

Deep Reinforcement Learning for Inverse Synthetic Polymer Design

by

M.P.C. van der Werf

to obtain the degree of Master of Science
at Delft University of Technology,
to be defended publicly on July 8th, 2024 at 14:00.

Student number: 4694252
Project duration: November 2023 – July 2024

Thesis committee:	Prof. dr. ir. M.J.T. Reinders,	TU Delft, Responsible advisor
	G. Vogel MSc.	TU Delft, Daily supervisor
	Dr. J.M. Weber	TU Delft, Daily co-supervisor
	Dr. M. Khosla	TU Delft

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Deep Reinforcement Learning for Inverse Synthetic Polymer Design

M.P.C van der Werf

June 24, 2024

Abstract

Synthetic polymers are crucial in diverse industries, but current AI-driven design methodologies primarily target linear homopolymers, with limited emphasis on developing customized approaches for copolymers. To address this gap, we introduce a generative model for goal-directed synthetic copolymer design using reinforcement learning. Our model operates in a graph generation environment, facilitating efficient monomer unit design while incorporating domain-specific constraints to ensure high validity rates. In a case study optimizing for Hydrogen Evolution Rate (HER) and synthetic accessibility, our approach showcases the efficacy of reinforcement learning in advancing copolymer design. Furthermore, experimental results underscore the challenges in designing effective scoring functions due to the sparse nature of polymer datasets, emphasizing the need for robust property predictors in polymer design methodologies before integrating more complex generative models into the design process.

1 Introduction

Polymer materials are essential in many industries due to their unique properties and versatility. Designing new molecules is a key objective in various chemical applications, from small molecules for drug design to large macromolecules for polymer synthesis. The vast extent of the chemical space, with an estimated order of 10^{60} possible structures for drug-like molecules alone [1], makes exhaustive enumeration approaches infeasible. As a result, significant effort has been devoted to exploring AI-driven techniques for efficient proposal of new candidate molecules [2, 3]. These advancements enable the concept of inverse molecular design [4], where scientists start with the desired properties of a molecule and work backward to determine the molecular structure that will exhibit these properties. This approach allows for a more systematic method compared to direct molecular design, where an initial molecular structure is incrementally updated to better suit imposed requirements.

Generative models for polymer design are still in the early stages of development [5], with significant potential yet to be realized. Among existing methods, one strategy leverages a Variational Autoencoder (VAE) [6] to learn the underlying distribution of a polymeric dataset [7, 8]. To tailor these VAE-based models for inverse molecular design, examples from molecular design illustrate how property predictors can be employed to filter candidate molecules [9] or be used to optimize over the latent space [10]. Alter-

natively, the latent space can be structured according to a molecular property by training predictors in parallel with the VAE [7] or by employing a Conditional VAE [11]. However, while effective within the boundaries of the latent distribution, these methods can face challenges in extrapolating beyond the training dataset. This effect is likely even more pronounced in polymer design due to the often sparse, inaccessible, or incomplete nature of available datasets [7].

Kim et al. [12] employed an Evolutionary Algorithm (EA) to discover homopolymers exhibiting high thermal stability at high temperatures, leveraging biologically inspired mechanisms such as mutation and crossover to evolve a population of candidate molecules iteratively. More recently, Ma et al. [13] rephrased the polymer design problem as a sequential decision-making task, with Reinforcement Learning (RL) being utilized to develop homopolymers with desired thermal conductivity. In this approach, RL algorithms learn to generate molecules by iteratively selecting actions that modify a molecular structure while aiming to optimize domain-specific properties [14]. This methodology has been explored across a wide range of molecular design benchmarks [15], demonstrating either superior or similar performance compared to other methods within restricted oracle call budgets, a factor particularly significant in the context of polymer design.

The choice of representation for molecules is highly relevant when designing a molecular design environment amenable to Reinforcement Learning. In this context, text-based representations such as SMILES [16] have been a conventional choice [3]. In SMILES,

each character represents a specific atom or bond. This format enables straightforward manipulation by RL algorithms, allowing actions that directly augment the string, changing the molecular structure [14, 17]. To denote the stochastic nature of polymers, Lin et al. [18] introduced BigSMILES, representing polymeric fragments by a list of repeating units, which are encoded following the SMILES syntax. Recently, Schneider et al. [19] extended this representation by incorporating additional information, such as the connection probabilities among repeating units.

While SMILES-based representations are compact and easy to generate, their inherent redundancies and lack of structural detail can hinder the ability of generative models to fully understand chemical and structural relationships between molecules [20]. Other text-based representations attempt to address some limitations of SMILES. For example, Thiede et al. [21] employed the SELFIES [22] syntax in their molecular design environment, leveraging its modular structure and 100% validity. However, due to their simplified grammar and the lack of explicit bond information, such representations may still struggle with capturing complex structures, like polymeric materials.

Recent years have seen the introduction of RL-based generative models that directly manipulate molecular graphs [23, 24, 25, 26]. These models typically represent the state as the current molecule, with actions corresponding to modifications to the graph, such as adding or removing atoms or bonds. For instance, Zhou et al. [24] used Morgan fingerprints, a type of molecular fingerprint, to encode the current state into a fixed-size vector, which was then used in the decision-making process for goal-directed small molecule design. Derived from the extended connectivity fingerprint (ECFP) [27], Morgan fingerprints encode molecular structure information by considering atom neighborhoods within a specified radius, converting this structural data into a bit vector. Due to their numerical vector format and computational efficiency, fingerprints are frequently adopted as features in machine learning models for chemistry [28]. They are also utilized in polymer discovery applications, including techniques tailored for polymer networks [29].

Over the past years, several frameworks have been developed for graph-based molecular design using graph neural networks (GNNs) [23, 26]. These frameworks, operating directly on the molecular graph, have effectively employed reinforcement learning for goal-directed molecule design [23, 30]. Unlike text-based and fingerprint-based representations, graph

representations can be easily extended to accommodate additional features by augmenting the graph matrix representation. Aldeghi and Coley [31] introduced a graph representation tailored specifically for polymers, coupled with a weighted directed message passing neural network (wd-MPNN) operating directly on the graph structure. Their evaluation on a property prediction task revealed that this method outperformed a traditional fingerprinting technique when sufficient data was available. This underscores the potential benefit of employing end-to-end graph-based approaches, which enable the explicit integration of molecular ensemble information within the architecture.

Existing polymer design methods are limited to homopolymers [7, 12, 13], or leverage a Variational Autoencoder (VAE) [7, 8], limiting their applicability for truly de novo design. To address this gap, we propose a graph-based design method amendable by reinforcement learning. While our work is constrained to the design of the monomer units, future extensions could enlarge the decision-making to form synthetic polymers from scratch. By leveraging a relational graph convolutional network (R-GCN), our model learns to generate polymers end-to-end. Moreover, the use of a graph representation allows us to incorporate domain-specific constraints on the action space, resulting in high validity rates amongst generated polymers.

We illustrate our approach in a case study, targeting the generation of synthetically accessible copolymers with high sacrificial Hydrogen Evolution Rates (HERs). Experimental results show that the model can be pretrained to generate polymers similar to an expert dataset, containing over 39K copolymers. By the subsequent use of reinforcement learning, we show that the output distribution of the model can be successfully moved towards more desired regions of the chemical space. Our study highlights the importance of including sufficient atom descriptors in the graph representation. Furthermore, our work sheds light on potential pitfalls in designing robust property predictors customized for polymers.

2 Problem Definition

In this chapter, we begin by explaining the employed graph representation for polymers, which is essential for generating polymer graphs. We then discuss how our graph generative environment defines actions and how these actions can be used to augment polymer graphs. Lastly, we describe how the environment enables the efficient detection of invalid actions.

