Entropy-Reinforced Planning with Large Language Models for Drug Discovery

Xuefeng Liu [1 2 *] **Chih-chan Tien** [1 2 *] **Peng Ding** [1] **Songhao Jiang** [1 2] **Rick L. Stevens** [1 3]

Abstract

The objective of drug discovery is to identify chemical compounds that possess specific pharmaceutical properties toward a binding target. Existing large language models (LLMs) can achieve high token matching scores in terms of likelihood for molecule generation. However, relying solely on LLM decoding often results in the generation of molecules that are either invalid due to a single misused token, or suboptimal due to unbalanced exploration and exploitation as a consequence of the LLM's prior experience. Here we propose ERP, Entropy-Reinforced Planning for Transformer Decoding, which employs an entropy-reinforced planning algorithm to enhance the Transformer decoding process and strike a balance between exploitation and exploration. ERP aims to achieve improvements in multiple properties compared to direct sampling from the Transformer. We evaluated ERP on the SARS-CoV-2 virus (3CLPro) and human cancer cell target protein (RTCB) benchmarks and demonstrated that, in both benchmarks, ERP consistently outperforms the current state-of-the-art algorithm by 1-5 percent, and baselines by 5-10 percent, respectively. Moreover, such improvement is robust across Transformer models trained with different objectives. Finally, to further illustrate the capabilities of ERP, we tested our algorithm on three code generation benchmarks and outperformed the current state-of-the-art approach as well. Our code is publicly available at: https://github.com/xuefeng-cs/ERP.

1. Introduction

The emergence and rapid evolution of COVID-19 (Yuki et al., 2020; Hadj Hassine, 2022) has created an urgent need for the expedited discovery of effective drugs—a process in which approaches based on large language models (LLMs) have begun to play a pivotal role (Frey et al., 2023; Bagal et al., 2021).

Transformer (Vaswani et al., 2017) and Transformer-based LLMs such as GPT (Brown et al., 2020; Ouyang et al., 2022; Achiam et al., 2023) and BERT (Devlin et al., 2018) have shown remarkable success in diverse fields, from robotics (Vemprala et al., 2023) to education (Tang et al., 2023) and speech recognition (Yang et al., 2021). However, LLM applications in drug discovery and molecular design have not yet achieved comparable levels of performance—shortcomings that we attribute to the challenges of generating molecular sequences that both are valid (difficult given that even a single erroneous token can lead to complete failure) and meet multiple other criteria simultaneously.

A consequence of these challenges is that existing LLM-based molecular discovery algorithms (Frey et al., 2023; Bagal et al., 2021) must sample inordinate numbers of molecular structures. The primary factor contributing to this sample inefficiency is the use of the Transformer beam search algorithm. While often an effective heuristic for text generation (Meister et al., 2020), beam search has two major deficiencies when used to generate molecular structures: it can neither terminate generation early if it detects that the molecular sequence being produced is likely to fail, nor guide the generation process toward structures with superior pharmaceutical properties.

Meanwhile, planning algorithms like Monte Carlo Tree Search (MCTS) have shown promise for decision making problems. Since the introduction of the Upper Confidence Bound applied to Trees (UCT) method (Kocsis and Szepesvári, 2006), MCTS has demonstrated exceptional performance in various domains, notably AlphaGo (Silver et al., 2017), where MCTS was integrated with deep neural networks (DNNs). However, when applied to molecular design it still incurs a non-trivial search cost on unpromising molecules and thus also suffers from sample inefficiency.

Given the vast search space for molecular sequences, rely-

[*]Equal contribution [1]Department of Computer Science, University of Chicago, Chicago, IL, USA [2]Healin-AI LLC, Chicago, IL, USA [3]Argonne National Laboratory, Lemont, IL, USA. Correspondence to: Xuefeng Liu <xuefeng@uchicago.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

ing solely on the MCTS planner or combining it with conventional DNN approaches may not identify high-reward molecules efficiently. This is where we demonstrate the pretrained molecular Transformer proves to be invaluable for molecular design. Recently, Planning-Guided Transformer Decoding (PG-TD) (Zhang et al., 2023a) has employed the MCTS planning algorithm to guide the Transformer decoder for code generation with a single pass/fail reward function. However, PG-TD adapts the Transformer decoder's next token selection, which might lead to the over-exploitation of certain tokens while failing to explore regions of uncertainty where the true optimal solution might be hidden. In this study, we ask: How can we improve the exploration and exploitation mechanisms in the MCTS planning-guided Transformer decoding process for multiple properties enhancement in drug discovery?

To address this challenge, we introduce a novel algorithm, **E**ntropy-**R**einforced **P**lanning for Transformer Decoding, or ERP, which integrates an entropy based planning algorithm into the Transformer-based generation process. Specifically, for the selection procedure, ERP goes beyond pure MCTS, which only considers the visitation numbers of parent and child nodes in the bonus term. Instead, ERP applies the Transformer decoder for the next token probability and an e-step forward entropy measurement term to evaluate the potential uncertainty following the token. Moreover, the entropy-reinforced MCTS planner directs the Transformer decoder towards exploring promising underlying molecular spaces, instead of merely optimizing the likelihood of the sequences it generates, and also enables the creation of more controllable candidate molecules. Specifically, ERP incorporates the Transformer's next token probabilities and the e-steps ahead confidence method for assessing token uncertainties into the MCTS planner's Upper Confidence Bound (UCB)-based exploration bonus function. In so doing, it increases the efficiency of sampling sequences that yield higher rewards. Experiments show that the incorporation next token probability and the e-step forward entropy measurement boosts the normalized reward by 5-10 percent. In the expansion stage, ERP uses both the TOP-K and TOP-P approaches of the Transformer to select the most likely next token given the current state, improving sample efficiency by excluding non-promising action spaces. For the evaluation stage, we employ the Transformer's beam search to estimate the reward for an incomplete, partial molecular sequence and then backpropagate the value to ancestor nodes.

Our contributions are as follows:

We present a novel algorithm, ERP, which employs an Entropy-Reinforced Planner to guide the decoding process of a pretrained Transformer. This algorithm combines LLMs and planning algorithms to accelerate drug discovery by enhancing the quality, diversity, and sample efficiency of generated molecules.

We introduce a novel selection algorithm, P$\mathcal{H}$-UCT, that takes into account the Transformer's next token probability and incorporates an e-step forward entropy measurement for token selection. These methods reduce uncertainty and improve exploration and exploitation in decision-making.

Lastly, we conduct extensive experiments on the 3CLPro (PDBID: 7BQY) SARS-CoV-2 protein and RTCB (PDBID: 4DWQ) human cancer cell protein benchmarks to compare ERP with PG-TD (Zhang et al., 2023a), UCT (Kocsis and Szepesvári, 2006), beam-search, and sampling algorithm from baseline Transformer models. Our results show that ERP outperforms the current state-of-the-art (PG-TD) as well as all baseline models across these experiments. Moreover, ERP can leverage general pretrained models to improve sample efficiency, correct biased models through controlled generation, and achieve continuous improvement over reinforcement learning (RL) well-fine-tuned Transformer models by balancing exploitation and exploration for potentially higher-reward molecular spaces. To further illustrate the capabilities of ERP, we conducted experiments on three code generation benchmarks and demonstrated its superiority compared to the current state-of-the-art approach.

2. Related Work

We review related work in LLM-based drug discovery, planning and RL for molecule generation, interactive imitation learning, and planning in natural language generation.

2.1. LLM-based drug discovery

LLM generative capabilities have been applied to molecule generation (Bagal et al., 2021; Rothchild et al., 2021; Wang et al., 2022b) in works like MolGPT (Bagal et al., 2021), ChemGPT (Frey et al., 2023) and C5T5 (Rothchild et al., 2021), and to drug discovery (Liu et al., 2023a; Seidl et al., 2023), with molecules represented via a linear string representation, such as simplified molecular-input line-entry system (SMILES) (Weininger, 1988) or self-referencing embedded strings (SELFIES) (Krenn et al., 2020). These approaches have at times demonstrated performance on par with conventional methods that use traditional molecular representations, with the potential to provide distinctive predictive outcomes. Nevertheless, they remain limited to basic molecular transformations and are only capable of supporting roles in molecular design procedures. As noted by Murakumo et al. (2023), the direct creation of molecules through pre-trained LLMs generally results in less impressive outcomes. In contrast, our approach integrates a tree search planning algorithm and multiple critics to steer pretrained LLMs toward generating higher-quality molecules.

