo*, 2021.

[33] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[34] David Weininger, Arthur Weininger, and Joseph L Weininger. Smiles. 2. algorithm for generation of unique smiles notation. *Journal of chemical information and computer sciences*, 29(2):97–101, 1989.

[35] Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012.

[36] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[37] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.