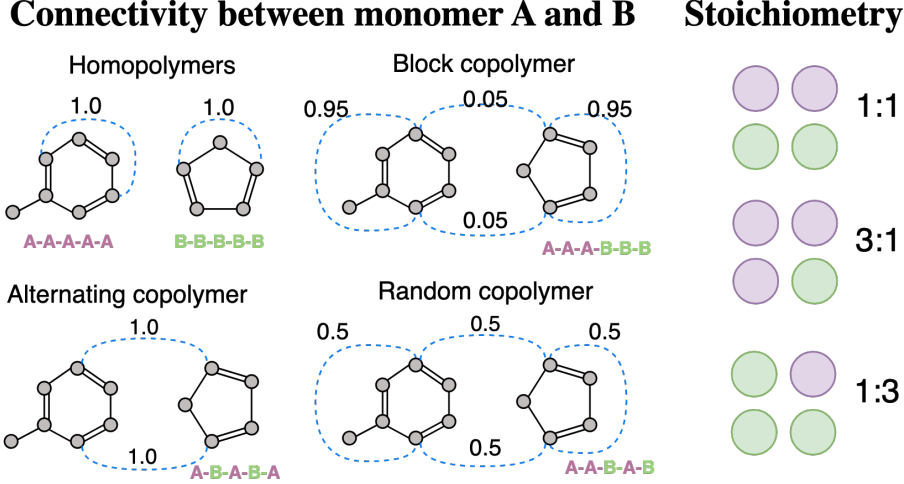


Figure 1: Different polymer topologies represented by a graph and associated stoichiometry. The graph contains a deterministic subgraph for each unique monomer unit found across the polymer chain. Cross-monomer bonds are represented by stochastic edges ($p \leq 1.0$). Illustrated by the blue, dashed lines, the associated probability of a bond of this type reflects the chance of this bond occurring in the polymer chain. Another important aspect of the polymer topology is the stoichiometry, depicting the relative abundance of each monomer unit in the entire polymer chain. Figure adapted from [31].

Graph Representation We represent a polymer chain as a stochastic graph G with associated stoichiometry ν [31] (Fig. 1). The graph consists of a union of all m unique monomer units. In G , nodes represent atoms, and edges represent bonds. Moreover, each edge is weighted with a probability p reflecting the chance of a bond occurring in a polymer chain. Here, we can distinguish between two types of bonds, namely cross-monomer bonds ($p \leq 1.0$) and within-monomer bonds ($p = 1.0$). Within-monomer bonds describe how atoms within a unique subgraph are connected, whereas the combination of cross-monomer bonds and stoichiometry, hereafter referred to as higher-order structure, are used to describe the average structure of the repeating unit.

In this work, we define G as (M, E, \hat{E}, F) . Here, $M \in \{0, 1\}^{n \times m}$ encodes for each node n to which monomer unit it belongs to. $E \in \{0, 1\}^{b \times n \times n}$ and $\hat{E} \in [0, 1]^{b \times n \times n}$ are both edge-conditioned adjacency tensors, assuming b edge types. Here, E and \hat{E} capture, respectively, all possible within-monomer and cross-monomer bonds. Lastly, $F \in \mathbb{R}^{n \times d}$ is the feature matrix, assuming all nodes have d features. The main objective of this work is to generate polymers that optimize a scoring function $S(G, \nu) \in \mathbb{R}$. This translates to maximizing $\mathbb{E}_{G', \nu'}[S(G', \nu')]$, where G' denotes a generated graph, ν' reflects the chosen stoichiometry, and S represents domain-specific statistics of interest.

In this work, we assume that we can perfectly subdivide $S(G, \nu)$ in a summation over scores over all unique monomer units (Eq. 1). Importantly, the scoring of an individual monomer unit k consists of a multiplication between a term dependent on the entire polymer graph and stoichiometry $S_{pol}(G, \nu)$, and a component solely reliant on the monomer unit chemistry $S_{mon}(g(G, k))$ (Eq. 2). Here, $g(G, k) \subset G$ represents the subgraph corresponding to monomer unit k , which is one of the m subgraphs obtained by removing all cross-monomer bonds and all nodes v_i not part of the monomer unit, i.e. $M_{i,k} \neq 1$. This formulation allows us to explore generative models that benefit from more precisely identifying the actions responsible for specific parts of the scoring.

$$S(G, \nu) = \frac{1}{m} \sum_{k=1}^m S_M(G, \nu, k) \quad (1)$$

$$S_M(G, \nu, k) = S_{pol}(G, \nu) * S_{mon}(g(G, k)) \quad (2)$$

Polymer Completion Environment We consider the problem of polymer chain generation as a hierarchical decision-making problem. Specifically, we explore a two-phase design, where the first phase is responsible for describing the higher-order structure of the polymer chain, and the second phase entails the fine-grained design of its monomer units. In this work, we focus solely on the second phase of the proposed polymer design approach. We refer to the re-

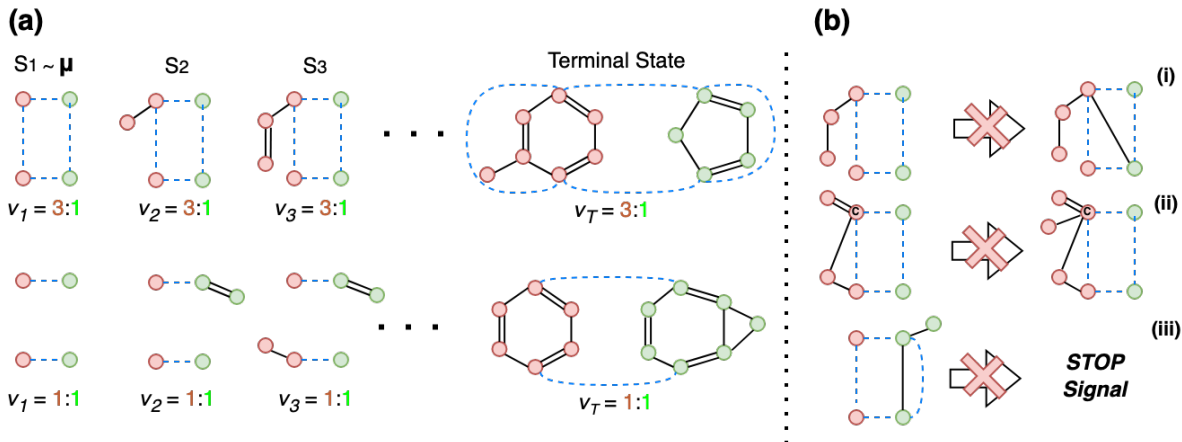


Figure 2: An overview of the Markov Decision Process (MDP) used for modeling the proposed polymer completion optimization problem. (a) The initial state $s_1 = (G_1, \nu_1) \sim \mu$ is sampled from the initial state distribution μ . At each timestep t , an action $a_t \in A_{s_t}$ can be selected that either augments the graph or transitions the state to a terminal state. (b) We can identify several actions leading towards the generation of invalid polymers. Specifically, from top to bottom, (i) actions connecting two existing nodes belonging to different monomers using a within-monomer bond, (ii) actions violating valency constraints, (iii) actions signaling for graph completion whilst at least one monomer unit (red) is still 'disconnected'.

sulting problem as *polymer completion*. Illustrated in Figure 2a, we model this problem using a Markov-Decision Process. A state s_t after t timesteps in the environment represents a polymer chain, reflected by a graph G_t and a stoichiometry ν_t . An episode starts by sampling an initial state from the initial state distribution $s_1 = (G_1, \nu_1) \sim \mu$. Importantly, G_1 adheres to a few constraints. Namely, it does not include any within-monomer bonds (Eq. 3), and each node $v_i \in G_1$ contains at least one cross-monomer bond (Eq. 4).

$$\max_{j,r} (E_{r,i,j} + E_{r,j,i}) = 0 \quad \forall v_i \in G_1 \quad (3)$$

$$\max_{j,r} (\hat{E}_{r,i,j} + \hat{E}_{r,j,i}) > 0 \quad \forall v_i \in G_1 \quad (4)$$

We model the action space of the polymer completion environment similar to the approach of GCPN [23], which introduced a molecular graph generation environment amendable by reinforcement learning. An action in the environment corresponds to any of 3 types of actions (1) connecting two existing nodes in G_t , (2) connecting an existing node $v_i \in G_t$ to a new node $v_u \in C$, (3) signaling graph completion. Importantly, actions can only augment the graph using within-monomer bonds, keeping intact the stoichiometry and set of cross-monomer bonds. With these higher-order features remaining fixed during an episode, it is conceivable that optimal behavior in the environment entails adjusting the selection of action,

especially if the optimal monomer chemistry relies heavily on the higher-order structure to exhibit desired properties.

$$a_t = (a_{first}, a_{second}, a_{edge}, a_{stop}) \quad (5)$$

An action a_t at timestep t in the environment comprises four sub-components, namely the selection of a first node, a second node, edge type, and stop signal (Eq. 5). If the stop signal $a_{stop} = 1$ is selected, all other subactions are ignored and the environment transitions to a terminal state, completing the graph. The choice of the first node is constrained to existing nodes in G_t , whereas for selecting the second node it is also possible to choose a new node from the set of candidate nodes C . In the environment, we allow for a maximum of T actions, finalizing the graph automatically if this limit is reached.