2.2. Planning and RL for molecule generation

Several researchers (Yang et al., 2017; 2020; Yoshizawa et al., 2022; M. Dieb et al., 2017; Hong et al., 2023) have integrated planning algorithms like MCTS into molecule generation. However, those workers focused on objectives outside of drug discovery, such as material science; eschewed (in order to enhance sample efficiency) the use of LLMs; and/or overlooked incorporating confidence in their exploratory processes. Online RL has been applied to improve molecule generation, both on SMILES string representations (Born et al., 2021; Guimaraes et al., 2017; Neil et al., 2018; Olivecrona et al., 2017; Popova et al., 2018; Ståhl et al., 2019; Tan et al., 2022; Wang et al., 2022a; Zhang et al., 2023b; Zhou et al., 2019) and graph-based representations (Atance et al., 2022; Gottipati et al., 2020; Jin et al., 2020; Wu et al., 2022; You et al., 2018). As we describe in the following, we employ RL but with guided exploration by an MCTS planner. In this regard, our problem setting aligns more closely with imitation learning, which utilizes an expert to guide the learning process.

2.3. Interactive imitation learning

Imitation learning (IL) is a machine learning technique in which an agent learns to perform tasks by observing and mimicking the actions of an expert. Unlike pure RL, which struggles with large action and state spaces, IL has shown to be more sample-efficient and effective in environments with sparse rewards due to its ability to leverage expert guidance (Ross et al., 2011). Interactive IL methods such as DAgger (Ross and Bagnell, 2014), MAPS (Liu et al., 2023d), and RPI (Liu et al., 2023c) employ a Roll-in-Roll-out (RIRO) strategy to provide guidance. The learner first performs a default roll-in from the initial state and then adopts the expert's guidance to roll out the subsequent steps in the trajectory, thereby correcting the learner's behavior. In contrast, we use an MCTS planner to guide the whole decoding process of the Transformer and rely on the Transformer's beam search to complete the trajectory by forming a whole molecule. Instead of engaging the learner only in the roll-in phase and expert only during the roll-out phase, we involve the Transformer decoder in both the roll-in and roll-out phases.

2.4. Planning in natural language generation

Scialom et al. (2021), Leblond et al. (2021), and Chaffin et al. (2021), among others, have applied the MCTS planning algorithm to optimize text outputs for various NLP tasks. PG-TD (Zhang et al., 2023a) focuses on code generation using a single reward function (pass or fail). Our work here is the first to integrate a tree search planning algorithm with an LLM decoder specifically for de novo drug discovery. Unlike previous approaches that employed pretrained LLMs without evaluating the potential certainty of each generated token and that optimized for only a single objective, we introduce an e-step forward entropy measurement, with the reward computed based on multiple criteria. We show that by combining the entropy-reinforced planner with the Transformer decoder, our algorithm balances exploration and exploitation in the molecular space and thus discovers molecules with high rewards.

3. Preliminaries

We describe LLM, MCTS, and drug discovery along with their mathematical notation in the subsequent sections and unify them under the Markov decision processes framework.

Markov decision processes. We consider a finite-horizon Markov decision process (MDP) (Puterman, 2014) $\mathcal{M}_0 = \langle \mathcal{S}, \mathcal{A}, H, \mathcal{P}, R \rangle$ where $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; H represents the planning horizon; $\mathcal{P}$ is the deterministic transition dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}'$, which concatenates a state s with a token a, and an episode ends when the agent performs the termination action; and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, which only scores a complete molecule (the reward of a partial molecule is 0). The initial state distribution is d_0. The policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ assigns a distribution over actions based on the current state. The Q-value function, expressed as $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, evaluates the performance of the policy.

LLM. We represent each molecule as a state s comprising a start token [BOS], a SMILES (Weininger, 1988) string, and (for both partial and complete molecules) a termination action [EOS]. All possible molecules form the state space $\mathcal{S}$. Let $\mathcal{Y}_t$ be the hypothesis space in step t (sequence length t). We have $\mathcal{Y}_t \subseteq \mathcal{S}_t \subseteq \mathcal{S}|_{t\in[H]}$. We define the set of complete hypotheses, i.e., complete molecules, as

$$\mathcal{Y}_H := \{[\text{BOS}] \circ \mathbf{v} \circ [\text{EOS}] \mid \mathbf{v} \in \mathcal{V}^*\}, \tag{1}$$

where $\mathcal{V}^*$ represents the Kleene closure of $\mathcal{V}$ and $\circ$ denotes string concatenation. Each action $a \in \mathcal{A}$ is represented as token y in the Transformer's vocabulary $\mathcal{V}$, $\mathcal{V} := \mathcal{A}$. Given a set of molecules $\mathcal{B}$, we train the LLM generator policy π_θ to acquire prior knowledge that guides generation of valid molecules. The generator policy π_θ, parameterized by a DNN with learned weights θ, is defined as the product of probability distributions: $\pi_\theta(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} \pi_\theta(y_t|\mathbf{x}, \mathbf{y}_{<t})$, where $\pi_\theta(\cdot|\mathbf{x}, \mathbf{y}_{<t})$ is a distribution, $\mathbf{y}_{<1} = y_0 := [\text{BOS}]$, and $\mathbf{x}$ is an input sequence.

Recall that the decoding process in text generation seeks to identify the most likely hypothesis from all potential candidates by solving the optimization problem:

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}\in\mathcal{Y}_H} \log \pi_\theta(\mathbf{y}|\mathbf{x}). \tag{2}$$

To estimate the expected reward for a partial molecule,

we employ beam search to navigate the exponentially vast search space to form a complete valid molecule. Beam search (breadth-first search with breadth limited to $b \in \mathbb{R}^+$: the 'beam') can be expressed as the following recursion:

$$\mathcal{Y}_0 = \{[\text{BOS}]\}, \tag{3}$$

$$\mathcal{Y}_t = \underset{\mathcal{Y}' \subseteq \mathcal{D}_t, |\mathcal{Y}'|=b}{\arg\max} \log \pi_\theta(\mathcal{Y}'|\mathbf{x}), \tag{4}$$

where $\mathcal{D}_t = \{\mathbf{y}_{t-1} \circ y \mid \mathbf{y}_{t-1} \in \mathcal{V} \text{ and } \mathbf{y}_{t-1} \in \mathcal{Y}_{t-1}\}$ is the candidate set. Because this method focuses on maximizing likelihood without concern for any specific metric of interest, it is not readily adaptable to optimize objectives that differ from those in its training set, as encoded in π_θ. Therefore, we cannot directly employ these generation algorithms to create molecules that achieve a different objective like a significant docking score for a specific target site.

MCTS. We employ a MCTS planning algorithm to identify the optimal policy in a MDP. This algorithm employs a tree structure for its search, where each node represents a state s and each edge an action a. For every node s, the algorithm tracks the visit count $N(s)$. Additionally, $N(s,a)$ represents the frequency with which action a has been selected from state s during the tree's construction. The algorithm maintains an action value function, $Q(s,a)$, representing the best reward received by beginning in state s and executing action a.

UCT (Kocsis and Szepesvári, 2006) (Upper Confidence bounds applied to Trees) adopts the UCB (Auer et al., 2002) algorithm from Multi-Arm bandit for node selection:

$$\text{UCB} = Q(s,a) + c_p \cdot \sqrt{\frac{\log(N(s))}{N(s,a)}}, \tag{5}$$

so as to balance between exploiting states known to be advantageous and exploring those less visited. Value-guided MCTS (V-MCTS) (Silver et al., 2017; Zhang et al., 2023a) improves UCT's sample efficiency by integrating both a policy π and a value network V, yielding P-UCT:

$$\text{P-UCT}(s) = \underset{a \in \mathcal{A}}{\arg\max}\, \text{P-UCB}(s,a), \tag{6}$$

where:

$$\text{P-UCB}(s,a) = Q(s,a) + c_p \cdot \pi_\tau(a|s) \cdot \frac{\sqrt{\log(N(s))}}{1+N(s,a)},$$

c_p is a tunable constant, and τ represents a temperature parameter that modifies the policy $\pi_\tau(a|s) = \frac{\pi(a|s)^{\frac{1}{\tau}}}{\sum_b \pi(b|s)^{\frac{1}{\tau}}}$. PG-TD (Zhang et al., 2023a), a state-of-the-art implementation of V-MCTS, uses a pre-trained Transformer as the policy network π and applies it to code generation.