Detection of Invalid Actions Since partially generated subgraphs represent valid substructures, the environment enables the efficient detection of actions violating domain-specific constraints (Fig. 2b). For instance, an action is invalid if it connects two existing nodes belonging to different monomers i.e. nodes u, v for which $M_u \neq M_v$. Moreover, an action signaling for graph completion is only considered valid if all monomer units are 'connected'. We can validate this by looking for a path $p \subseteq E$ over within-monomer bonds between any two nodes u, v belonging to the same monomer $M_u = M_v$.

Additionally, actions are invalid if they violate the octet rule for one or both endpoints of a newly introduced bond, a constraint already incorporated in other molecular graph generative models [23, 24]. To adapt this rule for polymer graph representations, we require $r(v_i) \geq 0$ for any node v_i in the resulting graph G_{t+1} (Eq. 6), assuming hydrogen atoms are implicit.

$$r(v_i) = \text{val}(v_i) - \sum_{j \in N_i^r} \sum_{r=1}^b d_r E_{r,i,j} - \max_{j,r} (d_r [\hat{E}_{r,i,j}]) \quad (6)$$

Above, $\text{val}(v_i)$ is the intrinsic valence of node v_i , and d_r denotes the bond degree (single, double, or triple). The term $\max_{j,r} (d_r [\hat{E}_{r,i,j}])$ accounts for the highest degree of polymerization in the worst-case scenarios, assuming \hat{E} is symmetric for simplicity. This equation helps detect invalid actions by ensuring nodes do not exceed permissible valence, maintaining the chemical validity of polymer structures.

3 Methodology

In designing our approach for polymer completion, we developed an agent capable of dynamically interacting within this specialized environment. At the core of this agent’s decision-making lies a policy network π_θ , which is parameterized by θ . This network, conditioned on the current state of the environment, generates a probability distribution over all possible actions $\pi_\theta(a|s) = P(a|s; \theta)$. Subsequently, by continuously sampling actions from π_θ , this network facilitates the generation of new graphs.

In this chapter, we describe the architecture of the agent policy π_θ , which uses graph convolution at the core of its decision-making. Moreover, we describe how the agent can be pretrained to generate polymers similar to an expert dataset, and how it can be finetuned through reinforcement learning to generate more desired polymers.

Graph Convolution To facilitate action prediction, we compute node embeddings for all nodes in the current graph G_t using a Relational Graph Convolutional Network (R-GCN) [32]. This type of neural architecture operates directly on graphs and is designed specifically to deal with multi-relational data. In this work, we define the propagation rule for updating the embedding of a node v_i as follows:

$$h_i^{(l+1)} = \sigma(W_0^{(l)} h_i^{(l)} + m_i^{(l+1)}) \quad (12)$$

Here, $m_i^{(l+1)}$ reflects a relational-specific transformation of all messages received from neighboring nodes:

$$m_i^{(l+1)} = W_{agg}^{(l)} (m_{i,0}^{(l+1)} \oplus \dots \oplus m_{i,b}^{(l+1)}) \quad (13)$$

$$m_{i,r}^{(l+1)} = \sum_{j \in N_i^r} c_{i,j,r} W_r^{(l)} h_j^{(l)} \quad (14)$$

$$c_{i,j,r} = \frac{E_{r,j,i} + \hat{E}_{r,j,i}}{\sum_{k \in N_i^r} E_{r,k,i} + \hat{E}_{r,k,i}} \quad (15)$$

We apply convolution to the current graph G_t . In this light, we define $X^{(L)} = \{h_1^{(L)}, h_2^{(L)}, \dots, h_n^{(L)}\}$ as the set of final node embeddings obtained after applying L layers of convolution starting from the node feature matrix F . The propagation rules we defined closely follow the original work, except for our definition of $c_{i,j,r}$. This coefficient is adjusted to accommodate stochastic edges by normalizing edge weights based on the total incoming sum of edge weights, as opposed to normalizing by the number of incoming edges.

Action Prediction For selecting subactions, we largely follow the modeling approach and architecture of GCPN [33] which involves a sequential selection process. Specifically, this action selection process starts with selecting the first node, before the second node, edge type, and stopping signal. For each subaction type, the policy network π_θ contains a Multilayer Perceptron m , which is used to map an input vector to a continuous value for each available subaction. These continuous values are then used to form an action probability distribution via the softmax function (Eq. 7-11). Importantly, while GCPN resamples a new action if it is invalid, we solely softmax over all actions that do not violate any of the aforementioned validity constraints on the action space (Fig. 2b). With the large amounts of invalid actions in the environment, we observed that resampling in case of invalid actions did not constitute a computationally feasible approach.

In comparison to the architecture of GCPN, we incorporate several vectors specific to polymer design in the input space of the action selection networks. First of all, we incorporate the stoichiometry ν_t by including a weight matrix $W \in [0, 1]^n$ that denotes, for each node $v_i \in G_t$, the relative frequency of the monomer unit v_i belongs to. Depending on the subaction, this matrix is either concatenated to the input vector (Eq. 7-10), or used as a scaling factor (Eq. 11). In a similar manner, we incorporate a connectivity mask $C \in \{0, 1\}^n$ that is used to label nodes

$$N^{:n} = X \oplus W \oplus C; N^{n:} \sim \mathcal{N}(0, 1)^{c \times (l+2)} \quad (7)$$

$$a_{first} \sim f_{first}(s_t) \in \{0, 1\}^n, \quad f_{first}(s_t) = \text{SOFTMAX}_{\text{valid}}(m_f(N^{:n}, B^{:n})) \quad (8)$$

$$a_{second} \sim f_{second}(s_t) \in \{0, 1\}^{n+c}, \quad f_{second}(s_t) = \text{SOFTMAX}_{\text{valid}}(m_s(N_{a_{first}}, B_{a_{first}}, N)) \quad (9)$$

$$a_{edge} \sim f_{edge}(s_t) \in \{0, 1\}^b, \quad f_{edge}(s_t) = \text{SOFTMAX}_{\text{valid}}(m_e(N_{a_{first}}, B_{a_{first}}, N_{a_{second}})) \quad (10)$$

$$a_{stop} \sim f_{stop}(s_t) \in \{0, 1\}, \quad f_{stop}(s_t) = \text{SOFTMAX}_{\text{valid}}(m_t(\text{Agg}(X^{:n} * W), \max(C))) \quad (11)$$

in case a monomer unit is still 'disconnected' (Fig. 2b). In this scenario, $C_i = 1$ for each node $v_i \in G_t$ which does not belong to the largest cluster within the subgraph reflecting the monomer unit.

Additionally, following the Torchdrug [34] implementation of GCPN, we incorporate graph-level information B when selecting most subactions (Eq. 8-10). In this work, we explore three definitions of this vector. Namely, an aggregation of all node embeddings (Eq. 16), an aggregation of all node embeddings of nodes in the monomer subgraph (Eq. 17), and a hybrid approach that combines these strategies to enhance contextual understanding (Eq. 18). Incorporating graph-level information when selecting nodes and edge types allows for the inclusion of more contextual information, as opposed to being restricted to the neighborhood within L layers of convolution.

$$B_i^{graph} = \text{Agg}(X^{:n} * W) \quad (16)$$

$$B_i^{mon} = \text{Agg}(\{X_j^{:n} * W_j \mid v_j \in G_i, M_j = M_i\}) \quad (17)$$

$$B_i^{focus} = m_l(B_i^{mon}, B_i^{graph}) \quad (18)$$

Pretraining We pretrain the agent policy π_θ using an NLL loss, increasing the likelihood of reproducing polymer graphs within an expert dataset (Eq. 19). However, for graphs, there are numerous methods for spitting up expert graphs into series of state, action pairs, hereafter referred to as a trajectory. Solely responsible for determining the trajectory is the Graph Traversal Algorithm (GTA), which labels all nodes according to a predefined strategy. Mercado et al. [35] observed that the choice of GTA can influence the molecular graphs being generated after pretraining, with a BFS strategy leading to more 'circular' molecules, as opposed to DFS, in which the agent appeared to be more biased towards generating longer structures.

In this work, we explore two different GTA's, which we refer to as *BFS* and *BFS_M*. Regardless of the GTA, each unique polymer (G^*, ν^*) in the dataset maps to a single initial state $s_1 = (G_1, \nu_1)$. Importantly, multiple polymers in the dataset can map to the same initial state. The graph G_1 belonging to the

initial state contains all cross-monomer bonds of the expert graph, in addition to all nodes being either the source or destination of at least one of these bonds. For both GTA's, the same approach is used to obtain node orderings. This process involves creating a node ordering for all monomer units in G^* individually, i.e. for monomer unit k we sample a node $v \in f(G_1, k)$ and label all nodes in $f(G^*, k) - f(G_1, k)$ randomly in a BFS order.

The difference between both GTA's lies in the way the orderings for all monomer units are combined to form the series of actions A (Fig. 3), with the NLL loss designed to maximize the likelihood of taking the corresponding actions $P(A)$. Whereas in *BFS_M* the monomer units are (re)constructed one after another in A , *BFS* builds them up in a more distributed manner, alternating between the monomer units. The pretraining loop can be summarized by the steps below:

1. Sample the next batch of N expert polymers from the dataset. Each polymer is reflected by a tuple (G^*, ν^*) .
2. For each polymer, extract the initial state and create a series of actions \mathbb{A}_i (Fig. 3).
 - (a) Extract the unique initial state: $s_1 = (G_1, \nu_1)$ with $G_1 \subset G^*$ and $\nu_1 = \nu^*$.
 - (b) Obtain a random BFS node ordering for all nodes in $G^* - G_1$.
 - (c) Combine all node orderings according to the chosen GTA, and use this to form \mathbb{A}_i .
3. Update π_θ as to minimize the NLL loss (Eq. 19).
4. Go back to step 1.

$$J^{NLL}(\theta) = -\frac{1}{N} \sum_{n=1}^N \log P(\mathbb{A}_i)_{\pi_\theta} \quad (19)$$

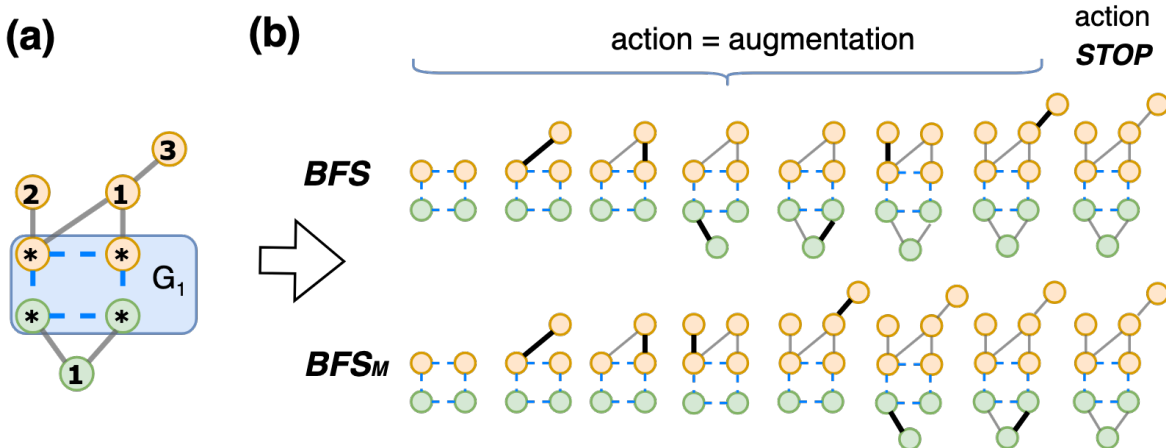


Figure 3: Method used in this work during pretraining to create a series of actions \mathbb{A}_i from a tuple (G^*, ν^*) , representing a polymer in an expert dataset. (a) After extracting the unique initial state $G_1 \subset G^*$, a random BFS node ordering is created for each monomer unit. (b) These node orderings are then combined using a specified Graph Traversal Algorithm (GTA). Starting from G_1 , the BFS approach reconstructs the expert polymer in a balanced manner, while BFS_M assembles the polymer monomer-by-monomer.

$$J^{BAR}(\theta) = \frac{1}{M} \sum_{i=1}^M \alpha \cdot L(\pi_{prior}, (\mathbb{G}_i, \mathbb{V}_i), \mathbb{A}_i, \theta) + (1 - \alpha) \cdot L(\pi_{best}, (\hat{\mathbb{G}}_i, \mathbb{V}_i), \hat{\mathbb{A}}_i, \theta) \quad (20)$$

$$L_P(\pi_{ref}, (G, \nu), A, \theta) = -[\log P(A)_{\pi_{ref}} + \sigma S(G, \nu) - \log P(A)_{\pi_{\theta}}]^2 \quad (21)$$

$$L_M(\pi_{ref}, (G, \nu), A, \theta) = -\frac{1}{m} \sum_{k=1}^m [\log P(f(A, k))_{\pi_{ref}} + \sigma S_M(G, \nu, k) - \log P(f(A, k))_{\pi_{\theta}}]^2 \quad (22)$$

Finetuning For finetuning the agent policy π_{θ} we adapt the Best Agent Reminder (BAR) loss [30]. In addition to π_{θ} , this loss function keeps track of two other policies, namely the prior policy π_{prior} and the best policy π_{best} . The prior policy π_{prior} constitutes the policy obtained after pretraining and remains fixed during finetuning, whereas π_{best} is periodically set to π_{θ} if the agent policy improves.

The BAR loss takes the form as written in Eq. 20, with α serving as a scaling factor, and L defined as the agreement between the agent policy π_{θ} , and an augmented log-likelihood. In this work, we explore two variants of L , namely L_P (Eq. 21) and L_M (Eq. 22). Both variants define the augmented log-likelihood as the log-likelihood of a reference policy modulated by a score. However, while L_P follows the traditional approach of using all actions taken by the agent and the score defined on the entire graph [14, 30], L^M exploits the fact that the score is defined as a summation of scores of individual monomers (Eq. 2). Specifically, L^M calculates the average agreement between the agent policy and augmented log-likelihood across

all individual monomers, which uses the scoring S_M (Eq. 2) and model log-likelihood across a subset of all actions $f(A, k) \subset A$. Here, $f(A, k)$ contains solely the stop action, in addition to all actions augmenting the neighborhood of at least one node $v_i \in g(G, k)$ that belongs to monomer unit k ($M_{i,k} = 1$).

By anchoring the agent policy π_{θ} in the prior policy π_{prior} , the BAR loss function shifts the probability distribution from that of the prior policy towards a distribution modulated by the desirability of the graphs, favorable for chemical diversity [14]. Moreover, by generating half of the molecules using the best policy π_{best} , the BAR loss reminds the agent policy of actions leading towards high-scoring graphs. The finetuning loop can be summarized as follows:

1. Initialize π_{θ} and π_{best} to the policy π_{prior} obtained after pre-training.
2. Equally distributed amongst all unique initial states encountered during pretraining, create a batch of M initial states \mathbb{S}_1 (subgraphs with associated stoichiometries \mathbb{V}).