Drug generation process. Following Liu et al. (2023b), we formalize the drug discovery problem within the context of Markov decision processes (MDP). Our goal is to train a generative policy π_θ to generate a high-quality sequence denoted as $Y_{1:H} = (y_1, \ldots, y_t, \ldots, y_H), y_t \in \mathcal{V}$. At each time step t, the state s_{t-1} includes the tokens generated so far, $(y_1, \ldots, y_{t-1})$, and the action a corresponds to choosing the subsequent token y_t. We want the generative policy π_θ, when starting from an initial state Y_1, to maximize the expected final reward for a complete sequence:

$$J(\theta) = \mathbb{E}_{Y_1 \sim d_0}[r_H|\theta], \tag{7}$$

where r_H is the reward calculated for a generated sequence.

Limitation of previous works: Conventional MCTS faces computational challenges when searching for high-quality molecules. Relying solely on pre-trained LLM decoding for drug discovery results in inconsistencies in generating valid and high-quality molecules and lacks adaptability for different objectives. The current state-of-the-art, PG-TD, combines a Transformer with MCTS in a simplistic manner that does not account adequately in decision making for the policy's (lack of) certainty. In so doing, it prioritizes exploitation over exploration. A second deficiency of PG-TD is that it can apply only a single reward objective. In this work, we introduce ERP, an entropy-reinforced planning approach for the Transformer decoder that addresses the limitations of PG-TD and other previous efforts by enhancing the exploration and exploitation tradeoff and permitting multiple objective improvement in the generation process.

4. The ERP Algorithm

Our Entropy-Reinforced Planning for Transformer Decoding (ECP) algorithm enhances Transformer decoding with an *e*-steps entropy-reinforced MCTS planner. ECT enhances sample efficiency by using the Transformer's Beam-Search, TOP-P, and TOP-K functions to narrow the molecule search space. It also leverages the controlled generation capabilities of MCTS and improves exploration and exploitation through the use of *e*-step forward entropy measurement.

We present the pseudocode of our algorithm in Algorithm 1 and detail the entire process in the following. At each step of decoding, we allocate a fixed number of rollouts to construct a tree representing potential future trajectories. Each rollout involves four key steps:

Selection. We employ a new selection algorithm, P$\mathcal{H}$-UCT, to decide which node to select. In this algorithm we enhance the bonus term to incorporate the probability of the next token as determined by the Transformer and its subsequent *e*-step entropy. Thus, the tree search tends to select tokens that balance exploitation and exploration while reducing global uncertainty. The process involves recursively choosing child

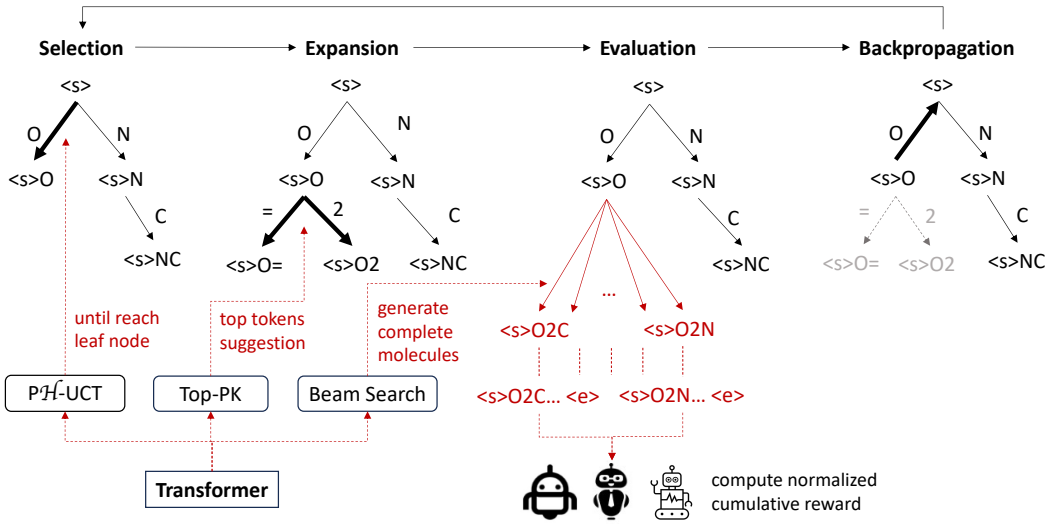


Figure 1: Illustration of the application of the ERP algorithm in the Transformer's process for generating molecules. Here, <s> denotes the start token [BOS], and <e> signifies the end token [EOS]. The parts highlighted in red are from Transformer.

nodes based on the P$\mathcal{H}$-UCT formula, starting at the root and continuing until an unexplored leaf node s is reached:

$$\text{P}\mathcal{H}\text{-UCT}\left(s_t, e\right) = \arg\max_{a \in \mathcal{A}} \text{P}\mathcal{H}\text{-UCB}\left(\left(s_t, a\right), e\right), \quad (8)$$

$$\text{where P}\mathcal{H}\text{-UCB}\left(\left(s_t, a\right), e\right) = Q\left(s_t, a\right) \quad (9)$$
$$+ c_p \frac{\sqrt{\log N\left(s_t\right)}}{1 + N\left(s_t, a\right)} \cdot \underbrace{\pi_\tau\left(a|s_t\right) \frac{1}{e} \sum_{i=1}^{e} \mathcal{H}\left(\pi_\tau\left(\cdot|s_{t+i}\right)\right)}_{\text{Entropy-Reinforced Planning}},$$

c_p is a tunable constant, τ is a temperature parameter applied to the policy $\pi_\tau\left(a|s\right) = \frac{\pi(a|s)^{\frac{1}{\tau}}}{\sum_b \pi(b|s)^{\frac{1}{\tau}}}$, and $\frac{1}{e}\sum_{i=1}^{e} \mathcal{H}\left(\pi_\tau\left(\cdot|s_{t+i}\right)\right)$ is the e-steps averaged entropy defined as follows:

$$\frac{1}{e}\sum_{i=1}^{e} \mathcal{H}\left(\pi_\tau\left(\cdot|s_{t+i}\right)\right) = \frac{1}{e}\sum_{i=1}^{e}\sum_{a \in \mathcal{V}^\star} \pi_\tau\left(a|s_{t+i}\right) \cdot \log\left(\pi_\tau\left(a|s_{t+i}\right)\right), \quad (10)$$

where $\mathcal{V}^\star$ represents the candidate token space conditioned on s_{t+i}. Intuitively, the function P$\mathcal{H}$-UCT$\left(s_t, e\right)$ is more likely to select an action a if 1) $Q\left(s_t, a_t\right)$ is high, indicating the discovery of a high reward molecule with prefix s_{t+1}; or 2) $\pi_\tau\left(a|s_t\right)$ is high, suggesting that the Transformer predicts a as a highly probable next token; or 3) $N\left(s_t\right)$ is large while $N\left(s_t, a\right)$ is small, implying that s_{t+1} has not been adequately explored. Additionally, 4) the selection is favored when the entropy of the subsequential tree is large in expectation, suggesting high uncertainty in its subsequence after performing action a.

Expansion. When P$\mathcal{H}$-UCT selects a node s_{t+1} which has no leaf nodes, we need to identify potential next tokens and incorporate the corresponding next states as new child nodes. Unlike standard MCTS, which may randomly sample a token, potentially leading to an invalid molecule, we employ a mixture of the Top-P and Top-K functions:

$$\text{Top-PK}\left(\mathbf{y}_{<i}, p, k\right) = \mathcal{A}_{\mathbf{y}_{<i}}, \quad (11)$$
$$\text{where } \mathcal{A}_{\mathbf{y}_{<i}} = \left\{y_1, \ldots, y_i, \ldots, y_j\right\}, y_i \in \mathcal{V},$$
$$j = \min\left\{\arg\min_{j'} \sum_{l=1}^{j'} \pi_\theta\left(y_l|\mathbf{y}_{<i}\right) \geq p, k\right\},$$
$$\text{and } \pi_\theta\left(y_g|\mathbf{y}_{<i}\right) > \pi_\theta\left(y_h|\mathbf{y}_{<i}\right), \forall g < h.$$

For a given state s_{t+1}, Top-PK $\left(s_{t+1}, p, k\right)$ (11) of the pretrained Transformer retrieves the at most k probable subsequent tokens $\mathcal{A}_{\mathbf{y}_{<i}}$ with a maximum accumulated probability p based on prior experience, where k is the maximum number of children. The at most k next states, formed by appending each of these suggested tokens from the Transformer to the current state, are then added to the children list of the current node. After the tree is thus expanded, we then need to evaluate the selected node s_t.

Evaluation. The selected node s_t might still be an incomplete molecule, but our reward function only calculates the reward for a complete molecule. To evaluate a partial molecule, we employ the Beam-Search $\left(s_t, b\right)$ (3) function from Transformer to generate complete molecules s_H with a prefix of partial molecule s_t and beam size b. Generated molecules are evaluated with the reward function, and the result is used as the value of node s_t.

Backpropagate. The reward of the completed molecule,

denoted r_H, is recursively propagated up the tree from the initially selected node until it reaches the root. During this tree traversal, each value $Q(s_i, a_i)|_{0 \le i < t}$ encountered is updated with the newly obtained value r_H, via the formula

$$Q(s_i, a_i) \leftarrow \max\{Q(s_i, a_i), r_H\}, \tag{12}$$

where $0 \le i < t$, and $s_i \circ a_i = s_{i+1}$.

Illustration of how *e*-step entropy makes the difference. Fig 2 assumes an environment with the initial state (partial molecule) s_0 = <s> and an action space defined as $\mathcal{A} = \{\text{left}, \text{right}\}$. Given a pretrained Transfomer decoder π_θ, if we consider only the initial s_0, then π_θ presents equal probability for both actions: $\pi_\theta(\text{left}|\mathbf{x}) = \pi_\theta(\text{right}|\mathbf{x})$, where $\mathbf{x}$ = <s>. However, if we look a few steps ahead, the subsequent tree under the right action presents much higher uncertainty that does that under the left action, indicating that the Transformer has less experience on the right-hand branch. Thus, in this scenario, we need to reduce the uncertainty of π_θ via greater exploration of the right tree, if we are not to miss the hidden high-reward state.

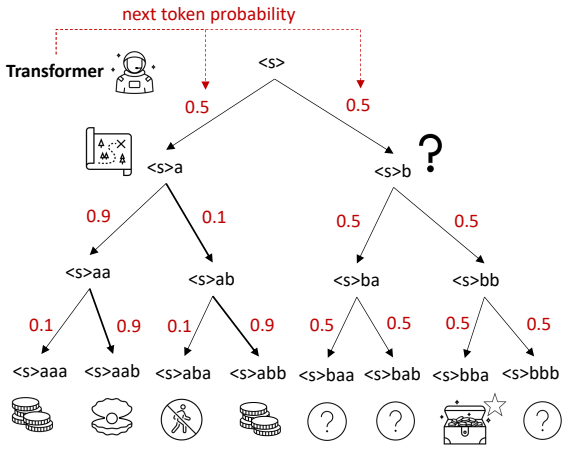


Figure 2: An environment with action space $\mathcal{A} = \text{left}, \text{right}$, in which each node (state) is connected by only two edges (actions), and each edge is associated with a probability of being sampled, as determined by the pretrained Transformer decoder π_θ. The red values are inferred by the Transformer.

Multi-critic normalized reward. We have an ensemble array of critics as follows:

$$\mathbf{C} = \left[C^{\text{Druglikeness}}, C^{\text{Solubility}}, C^{\text{Synthesizability}}, C^{\text{Docking}}\right],$$

where each critic $C : Y_{1:H} \to \mathbb{R}$ represents a unique quality evaluator of a pharmaceutical property. Here we reference the multiple critic from Liu et al. (2023b), and designed a normalized reward function to align the drug optimization with multiple objectives as follows,

$$R^{\text{sum}}_{\text{norm}}(s_H) = \sum_{i=0}^{|\mathbf{C}|-1} \text{Norm}\left(C_i(s_H)\right), \tag{13}$$

where $C : Y_{1:H} \to \mathbb{R}$, and $C_i \in \mathbf{C}$. We use Norm to normalize different attributes onto the same scale. Here, we define Norm as min-max normalization to scale the attributes onto the range [0, 1].

Algorithm 1 Entropy-Reinforced Planning (ERP)

Require: $root$: s_0, c_{p}: exploration parameter
N: rollout number
p: maximum cumulative probability
k: maximum number of node's children
b: beam number
e: forward entropy steps
$cache$ = DICTIONARY ().
for $i \leftarrow 1, 2, ..., N$ **do**
  $node \leftarrow root$.
  ▷ /* Selection */
  while $|node.children| > 0$ **do**
    $node \leftarrow$ P$\mathcal{H}$-UCT $(node.child, e)$ in Eq. (8)
  ▷ /* Expansion */
  $[next\_tokens] \leftarrow$ TOP-PK $(node, p, k)$ Eq. (11)
  for $next\_token \in [next\_tokens]$ **do**
    $next\_state \leftarrow$ CONCAT $(node, next\_token)$
    Create a node new_node for $next\_state$
    Add new_node to the children of $node$
  ▷ /* Evaluation */
  $\{mols\} \leftarrow$ Beam-Search $(node, b)$
  for $mol \in \{mols\}$ and $mol \notin cache$.keys() **do**
    $r \leftarrow$ GET-REWARD (mol) in (13)
    $cache\,[mol] = r$
  $r_H = \max\{cache[mol]\}, mol \in \{mols\}$
  ▷ /* Backpropagation */
  Update the values of $node$ with r_H
  Update its ancestors with r_H by Eq. (12)
return Top reward molecules in $cache$

5. Experiments

We describe our experimental setup and present results.

5.1. Drug discovery

5.1.1. Experimental configuration

The language model. We train three GPT-2-like Transformers on the task of causal language modeling, as follows:

The *pretrained model* is a 124M GPT2 model trained using a BPE tokenizer (Bostrom and Durrett, 2020) on a diverse dataset of 10.7 million SMILES strings of drug-like molecules randomly sampled from the ZINC database (Irwin and Shoichet, 2005).

The *biased model* is fine-tuned based on the pretrained model with a distinct objective focused solely on the dock-

Target	Algorithm	Best Norm Reward ↑	Avg Valid Norm Reward ↑	Avg Top 10% Norm Reward ↑	Unique Valid Molecule ↑	Docking ↓	Druglikeliness ↑	Synthesizability ↓	Solubility ↑
3CLPro	**Beam Search**	2.53 (-10.99%)	2.38 (-0.75%)	2.53 (-7.40%)	16 (-98.63%)	-8.67 (6.30%)	0.86 (26.91%)	3.35 (-59.18%)	3.01 (-31.87%)
(PDBID:	**Sampling**	2.85 (0%)	2.39 (0%)	2.74 (0%)	1609 (0%)	-8.15 (0%)	0.68 (0%)	2.11 (0%)	4.42 (0%)
7BQY)	**UCT**	3.10 (8.86%)	2.25 (-5.83%)	2.65 (-3.30%)	**3013 (157.96%)**	-8.45 (3.67%)	**0.89 (31.34%)**	2.02 (4.17%)	4.39 (-0.84%)
	PG-TD	3.20 (12.50%)	2.59 (8.05%)	2.99 (9.38%)	2549 (118.24%)	-8.37 (2.70%)	0.88 (30.30%)	1.79 (15.15%)	**4.79 (8.37%)**
	ERP (Ours)	**3.24 (13.71%)**	**2.62 (9.42%)**	**3.07 (12.25%)**	2575 (120.46%)	**-9.13 (11.98%)**	**0.89 (31.00%)**	**1.71 (18.78%)**	4.44 (0.50%)
RTCB	**Beam Search**	2.56 (-17.46%)	2.39 (-3.35%)	2.39 (-17.02%)	8 (-99.32%)	-7.11 (-30.85%)	0.82 (-3.55%)	3.02 (-55.02%)	3.55 (1.68%)
(PDBID:	**Sampling**	3.10 (0%)	2.47 (0%)	2.88 (0%)	1168 (0%)	**-10.28 (0%)**	0.85 (0%)	1.95 (0%)	3.49 (0%)
4DWQ)	**UCT**	2.97 (-4.12%)	2.28 (-7.61%)	2.73 (-5.36%)	**1491 (27.65%)**	-8.62 (-16.22%)	0.83 (-2.77%)	1.89 (3.18%)	3.41 (-2.46%)
	PG-TD	3.10 (0.05%)	2.53 (2.26%)	2.93 (1.63%)	1266 (8.39%)	-9.22 (-10.30%)	0.84 (-1.82%)	1.72 (11.88%)	3.67 (5.03%)
	ERP (Ours)	**3.24 (4.52%)**	**2.61 (5.61%)**	**3.07 (6.68%)**	1343 (14.98%)	-8.81 (-14.34%)	**0.93 (9.40%)**	**1.56 (20.01%)**	**3.87 (10.92%)**