3. For each $s_1^i \in \mathbb{S}_1$, generate a graph..
 - (a) using the agent policy π_θ , with \mathbb{A}_i representing the episodic set of actions taken to form the final state $(\mathbb{G}_i, \mathbb{V}_i)$.
 - (b) using the best policy π_{best} , with $\hat{\mathbb{A}}_i$ representing the episodic set of actions taken to form the final state $(\hat{\mathbb{G}}_i, \mathbb{V}_i)$
4. Calculate a score for all final states (Eq. 1).
5. Compute the action probabilities of
 - (a) π_θ and π_{prior} assigned to \mathbb{A} .
 - (b) π_θ and π_{best} assigned to $\hat{\mathbb{A}}$.
6. Update π_θ as to minimize the BAR loss (Eq. 20).
7. Every 5 learning steps, repeat steps 2 and 4 with a larger batch of generated molecules. If the graphs generated with π_θ have a higher mean score, initialize π_{best} with the weights of π_θ .
8. Go back to step 2.

4 Case Study and Dataset

We illustrate our approach in a case study targeting the generation of synthetically accessible linear copolymers with high sacrificial Hydrogen Evolution Rates (HERs). This metric reflects the ensemble’s capability to catalyze the hydrogen evolution reaction, thereby generating hydrogen gas. Since this is a clean-burning fuel, the discovery of novel polymers with high HERs could contribute to a transition towards more sustainable energy sources. However, predicting HERs of novel compounds is difficult, since it does not correlate strongly with any single physical property.

To bridge this gap, Bai et al. [36] created a machine learning model predicting HERs using four molecular properties. In a dataset comprising 6354 copolymeric materials, the authors observed an inverse relationship between HER and Electron Affinity (EA). Moreover, polymers with higher HERs appeared to frequently have an Ionization Potential (IP) closer to 1.0 eV. In this work, we combine these two observations with a Synthetic Accessibility (SA) score, calculated for each monomer unit, to define the scoring function $S(G, \nu)$. As such, favoring polymers that are more likely to be synthesizable, and more probable to have high HERs. Each of the three aforementioned properties is individually scored, and depicted below.

$$S_{IP}(G, \nu) = [1 - |1 - IP(G, \nu)|]_0^1 \quad (23)$$

$$S_{EA}(G, \nu) = \left[\frac{-EA(G, \nu) - 3}{2} \right]_0^1 \quad (24)$$

$$S_{SA}(g(G, k)) = \left[\frac{7.0 - SA(g(G, k))}{4} \right]_0^1 \quad (25)$$

Above, $|x|_0^1$ returns the clipped value of x between 0 and 1. The scoring ranges were chosen to ensure that the polymers generated after pretraining exhibit a diverse range of scores across individual properties, while none achieve high scores across all properties simultaneously. $SA(g(G, k))$ is the predicted SA score calculated on the subgraph of $g(G, k) \subset G$ reflecting monomer unit k (Chapter 2). In this context, we modify $g(G, k)$ to include wildcard atoms, which are used to denote points of attachment to other monomer units. This representation is illustrated in Figure 4b, where monomer units are depicted with these wildcard atoms.

To calculate SA scores on this subgraph, we utilize the RDKit [37] implementation of the scoring algorithm introduced by Ertl et al. [38]. This algorithm incorporates a fragment score derived from statistical analysis of substructures in the PubChem database [39]. The fragment score reflects cumulative knowledge of synthetic accessibility based on common structural motifs found in a vast set of representative molecules. It is computed by summing contributions from these motifs and dividing by the number of such fragments present in the molecule under evaluation. Additionally, the scoring algorithm includes a complexity penalty factor. This penalty accounts for molecular characteristics such as the presence of large rings, non-standard ring fusions, and stereocomplexity. These factors collectively enhance the algorithm’s accuracy in predicting the synthetic accessibility of molecules.

The predictors IP and EA utilize a weighted Directional Multi-Passing Neural Network (wD-MPNN) to predict single molecular properties (Fig. 4a). This architecture is inspired by Aldeghi et al. [31], but differs by employing node-centered messages instead of messages centered on directed edges, as detailed by Yang et al. [40]. The predictors are trained on a dataset comprising 42,966 polymer graphs (Fig. 4b), where property values are derived from density functional theory calculations. This dataset augments the aforementioned dataset introduced by Bai et al. [36] by considering three chain architectures, and stoichiometries.

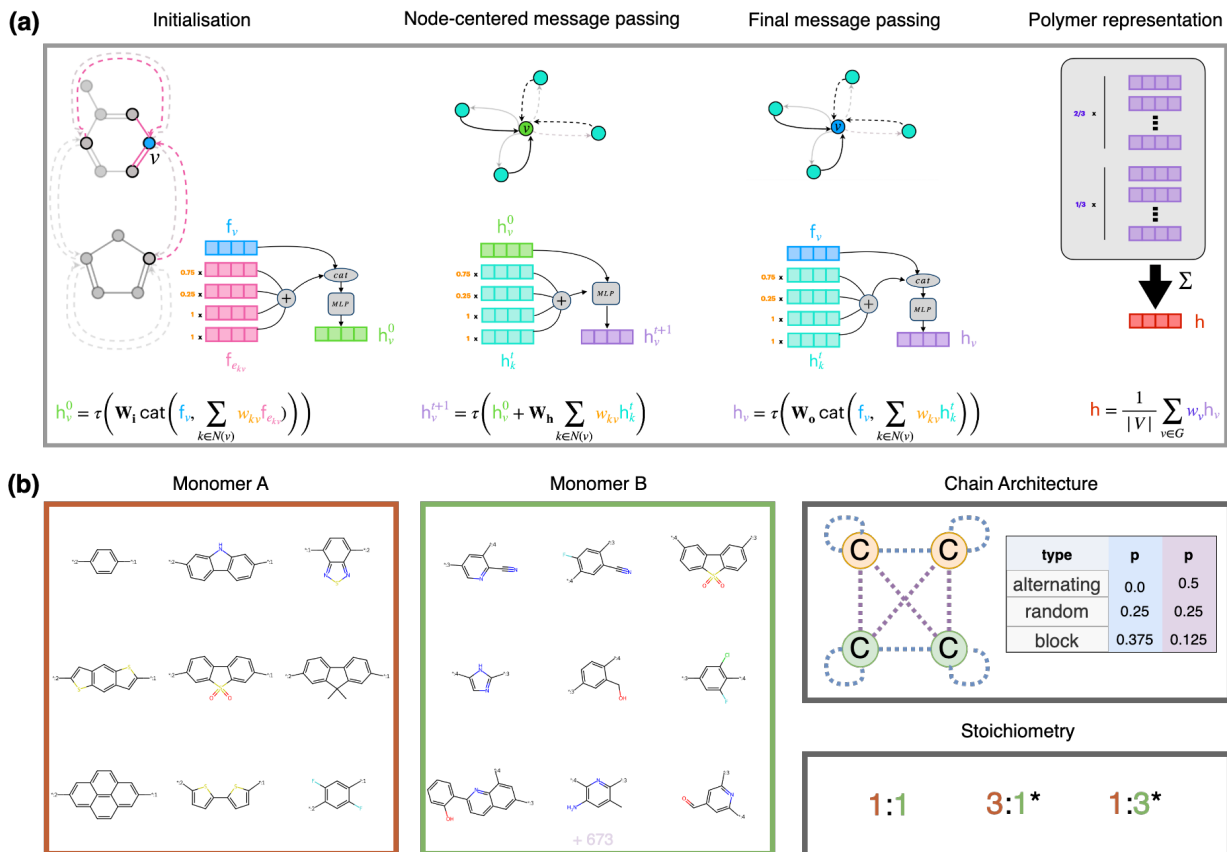


Figure 4: Architecture of the property predictor and dataset utilized for its training. **(a)** The architecture of the property predictor involves node-centered message passing. Node messages are initialized based on atomic properties within each node’s local neighborhood. This process includes 3 steps of message passing where node-centered messages are updated using current messages and weighted messages from neighboring nodes. In the final step of message passing, the node feature vector is concatenated with neighboring node messages before projecting to the final node embedding. To derive polymer features, node embeddings are aggregated via a weighted average. This involves scaling each node embedding according to the relative abundance of its corresponding monomer unit as defined by the polymer’s stoichiometry. Prediction of the property is performed using a multi-layer perceptron on these polymer features. **(b)** The dataset used to train the property predictor was generated by perturbing two lists of monomer units and including three types of chain architectures, along with three stoichiometry values [31]. * The alternating chain architecture is not available for this stoichiometry.

For training the property predictors, we considered a single run with an 80/20 train/test split. On the test set, the property predictor for *EA* achieved an R^2 of 0.998 and an RMSE of 0.027. For *IP*, we achieved similar accuracy with an R^2 of 0.997 and an RMSE of 0.025.

For pretraining the agent, we use the same dataset used for training the property predictors. However, in this study, our focus was solely on augmenting the graph with neutral atoms to simplify the process. Consequently, due to the inability to reconstruct polymers with formal charges, we excluded all such polymers from the dataset. This decision resulted in an 8% reduction in the number of B monomers, decreasing from 681 to 631. For finetuning, we combine the aforementioned scores for EA and IP to form the polymer-level score (Eq. 26), whereas the monomer-level score is equal to the score obtained for the SA component. (Eq. 27). Similar to the approach used by Atance et al. [30], we assign a zero score if a polymer is not unique, valid, or properly terminated (PT) via sampling of a terminal action.

$$S_{pol}(G, \nu) = \begin{cases} 0 & \text{if } (G, \nu) \text{ is not } \{\text{unique, valid, PT}\} \\ S_{IP}(G, \nu) \cdot S_{EA}(G, \nu) & \text{otherwise} \end{cases} \quad (26)$$

$$S_{mon}(g(G, k)) = S_{SA}(g(G, k)) \quad (27)$$

Experimental Details We use the Adam [41] optimization algorithm for pretraining and finetuning of the agent. Included in the appendix, Table S1 details how node features are derived, and Table S2 contains all remaining fixed hyperparameters used in the experiments. When generating molecular graphs, we

allow for a maximum of $T = 68$ actions, equalling the number of steps necessary to form the largest graph encountered during pretraining. Furthermore, we use a batch size of $N = 140$ during pretraining, and a batch size of $M = 70$ during finetuning. We evaluate the agent policy π_θ and best policy π_{best} after every 5 learning steps by generating a larger batch of 280 graphs with both policies, updating π_{best} with the weights of π_θ if it achieves a higher average score.

Our implementation of the environment and methodology closely follows the reimplementations of GCPN [23] from Torchdrug [34]. We conducted our experiments using the HPC cluster at Delft University of Technology, utilizing only CPUs for all computations. Pretraining the model takes about 22 hours while finetuning requires around 10 hours. For logging and saving model weights, we used Weights and Biases [42].

Evaluation Metrics For evaluating the final agents obtained after pretraining and finetuning, we use the agent policy π_θ to generate a batch of 2100 graphs, which equals 300 graphs per unique initial state (Fig. 4b). Across this batch of graphs, we use a similar notation as Mercado et al [26] and calculate the metrics described in Table 1. Importantly, since Morgan fingerprints are not specifically tailored for synthetic polymers and only apply to deterministic graphs, we calculate the fingerprint on the disconnected subgraph obtained after removing all cross-monomer bonds. This makes the diversity invariant to both the stoichiometry and the chain architecture.

Table 1: Evaluation metrics used to assess different agents in polymer generation. The metrics cover various aspects such as validity, uniqueness, diversity, and specific chemical properties of the generated polymers. * The diversity is estimated on a smaller batch of 400 polymers to ease computation.

Metric	Description
PV	Percent of valid polymers.
PPT	Percent of polymers that were properly terminated via sampling of $a_{stop} = 1$ within T timesteps.
PVPT	Percent of valid polymers in the set of PPT molecules.
PU	Percent of unique polymers (graph and stoichiometry) in the set of PVPT polymers.
PU_G	Percent of unique graphs amongst the set of PVPT polymers.
PN	Percent of novel polymers in the set of PU polymers.
Diversity	Average Tanimoto distance across the Morgan Fingerprints of PU _G polymers* ($r = 3$).
EA_{av}	Average electron affinity amongst the set of PU polymers.
IP_{av}	Average ionization potential amongst the set of PU polymers.
SA_{av}	Average synthetic accessibility score across all monomer units in the set of PU polymers.
V_{av}	Average number of atoms amongst the set of PU polymers.
B_{av}	Average number of bonds amongst the set of PU polymers.
Score_{av}	Average $S(G, \nu)$ amongst the set of PU polymers.

5 Results and Discussion

We evaluated the impact of various training setups and architectural changes on the molecule distribution generated by the agent policy π_θ after pretraining on the dataset and subsequent finetuning on the optimization task. To effectively compare these changes, we considered six different types of agents. The *baseline* agent only considers node embeddings in the current monomer when constructing the global information vector ($B = B_{mon}$). Moreover, during pretraining, expert graphs are split into a series of actions, reconstructing the polymer in a balanced manner across all monomer units (GTA = *BFS*).

We define two other agent types with identical model architecture. *gta_mon* uses a different graph traversal algorithm during pretraining which reconstructs polymers monomer-by-monomer (GTA = *BFS_M*). *obs_graph* constructs the global information vector differently compared to *baseline*, considering all node embeddings in the process ($B = B_{graph}$).

Three additional agents exhibit slight variations in model architecture compared to the aforementioned agents. *obs_focus* employs a hybrid approach that combines a global embedding and a local monomer-specific embedding into a single global information vector ($B = B_{focus}$). *only_symbol* simplifies the feature matrix F by including only the atom symbol in node featurization (Tab. S1). Lastly, unlike the other agent types, *no_stoich* does not incorporate the stoichiometry in the node and graph embeddings. By systematically comparing these agent types, we aimed to evaluate the impact of architectural choices and training strategies on the diversity and quality of molecules generated by the agent.

5.1 Pretraining

For each agent type, we conducted 9 runs, which equals three runs per weight decay value in $[0, 1e^{-4}, 1e^{-3}]$ with a fixed learning rate of $3e^{-3}$. For each run, we saved the model after 800, 1600, and 2400 learning steps, leading to a total of 27 agents per agent type. Mean NLL loss values obtained from these runs are summarized in Table S3 located in the appendix.

Tendency to Overfit Amongst all policies, we observe that no policy came very close to the dataset in terms of average property values and size (Fig. 5). Upon closer inspection, we observe a tendency where occasionally the agent generates very large molecules with high values for EA and SA, whereas otherwise,

the molecules appear more similar to those in the dataset (Fig. 9). Additionally, we observe that these high EA and SA polymers are often not properly terminated via sampling of a terminal action (PPT). We hypothesize that this effect is caused by the small size of the dataset, making it hard for the policy to generalize to unseen parts of the chemical space. Furthermore, we believe that this occasionally leads to the generation of invalid polymers, as the agent may not successfully connect all monomer units within the allocated time budget. This scenario is the only situation where invalid polymers can be generated, as indicated by the consistent PVPT of 100 across all agents in our experiments.

With a total of 39,753 datapoints, the dataset is way smaller than typical datasets, such as ZINC, used for pretraining generative models [3]. Moreover, we believe that the dataset lacks diversity in comparison to most molecular datasets, as the copolymers are derived from a limited set of monomer units, chain architectures, and stoichiometry values (Fig. 4b). Although training for more learning steps or with lower weight decay values seems to bring V_{av} , EA_{av} and other property values closer to the dataset, we observe that this comes at the cost of lower uniqueness values, PU and PU_G , and lower diversity. We believe this highlights again the small size and lack of diversity in the dataset, causing the model to easily overfit. This hypothesis would also be in line with the results of Mercado et al. [26], suggesting that an ideal dataset for GNN-based molecular design should contain at least 100,000 molecules to prevent overfitting.

Table 2 provides an overview of the hyperparameters and reported metrics for a single agent for each agent type that has a uniqueness greater than 96 amongst generated graphs (PU_G), and an average electron affinity (EA_{av}) closest to that of the dataset. We think this approach balances to some extent between overfitting and generating molecules similar to the dataset, although a more rigorous method would likely involve withholding a validation set and selecting the best-performing architecture based on its performance on this set. Moreover, we only considered a limited set of hyperparameters, for which their values might depend on the architecture at hand. For instance, agents trained using solely the atom symbol as node features (*only_symbol*) achieve lower PU (Fig. 5), suggesting they are more prone to overfitting and could potentially benefit from reduced model complexity.