Table 1: **Main results.** A comparison of four baselines {Beam Search, Sampling, UCT, PG-TD} and ERP on multiple objectives based on 3CLPro and RTCB datasets and pretrained Transformer model. Boldface denotes best. The percentage of improvement over the Sampling baseline is enclosed in parentheses. All experiments are conducted with 256 rollouts.

ing score optimization, using a biased dataset that only contains molecules with docking scores in the range [-6, -14]. Docking scores are numerical values generated by computational docking simulations, predicting how well a small molecule, such as a potential drug, fits into the binding site of a target protein. A higher score represents better strength and stability of the interaction.

The *RL fine-tuned model* is fine-tuned based on a pretrained model, using the same objective as ERP, with the multi-critic normalized reward function (13). We fine-tune the pretrained model by using an RL approach (Liu et al., 2023b) focusing on the top 10 sampled molecules optimization.

Baselines. We compare ERP with four baselines: 1) *Beam search*; 2) *Sampling*, in which the Transformer's top-k sampling algorithm is employed to sample a token from the top-k most likely tokens at each step to generate a set of molecules; 3) *UCT*, which adopts the UCB algorithm for node selection; and 4) *PG-TD*, the current state-of-the-art algorithm for integrating MCTS and Transformer by applying P-UCT (6). Both UCT and PG-TD are tailored to use the reward function (13).

Dataset. We employ, from the most recent Cancer and COVID dataset of Liu et al. (2023b), 1 million compounds from the ZINC15 dataset docked to the 3CLPro (PDB ID: 7BQY) protein associated with SARS-CoV-2 and the RTCB (PDB ID: 4DWQ) human cancer protein.

Critics and evaluation metric. We evaluate seven key attributes for pharmaceutical drug discovery: 1) *Best normalized reward* is the molecule with the top normalized reward; 2) *Average valid normalized reward* is the average normalized reward of all valid molecules; 3) *Average top 10% normalized reward* is the average normalized reward of top 10% valid molecule. 4) *Unique valid samples*; 5) *Druglikeness* assesses the probability of a molecule being a suitable drug candidate; 6) *Solubility* evaluates a molecule's water-octanol partition coefficient (LogP), indicating how well it can dissolve in water; 7) *Synthesizability* measures the synthesizability of a molecule, assigning a score of 1 for easy synthesis and a score of 10 for difficult synthesis (Ertl and Schuffenhauer, 2009); and 8) *Docking score* (generated, for efficient calculation, with a surrogate docking model: see Appendix A.1) evaluates the potential of a drug to inhibit the target site.

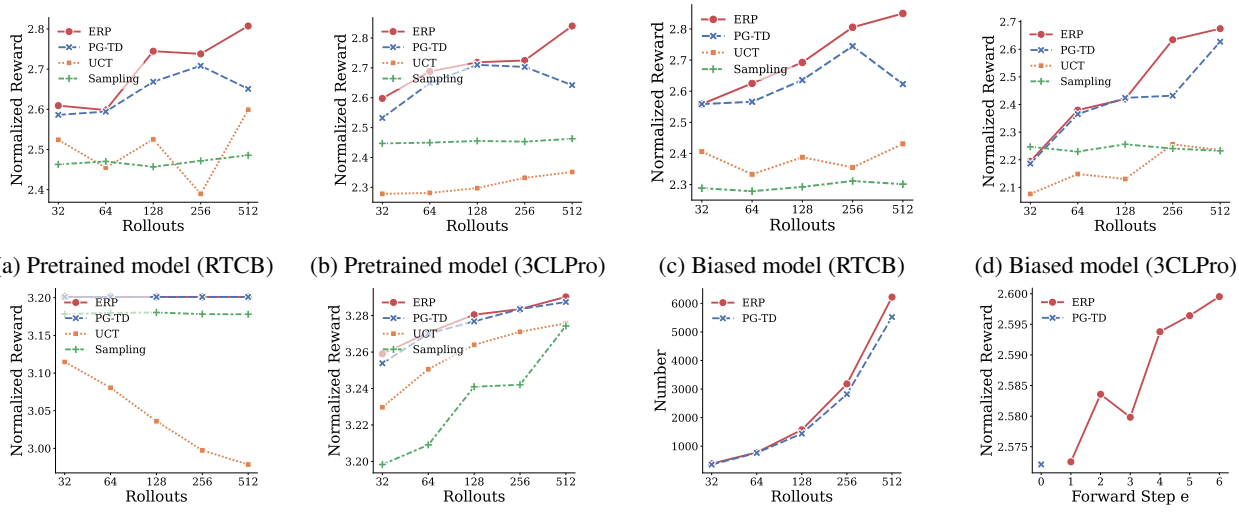
5.1.2. Results

The ERP (red) and baseline results in Table 1 and Fig 3 show that ERP significantly outperforms all baselines, including the current state-of-the-art, PG-TD (in blue), across various performance metrics. (For even more baselines of simpler models, see appendix A.5.3.) Notably, ERP enhances the performance of different types of pretrained Transformers (also used as the sampling baseline in green) as follows:

Pretrained LLM serves as a general model. Figs 3a and 3b show that the Transformer-based planning algorithms, ERP and PG-TD, outperform the non-Transformer UCT significantly. This result suggests that the prior experience of the pretrained Transformer benefits both ERP and PG-TD by improving sampling efficiency. We note also that ERP surpasses PG-TD in both benchmarks.

Biased LLMs, trained on objectives different from those in this work, result in the Transformer Sampling baseline performing significantly worse than the pure MCTS UCT baseline. This result indicates that the biased prior experience of the Transformer guides molecule exploration into low-reward regions. However, Figs 3c and 3d illustrate that ERP still outperforms the others in this situation, demonstrating the effectiveness of its controllable generation process. By adopting ERP's planning guidance, the generation process optimizes from the prior objective to the current one, showcasing its adaptability.

RL fine-tuned LLMs present a challenging benchmark, involving optimization based on the top 10 sampled



(a) Pretrained model (RTCB) (b) Pretrained model (3CLPro) (c) Biased model (RTCB) (d) Biased model (3CLPro)

(e) RL fine-tuned mod. (3CLPro) (f) Top 10 leading compounds (g) Num. unique valid molecules (h) ERP Step e (3CLPro)

Figure 3: **Ablation studies.** (a)(b)(c)(d)(e) Normalized rewards averaged among valid molecules for different LLMs and algorithms. ERP is our model. PG-TD (Zhang et al., 2023a) is the previous state-of-the-art method as described by Eq. (6), and UCT by Eq (5). We also do random sampling from the LM as the Sampling baseline. (f) Filtered the top 10 leading compounds from the molecules discovered in (e). (g) ERP vs. PG-TD for number of unique valid molecules in 3CLPro dataset across different rollouts. (h) Effects of entropy step e of ERP.

molecules until performance plateaus. Figs 3e and 3f demonstrate that ERP still enhances the capabilities of these optimized models. By implementing the entropy-reinforced planner, it continues to refine its performance through active exploration and exploitation of promising regions. Since the Transformer model is fine-tuned based solely on the top 10 samples, Fig 3e shows similar performance between ERP and PG-TD in terms of average valid molecules. Yet Fig 3f reveals that ERP surpasses PG-TD in identifying the top 10 leading compounds.

Effectiveness of ERP. Table 1 and Fig 3 demonstrate the effectiveness of ERP. In Figs 3a, 3b, and 3c, PG-TD experienced a non-trivial performance drop when the number of rollouts increased from 256 to 512. This indicates that PG-TD began to discover mostly low-reward molecules, significantly lower than those discovered previously, thereby reducing the overall performance compared to that in the 256-rollout scenario. However, ERP enjoyed a monotonic performance improvement, finding molecules with even higher normalized values compared to previous rollouts. Moreover, Fig 3g illustrates that ERP consistently discovered more unique samples than PG-TD. This metric assesses the algorithm's diversity and exploratory capability. These results indicates that PG-TD fails to conduct effective exploration as the number of rollouts grows, while ERP was capable of balancing exploration and exploitation, discovering higher-reward molecules as rollouts increase. For more ablations, see Appendix A.5.4.

Effect of e-step forward in entropy measurement. We see in Fig 3h that performance improves with the number of forward steps for entropy-reinforced measurement. This improvement occurs because deeper analysis of the subtree enables better estimation of certainty for balancing between exploitation and exploration.

5.2. Code generation

5.2.1. Experimental configuration

The language model. Due to limited time and computational resources, we confined both the PG-TD and ERP methods to a rollout of 64, and further configured the ERP with an e-step of 2. We reused the GPT-x models from the PG-TD release versions.

Dataset. We conducted experiments on three benchmarks in total: APPS Intro, APPS Inter, APPS Comp from APPS (Hendrycks et al., 2021).

Evaluation metric. Performance was assessed using two metrics—pass rate and strict accuracy—and two tasks: n@k and pass@k (Li et al., 2022). n@k measures the success rate of the top n selected programs in passing all private test cases. pass@k assesses the overall success rate of any of the k generated programs in passing all private test cases. For both ERP and PG-TD, the k samples consist of the full programs obtained from the initial k rollouts.

The pass rate represents the average percentage of private

n@k	Base Model	Algorithm	PR (%) Intro.	PR (%) Inter.	PR (%) comp.	SA (%) Intro.	SA (%) Inter.	SA (%) comp.
1@15	**GPT-2**	**PG-TD**	23.242	**12.481**	**19.055**	2.000	**3.433**	**8.600**
		ERP (Ours)	**23.955**	12.248	18.532	2.000	3.267	7.900
1@15	**GPT-Neo**	**PG-TD**	**24.399**	12.301	20.333	**2.200**	3.433	9.200
		ERP (Ours)	24.384	**12.490**	**21.117**	2.100	**3.467**	**10.100**
1@20	**GPT-2**	**PG-TD**	23.540	12.642	18.808	2.200	3.333	8.400
		ERP (Ours)	**25.204**	**13.277**	**20.174**	**2.300**	**3.467**	**9.000**
1@20	**GPT-Neo**	**PG-TD**	24.525	12.392	20.541	2.200	3.333	9.400
		ERP (Ours)	**26.034**	**13.978**	**21.342**	**2.400**	**3.933**	**10.200**
pass@15	**GPT-2**	**PG-TD**	27.873	23.226	32.732	2.200	8.367	**16.600**
		ERP (Ours)	**28.421**	**23.477**	**32.861**	**2.300**	**8.567**	15.900
pass@15	**GPT-Neo**	**PG-TD**	28.928	24.267	35.565	2.400	**9.067**	19.000
		ERP (Ours)	**29.302**	**24.417**	**35.896**	**2.500**	8.900	**19.700**
pass@20	**GPT-2**	**PG-TD**	28.142	23.977	33.420	2.300	8.700	16.800
		ERP (Ours)	**29.711**	**24.974**	**34.806**	**2.400**	**9.200**	**17.000**
pass@20	**GPT-Neo**	**PG-TD**	29.209	24.992	36.054	2.400	9.233	19.300
		ERP (Ours)	**30.916**	**26.004**	**37.660**	**2.700**	**9.433**	**20.400**

Table 2: ERP was compared to PG-TD for n@k and pass@k tasks (k=15, 20) on three APPS benchmarks (Introductory, Interview, Competition) using GPT-2 and GPT-Neo models. Evaluated by Pass Rate (PR) and Strict Accuracy (SA), ERP typically surpassed PG-TD.

	1@10	1@15	1@20
PG-TD	21.438	23.242	23.540
ERP (Ours)	**22.101**	**23.955**	**25.204**
	pass@10	pass@15	pass@20
PG-TD	25.683	27.873	28.142
ERP (Ours)	**25.958**	**28.421**	**29.711**

Table 3: The pass rate (%) of 1@k and pass@k of ERP ($e = 2$) and PG-TD, evaluated on the APPS introductory problems.

	pass@10	pass@15	pass@20
PG-TD	25.683	27.873	28.142
ERP ($e = 2$)	25.958	28.421	29.711
ERP ($e = 3$)	**26.073**	28.712	**29.991**
ERP ($e = 4$)	25.840	**28.745**	29.990

Table 4: The pass rate (%) of pass@k of ERP of varying forward steps $e \in \{2, 3, 4\}$ and PG-TD on the APPS introductory problems.

test cases that the generated programs successfully pass across all problems. Strict accuracy measures the percentage of problems for which the generated programs pass all private test cases. We report n@k and pass@k results with n and k values of 15 and 20, respectively.

5.2.2. RESULTS

On APPS task. As shown in Table 2, our approach outperformed the current leading method, PG-TD, in all three code generation tasks across two LLM models, GPT-2 and GPT-Neo, on two metrics: pass rate and strict accuracy. We report 1@k and pass@k with k being 15 and 20. Our results demonstrate that the ERP method overall surpasses the PG-TD method. In Table 3, we report more experimental results on APPS introductory problems with k from 10, 15, and 20, where ERP still performs better than the baseline.

Effect of e-step forward. The hyperparameter e in the forward step entropy reinforcement also has an effect on the performance on the task of code generation. As can be seen from Table 4, greater forward steps (3 or 4) lead to higher pass rates, which shows the importance of multi-step forward looking for our ERP algorithm.

6. CONCLUSION

We introduce a novel algorithm, ERP, Entropy-Reinforced Planning for Transformer decoding. ERP employs an *e*-step forward entropy-based MCTS planner to guide a Transformer decoder. Within the ERP framework, we incorporate the P$\mathcal{H}$-UCT algorithm for the selection phase and TOP-PK for the expansion phase of tree search planning. The resulting system: 1) enhances sample efficiency by leveraging Transformer sampling to draw upon prior knowledge in the expansive molecule search space; 2) achieves controllable generation by adapting Transformer, which is trained on various objectives, and optimizes its decoding process to meet our specific goals; and 3) reduces uncertainty and balances exploitation and exploration through entropy-reinforced planning, enhancing discovery of high-reward molecules in uncertain areas of molecular spaces. Empirical evaluations across a range of tasks demonstrate that ERP consistently surpasses the current state-of-the-art PG-TD and competing baselines. Our work highlights ERP's effectiveness in the domain of drug discovery and code generation, showing improvements in several pharmaceutical properties, as well as pass rates and strict accuracy for code generation. We encourage the application of ERP in domains beyond the scope of our current research.

Acknowledgements

We thank Ian T. Foster for polishing the draft and Alex Brace for providing feedback on it. We also thank Martin L. Putra for the initial discussion. This work is supported in part by the RadBio-AI project (DE-AC02-06CH11357), U.S. Department of Energy Office of Science, Office of Biological and Environment Research, the Improve project under contract (75N91019F00134, 75N91019D00024, 89233218CNA000001, DE-AC02-06-CH11357, DE-AC52-07NA27344, DE-AC05-00OR22725), the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Impact Statement

This paper presents work whose goal is to advance the field of machine learning for generation of structural sequences. A generative AI model for structural data has a few potential societal consequences, none of which we feel must be specifically highlighted here.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Sara Romeo Atance, Juan Viguera Diez, Ola Engkvist, Simon Olsson, and Rocío Mercado. De novo drug design using reinforcement learning with graph-based deep generative models. *Journal of Chemical Information and Modeling*, 62(20):4863–4872, 2022.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

Viraj Bagal, Rishal Aggarwal, PK Vinod, and U Deva Priyakumar. MolGPT: Molecular generation using a transformer-decoder model. *Journal of Chemical Information and Modeling*, 62(9):2064–2076, 2021.