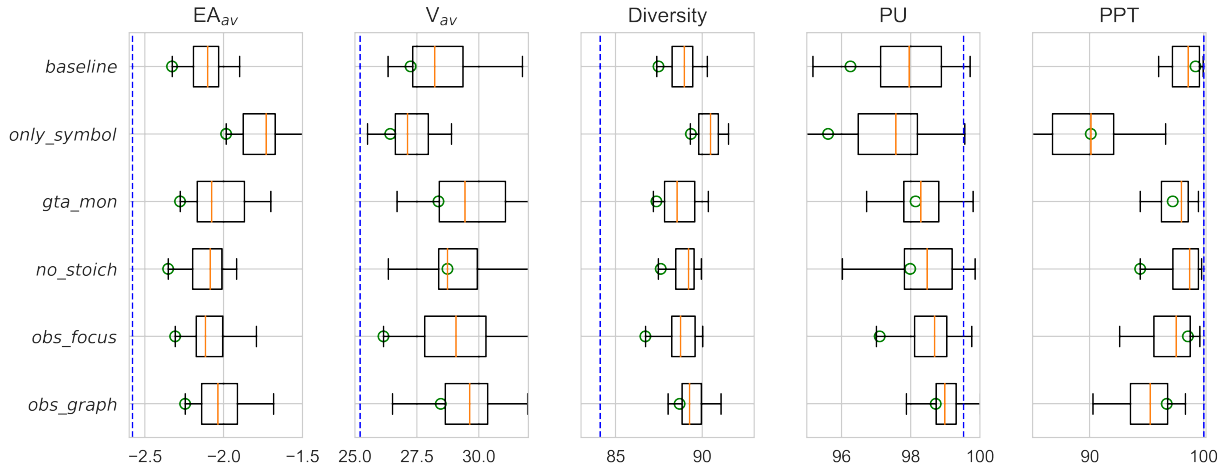


Figure 5: Comparison of mean electron affinity (EA_{av}), mean number of nodes (V_{av}), diversity, percentage of unique polymers (PU), and percentage of properly terminated polymers (PPT) across different agent types. Boxplots depict the distribution of these metrics across all 27 agents for each agent type. Green dots indicate the values for the agent with the lowest EA_{av} . Blue dashed lines show the average metric values from the pretraining dataset.

Table 2: Hyperparameters and reported metrics for the best agent found for each agent type. The best agent was selected based on having an EA_{av} closest to the dataset value of -2.58 , while also meeting the uniqueness constraint of $PU_G > 96.0$.

	Pretraining Configuration						
Agent Type	<i>baseline</i>	<i>only_symbol</i>	<i>gta_mon</i>	<i>no_stoich</i>	<i>obs_focus</i>	<i>obs_graph</i>	<i>dataset</i>
Connectivity Mask	✓	✓	✓	✓	✓	✓	-
Incorporate Stoichiometry	✓	✓	✓	X	✓	✓	-
All Node Features	✓	X	✓	✓	✓	✓	-
Graph-Level Information	B_{mon}	B_{mon}	B_{mon}	B_{mon}	B_{focus}	B_{graph}	-
Graph Traversal Algorithm	BFS	BFS	BFS_M	BFS	BFS	BFS	-
Learning Steps	2400	1600	800	2400	1600	2400	-
Weight Decay	$1e^{-3}$	$1e^{-3}$	0	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	-
	Evaluation Metrics						
PV	99.9	99.8	98.9	100.0	99.6	99.0	100
PPT	98.6	90.5	98.1	99.6	97.6	97.1	-
PVPT	100.0	100.0	100.0	100.0	100.0	100.0	-
PU	99.2	98.4	98.9	99.0	99.2	99.0	99.5
PU_G	97.5	97.1	96.1	96.3	97.2	96.4	14.2
PN	93.6	96.8	85.8	89.7	93.5	92.1	-
Diversity	89.7	91.0	89.1	89.6	89.6	89.7	84.1
EA_{av}	-2.11	-1.58	-2.0	-2.09	-2.1	-1.91	-2.58
IP_{av}	1.57	1.66	1.63	1.49	1.51	1.68	1.44
SA_{av}	4.99	5.55	5.04	5.03	5.07	5.11	4.54
V_{av}	31.0	28.0	29.5	28.4	31.2	29.5	25.2
B_{av}	34.1	30.2	31.8	30.8	34.2	31.9	27.0
$Score_{av} \uparrow$	0.01	0.0	0.01	0.01	0.01	0.01	0.03

Comparison Between Agent Types Experimental results show that the selected agent for *only_symbol* achieves low PTT and low similarity to the dataset in terms of average property values. We observe a similar effect for *obs_graph*, where the policy network shares the architecture of the baseline, but more global information is used to augment node embeddings ($B = B_{graph}$). We hypothesize that, for *only_symbol*, the agent lacks sufficient informative features for the task at hand, whereas the observations for *obs_graph* seem to suggest the relevance of more local information. This hypothesis is supported by the pretraining results, where the respective agent types consistently achieve higher training losses compared to other agents (Tab. S3).

However, an alternative explanation for the higher training loss observed in *obs_graph* could be the nature of the dataset itself, which is generated by perturbing a limited set of monomer units, chain architectures, and stoichiometry values (Fig. 4b). Consequently, the correct decision on how to augment a monomer unit during pretraining should rely solely on its intrinsic chemistry, while the chain architecture, stoichiometry, and chemistry of the other monomer can be regarded as 'noise'. Supporting this, we observe that not incorporating the stoichiometry, *no_stoich*, does not decrease PPT, or move property values further away from the dataset. A similar conclusion applies to *obs_focus*, which includes more global information (= 'noise') compared to *baseline*, but can distinguish more effectively between data from the current monomer and other monomers, unlike the agent for *no_graph*.

5.2 Finetuning

For finetuning, we employed a random search approach. Specifically, for the selected pretrained agents across all agent types (Tab. 2), we conducted 50 runs. For each run, we sampled a learning rate from the range [1.5e-4, 6e-4] and a weight decay value from the range [1e-2, 1e-5]. Specifically, we sampled a scalar $w \in [2, 5]$ and set the weight decay to $1e^{-w}$. We also varied σ within the range [5, 600]. Additionally, we randomly decided whether to define the cost function at the polymer level ($L = L_P$) or the monomer level ($L = L_M$). Another random decision was whether to include a best agent ($\alpha = 0.5$) or not ($\alpha = 0.0$). In scenarios without a best agent, we increased the number of epochs from 500 to 900 and the batch size from $M = 70$ to $M = 140$. This adjustment ensures a comparison with the same number of oracle calls, which is an important factor in molecular discovery applications.

Score and Diversity Trade-Off In our experiments, we observed a significant trade-off between Score_{av} and diversity, indicating that optimizing chemical structures often comes at the expense of reduced chemical diversity. However, this observation is expected, since high-scoring solutions typically occupy a smaller portion of the chemical space. Figure 6 provides an overview of the Pareto front between diversity and average score (Score_{av}) across various finetuning settings and agent types. It also includes more detailed insights into these Pareto front solutions, showing that higher-scoring solutions typically include larger molecules. Later in this section, we will investigate this phenomenon in more detail. Table 3 highlights six solutions sampled from the Pareto frontier, offering a comprehensive overview of all reported metrics.

Cost Function Notably, we observe that Pareto-optimal solutions with high Score_{av} consistently feature a cost function defined on the monomer level, which seems to suggest the benefit of being able to more precisely identify the source of scoring, thereby reducing variance. However, it is worth considering that this observation might be influenced by the fact that the monomer cost function defines the loss on a smaller set of actions, potentially resulting in larger updates during training. Hence, instead of employing a uniform interval for σ , individually tuning these values for each cost function could enable a fairer comparison. In a smaller significance, we believe the same principle applies to the different prior policies (Tab. 2). Among other factors, these policies can differ slightly in their degree of determinism, with more overfit policies likely requiring higher values for σ to obtain similar Score_{av} . Furthermore, incorporating the best agent ($\alpha = 0.5$) reminds the agent of actions leading to high-scoring polymers, in addition to the traditional approach of aligning with a prior agent [14], possibly amplifying the effect of smaller values for σ .