Jannis Born, Matteo Manica, Ali Oskooei, Joris Cadow, Greta Markert, and María Rodríguez Martínez. PaccMannRL: De novo generation of hit-like anticancer molecules from transcriptomic data via reinforcement learning. *Iscience*, 24(4), 2021.

Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901, 2020.

Antoine Chaffin, Vincent Claveau, and Ewa Kijak. PPL-MCTS: Constrained textual generation through discriminator-guided MCTS decoding. *arXiv preprint arXiv:2109.13582*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1:1–11, 2009.

Argonne Leadership Computing Facility. https://www.alcf.anl.gov/polaris, last accessed on 10-2-2023.

Nathan C Frey, Ryan Soklaski, Simon Axelrod, Siddharth Samsi, Rafael Gomez-Bombarelli, Connor W Coley, and Vijay Gadepally. Neural scaling of deep chemical models. *Nature Machine Intelligence*, pages 1–9, 2023.

Sai Krishna Gottipati, Boris Sattarov, Sufeng Niu, Yashaswi Pathak, Haoran Wei, Shengchao Liu, Simon Blackburn, Karam Thomas, Connor Coley, Jian Tang, Sarath Chandar, and Yoshua Bengio. Learning to navigate the synthetically accessible chemical space using reinforcement learning. In *International Conference on Machine Learning*, pages 3668–3679. PMLR, 2020.

Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

Ikbel Hadj Hassine. Covid-19 vaccines and variants of concern: A review. *Reviews in Medical Virology*, 32(4): e2313, 2022.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir

Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

Siqi Hong, Hankz Hankui Zhuo, Kebing Jin, Guang Shao, and Zhanwen Zhou. Retrosynthetic planning with experience-guided Monte Carlo tree search. *Communications Chemistry*, 6(1):120, 2023.

John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1): 177–182, 2005.

John J Irwin, Khanh G Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R Wong, Munkhzul Khurelbaatar, Yurii S Moroz, John Mayfield, and Roger A Sayle. Zinc20—a free ultralarge-scale chemical database for ligand discovery. *Journal of chemical information and modeling*, 60(12):6065–6073, 2020.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*, pages 4849–4859. PMLR, 2020.

Brian P Kelley, Scott P Brown, Gregory L Warren, and Steven W Muchmore. POSIT: Flexible shape-guided docking for pose prediction. *Journal of Chemical Information and Modeling*, 55(8):1771–1780, 2015.

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer, 2006.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Rémi Leblond, Jean-Baptiste Alayrac, Laurent Sifre, Miruna Pislar, Jean-Baptiste Lespiau, Ioannis Antonoglou, Karen Simonyan, and Oriol Vinyals. Machine translation decoding beyond beam search. *arXiv preprint arXiv:2104.05336*, 2021.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. ChatGPT-powered conversational drug editing using retrieval and domain feedback. *arXiv preprint arXiv:2305.18090*, 2023a.

Xuefeng Liu, Jiang Songhao, Archit Vasan, Alex Brace, Ozan Gokdemir, Tom Brettin, Fangfang Xia, Ian Foster, and Rick Stevens. DRUGIMPROVER: Utilizing reinforcement learning for multi-objective alignment in drug optimization. In *NeurIPS Workshop on New Frontiers of AI for Drug Discovery and Development*, 2023b.

Xuefeng Liu, Takuma Yoneda, Rick L Stevens, Matthew R Walter, and Yuxin Chen. Blending imitation and reinforcement learning for robust policy improvement. *arXiv preprint arXiv:2310.01737*, 2023c.

Xuefeng Liu, Takuma Yoneda, Chaoqi Wang, Matthew R Walter, and Yuxin Chen. Active policy improvement from multiple black-box oracles. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 22320–22337, 2023d.

Thaer M. Dieb, Shenghong Ju, Kazuki Yoshizoe, Zhufeng Hou, Junichiro Shiomi, and Koji Tsuda. MDTS: Automatic complex materials design using Monte Carlo tree search. *Science and Technology of Advanced Materials*, 18(1):498–503, 2017.

Clara Meister, Tim Vieira, and Ryan Cotterell. If beam search is the answer, what was the question? In *Conference on Empirical Methods in Natural Language Processing*, page 2173–2185. Association for Computational Linguistics, 2020.

Kusuri Murakumo, Naruki Yoshikawa, Kentaro Rikimaru, Shogo Nakamura, Kairi Furui, Takamasa Suzuki, Hiroyuki Yamasaki, Yuki Nishigaya, Yuzo Takagi, and Masahito Ohue. LLM drug discovery challenge: A contest as a feasibility study on the utilization of large language models in medicinal chemistry. In *AI for Accelerated Materials Design-NeurIPS 2023 Workshop*, 2023.

Daniel Neil, Marwin Segler, Laura Guasch, Mohamed Ahmed, Dean Plumbley, Matthew Sellwood, and Nathan Brown. Exploring deep recurrent models with reinforcement learning for molecule design. In *ICLR*, 2018.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9 (1):1–14, 2017.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7):eaap7885, 2018.

Martin L Puterman. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Stephane Ross and J Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*, 2014.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *14th International Conference on Artificial Intelligence and Statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Daniel Rothchild, Alex Tamkin, Julie Yu, Ujval Misra, and Joseph Gonzalez. C5T5: Controllable generation of organic molecules with transformers. *arXiv preprint arXiv:2108.10307*, 2021.

Thomas Scialom, Paul-Alexis Dray, Jacopo Staiano, Sylvain Lamprier, and Benjamin Piwowarski. To beam or not to beam: That is a question of cooperation for language GANs. *Advances in Neural Information Processing Systems*, 34:26585–26597, 2021.

Philipp Seidl, Andreu Vall, Sepp Hochreiter, and Günter Klambauer. Enhancing activity prediction models in drug discovery with the ability to understand human language. *arXiv preprint arXiv:2303.03363*, 2023.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Niclas Ståhl, Goran Falkman, Alexander Karlsson, Gunnar Mathiason, and Jonas Bostrom. Deep reinforcement learning for multiparameter optimization in de novo drug design. *Journal of chemical information and modeling*, 59(7):3166–3176, 2019.

Youhai Tan, Lingxue Dai, Weifeng Huang, Yinfeng Guo, Shuangjia Zheng, Jinping Lei, Hongming Chen, and Yuedong Yang. DRlinker: Deep reinforcement learning for optimization in fragment linking design. *Journal of Chemical Information and Modeling*, 62(23):5907–5917, 2022.

Ruixiang Tang, Yu-Neng Chuang, and Xia Hu. The science of detecting LLM-generated texts. *arXiv preprint arXiv:2303.07205*, 2023.

Archit Vasan, Rick Stevens, Arvind Ramanathan, and Vishwanath Venkatram. Benchmarking language-based docking models. 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. ChatGPT for robotics: Design principles and model abilities. *arXiv:2306.17582*, 2023.

Chenran Wang, Yang Chen, Yuan Zhang, Keqiao Li, Menghan Lin, Feng Pan, Wei Wu, and Jinfeng Zhang. A reinforcement learning approach for protein-ligand binding pose prediction. *BMC Bioinformatics*, 23(1):1–18, 2022a.

Wenlu Wang, Ye Wang, Honggang Zhao, and Simone Sciabola. A transformer-based generative model for de novo molecular design. *arXiv preprint arXiv:2210.08749*, 2022b.

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.

Shenghao Wu, Tianyi Liu, Zhirui Wang, Wen Yan, and Yingxiang Yang. RLCG: When reinforcement learning meets coarse graining. In *NeurIPS 2022 AI for Science: Progress and Promises*, 2022.

Chao-Han Huck Yang, Linda Liu, Ankur Gandhe, Yile Gu, Anirudh Raju, Denis Filimonov, and Ivan Bulyko. Multitask language modeling for improving speech recognition of rare words. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1087–1093. IEEE, 2021.

Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. ChemTS: An efficient Python library for de novo molecular generation. *Science and Technology of Advanced Materials*, 18(1):972–976, 2017.

Xiufeng Yang, Tanuj Kr Aasawat, and Kazuki Yoshizoe. Practical massively parallel Monte-Carlo tree search applied to molecular design. *arXiv preprint arXiv:2006.10504*, 2020.

Tatsuya Yoshizawa, Shoichi Ishida, Tomohiro Sato, Masateru Ohta, Teruki Honma, and Kei Terayama. Selective inhibitor design for kinase homologs using multiobjective Monte Carlo tree search. *Journal of Chemical Information and Modeling*, 62(22):5351–5360, 2022.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for

goal-directed molecular graph generation. *Advances in Neural Information Processing Systems*, 31, 2018.

Koichi Yuki, Miho Fujiogi, and Sophia Koutsogiannaki. Covid-19 pathophysiology: A review. *Clinical Immunology*, 215:108427, 2020.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*, 2023a.

Yunjiang Zhang, Shuyuan Li, Miaojuan Xing, Qing Yuan, Hong He, and Shaorui Sun. Universal approach to de novo drug design for target proteins using deep reinforcement learning. *ACS Omega*, 8(6):5464–5474, 2023b.

Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9(1):10752, 2019.

A. Appendix

A.1. Surrogate model

We utilize a BERT-like transformer surrogate model for calculating docking scores, as described by Vasan et al. (2023) and Liu et al. (2023b). Tokenized SMILES strings are input to the model and positionally embedded. The outputs are then processed through a series of five transformer blocks, each comprising a multi-head attention layer with 21 heads, a dropout layer, layer normalization with a residual connection, and a feed-forward network. The feed-forward network includes two dense layers, followed by dropout and layer normalization with a residual connection. Finally, the predicted docking score is output by the feed-forward network after the series of transformer blocks.

A.2. Binding sites of RTCB and 3clpro

(a) RTCB. (b) 3CLPro.

Figure 4: The binding sites of RTCB (PDB ID: 4DWQ) and proteins 3CLPro (PDB ID: 7BQY). The Open Eye software is used to define binding sites surrounding the crystallized compound (Kelley et al., 2015; Liu et al., 2023b).

A.3. Dataset detail

Pretrained LLM is trained using about 10.7 million druglike and in-stock molecules from ZINC20 (Irwin et al., 2020) with standard reactivity. These molecules have a minimum sequence length of 8, an average length of 46, and a maximum length of 252. The biased LLMs are finetuned from the pretrained LLM using the cancer and covid dataset of Liu et al. (2023b). More specifically, 1 million compounds docked to 3CLPro (PDB ID: 7BQY) protein associated with SARS-CoV-2 and the RTCB (PDB ID: 4DWQ) human cancer protein dataset with docking score in range [-14, -9], indicating string interactions, are selected for finetuning. When assessing the performance of our model, we go beyond solely checking if it can generate valid SMILES strings. Unlike code generation tasks, which are typically evaluated on a simple pass/fail basis depending on whether the code runs correctly, our evaluation for SMILES generation encompasses several additional pharmaceutical objectives: druglikeliness, docking score, synthesizability and solubility. This comprehensive approach ensures not only the generation of chemically valid structures but also evaluates their potential as effective drug candidates based on multiple critical factors.

Dataset	Min Legnth	Mean Length	Max Length
ZINC	8	46.0	252
RTCB	12	42.7	119
3CLPro	18	49.6	107

Table 5: Minimum, Mean and Maximum length of SMILES in each dataset

A.4. Experiments on Code generation task

In the main paper, we focus on addressing the drug discovery challenge, however the contribution of ERP is primarily methodological, which allows its application across various domains, including code generation. In this section, we conduct additional two experiments on two code generation dataset, APPS (Hendrycks et al., 2021) and CodeContests (Li et al.,

2022). Here, we conducted an evaluation on base model GPT-2 and GPT-Neo using three additional code generation benchmark APPS dataset [1] including APPS Intro, APPS Inter, APPS comp and one CodeContests [2] datasets benchmark, and compared our method to the existing state-of-the-art (PG-TD) on both pass rate and strict accuracy metric. Our approach surpassed the current leading method PG-TD on all the 4 code generation tasks across two LLMs model GPT-2, GPT-Neo on two metric pass rate, Strict Accuracy as well. Lastly, we have made updates to both papers (see appendix ??) and the codebase to reflect these new experiment results.

A.5. Experiments on Molecules

A.5.1. Setup and hyperparameters

For equitable assessment, we compare different algorithms based on similar setups. In particular, in each comparison of different algorithms in all tables and charts, they share the key hyperparameters of the number of rollouts, and all generation and prediction configuration of the Transformer-based language models.

We did limited hyperparameter search based on the possible values in the range shown in Table 6. Then we use the same hyperparameter to compare different algorithms.

Hyperparameter	Experimented values
GPT2 Pretraining and Fine-tuning	
Learning rate	5×10^{-5}
Batch size per GPU	8
# of GPU used	8
# of epochs	10
GPT2 Fine-tuning	
# of epochs	40
Tree Search with ERP	
Number of rollouts N	$\{32, 64, 128, 256, 512\}$
Exploration parameter c_p	$\{1, 4, 8\}$
LM top-p filter for expansion p	$\{0.9, 0.95\}$
LM top-k filter for expansion k	$\{15, 20\}$
LM beam size for evaluation b	$\{8, 16\}$
Forward step e	$\{1, 2, 3, 4, 5, 6\}$

Table 6: Hyperparameters of possible values. We made limited search among the possible values and compare different algorithms based on the same applicable hyperparameters.

A.5.2. Computing infrastructure and wall-time comparison

We trained our docking surrogate models using four nodes of the supercomputer where each node contains one 64-core CPU and four A100 GPU nodes (Facility). The training time for each model was approximately three hours. We conducted the Monte-Carlo tree search with Transformer inference on a cluster that includes CPU nodes and GPU nodes two Nvidia GPUs. Based on the computing infrastructure, we obtained the wall-time comparison in Table 7 as follows.

Methods	Total Run Time
Pretrain GPT	9 hours
Biased GPT	17 hours
RL finetuned GPT	17 hours
Sampling	10 min.
PG-TD	30 min.
ERP	40 min.

Table 7: Wall-time comparison between different methods.

A.5.3. EXTENDED BASELINES

We present results with some more baselines in Table 8, which demonstrated the complexity of the task of molecule generation. Specifically, it provides the UCT-only baseline, which cannot generate any valid molecules in our experimentation. Furthermore, using LM in the expansion phase but not in the selection has minuscule improvement over random sampling baseline, and shows the importance of using the LM in the selection phase as well.

Model	Avg. norm. reward	Ratio of valid mol.	# unique valid mol. found
Uniform random sampling	Null	0.0	0
UCT without LM	Null	0.0	0
UCT using LM in expansion only	1.99	0.0001	1
Beam search with pre-trained LM (as in Table 1)	2.38	1.0	16
Sampling with pre-trained LM (as in Table 1)	2.39	1.0	1609
UCT (as in Table 1)	2.25	1.0	3013
PG-TD (as in Table 1)	2.59	1.0	2549
ERP (as in Table 1)	2.62	1.0	2575

Table 8: Results with some more baselines with the same setting as in Table 1, which demonstrated the complexity of the task of molecule generation.

A.5.4. MORE ABLATIONS

Token-budgeted comparisons In Table 9, we show the performance of ERP and PG-TD with upper-bounded budget of the number of generated tokens by the LM in the tasks of molecular generation. It shows that ERP has a stronger performance than PG-TD given a LM-generated token budget, given number of rollouts being 32.

Number of sampled tokens	Average normalized reward	
	ERP with e-etep=6	PG-TD
1024	2.36	2.36
2048	2.28	2.25
4096	2.34	2.28
8192	2.39	2.35
16384	2.54	2.43
32768	2.60	2.57
65536	2.60	2.57

Table 9: Average normalized rewards given number of rollouts being 32, and varying budget of sampled tokens.

Effects of Top-pk expansion In Table 10 we show the ablation of removing Top-pk filtering during the expansion phase. The ablation results between Row 1 and Row 4 show the importance of top-pk filtering in ERP, whereas the ablation results between Row 2 and Row 4 show the effects of ERP.

E-step (Selection)	Top-P-K filtering (Expansion)	Average reward	Note
6	False	2.46	
0	True	2.57	Reduced to P-UCB, As in Figure 3 (h)
1	True	2.57	As in Figure 3 (h)
6	True	2.60	As in Figure 3 (h)

Table 10: Average normalized rewards showing the effects of top-pk filtering during expansion.