Inclusion of Best Agent We observe that including a best agent ($\alpha = 0.5$) does not seem to offer a significant advantage, as solutions trained without a best agent ($\alpha = 0.0$) still manage to converge to high-scoring outcomes within the same amount of oracle calls (Fig. 6). Several factors could explain this observation. For instance, the problem might not be sufficiently complex, reducing the need for a best agent to remind the current agent of actions leading to high-scoring solutions. Additionally, we only experimented with a single set of hyperparameters (such as α and update interval) based on values

Table 3: Hyperparameters and reported metrics for six agents on the Pareto front between Score_{av} and Diversity. Each column includes the hyperparameters used for training a single agent and the metrics reported across the batch of generated molecules by this agent.

	Finetuning Configuration						
Agent Type	<i>gta_mon</i>	<i>baseline</i>	<i>obs_focus</i>	<i>gta_mon</i>	<i>gta_mon</i>	<i>gta_mon</i>	<i>dataset</i>
Cost Function	L_M	L_P	L_P	L_M	L_M	L_M	-
Weight Decay	$8 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	$7 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$3 \cdot 10^{-3}$	$7 \cdot 10^{-3}$	-
Learning Rate	$4.2 \cdot 10^{-4}$	$1.5 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$	$4.5 \cdot 10^{-4}$	$2.4 \cdot 10^{-4}$	$4.5 \cdot 10^{-4}$	-
α	0.5	0.5	0.5	0.0	0.5	0.5	-
σ	20	597	87	340	284	458	-
	Evaluation Metrics						
PV	100.0	100.0	100.0	100.0	99.8	99.9	100
PPT	100.0	99.0	100.0	100.0	98.5	98.3	-
PVPT	100.0	100.0	100.0	100.0	100.0	100.0	-
PU	76.3	84.6	89.3	87.3	97.5	91.8	99.52
PU_G	57.9	67.9	69.9	69.8	91.5	82.5	14.2
PN	90.0	99.7	99.9	100.0	100.0	100.0	-
Diversity	79.1	73.7	69.2	65.0	60.8	53.5	84.1
EA_{av}	-3.55	-3.84	-3.92	-3.9	-4.01	-4.05	-2.58
IP_{av}	1.15	0.97	0.95	0.98	0.97	0.97	1.44
SA_{av}	4.89	4.13	4.03	3.6	3.56	3.45	4.54
V_{av}	21.1	31.0	29.5	39.0	44.5	50.5	25.2
B_{av}	21.6	32.8	31.1	42.2	48.3	55.3	27.0
$\text{Score}_{av} \uparrow$	0.14	0.27	0.3	0.35	0.4	0.42	0.03

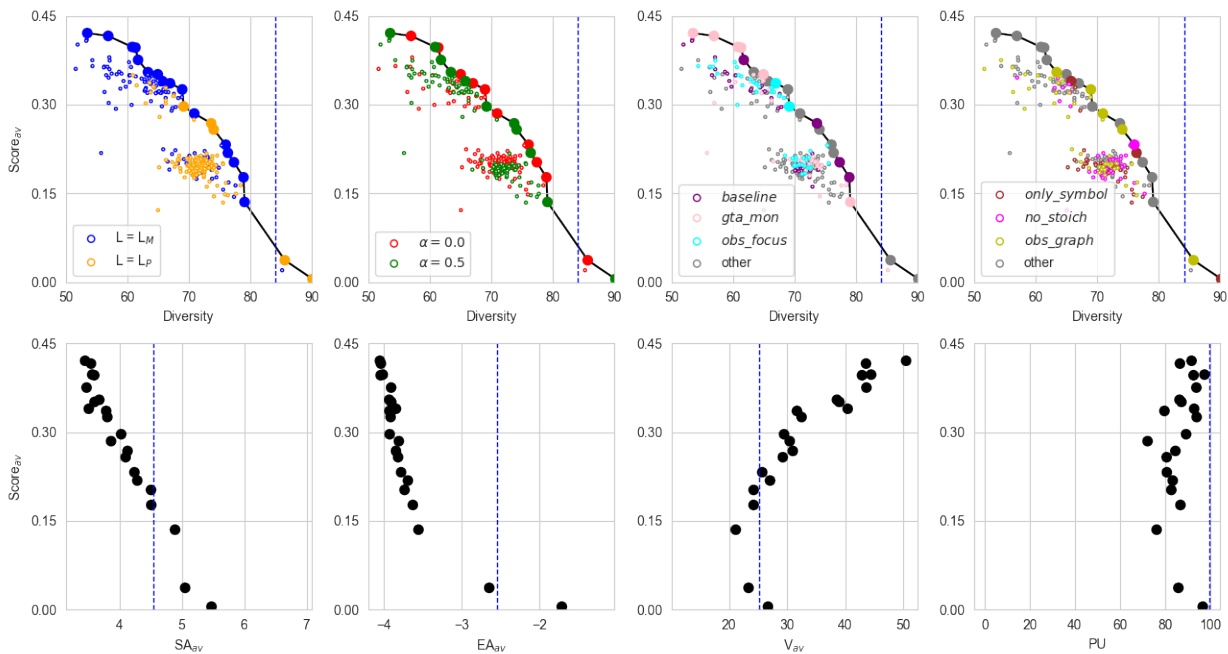


Figure 6: Pareto front analysis based on Score_{av} and diversity across the generated polymers by each agent. The top row illustrates the influence of cost functions, the inclusion of a best agent, and different agent types on Score_{av} and diversity, with larger points highlighting Pareto-optimal solutions. The blue vertical lines depict values for the reported metric calculated using the polymers in the dataset. The bottom row offers in-depth insights into the identified Pareto-optimal solutions, including average SA score (SA_{av}), average electron affinity (EA_{av}), average number of nodes (V_{av}), and percentage of unique polymers (PU).

Table 4: Top-2 agents found with highest Score_{av} across the agent types explored in all experiments. For each agent, the average score obtained (Score_{av}) and diversity are reported.

Agent Type	Score _{av}	Diversity	Score _{av}	Diversity	Agent Type	Score _{av}	Diversity	Score _{av}	Diversity
<i>gta_mon</i>	0.4214	53.49	0.4161	56.90	<i>obs_focus</i>	0.3749	57.12	0.3721	60.29
<i>baseline</i>	0.4100	53.39	0.4092	53.32	<i>no_stoich</i>	0.3458	56.96	0.3451	63.58
<i>obs_graph</i>	0.3753	54.33	0.3639	56.18	<i>only_symbol</i>	0.3292	66.41	0.2905	67.22

from Atance et al. [30], whereas optimal values may vary significantly depending on the specific problem. Importantly, we dedicated a large part of the oracle budget (28%) to evaluating the best agent against the current agent and deciding whether to update it. Possibly, reducing this fraction could improve sample efficiency when employing a best agent, provided that it does not significantly worsen the evaluation. Alternatively, combining the BAR loss with experience replay has shown the potential to directly enhance sample efficiency [43].

Global Information We observe that augmenting more global information to node embeddings (*obs_graph*) does not enhance performance, as few solutions incorporating this approach appear on the Pareto front (Fig. 6). Specifically, top-scoring agents, with the global information vector solely defined on nodes in the current monomer, outperform other approaches that include more node embeddings across the entire polymer (Tab. 4), suggesting the limited advantage of global insights. This conclusion also applies to the agent that can distinguish between local and more global information (*obs_focus*).

Upon closer examination, we observe that high-scoring agents appear to generate chemically similar molecules, predominantly forming polyphenyl chains with frequent amino groups (Fig. 7). Notably, the monomer units across all generated copolymers appear chemically similar, suggesting that the optimization target may prioritize smaller repetitive structures where the relative abundance of chemical structures is more important than their spatial distribution in the polymer. This observation suggests that the setup may be biased towards optimizing around smaller structures, where their relevance dominates the landscape, obscuring any potential benefits from integrating more global information. Alternatively, the optimization process may have converged to a global optimum that inherently favors local information, although this remains uncertain. Later, we will discuss how limitations of the scoring function could also play a role in our findings.

Stoichiometry We observe that finetuned agents without the stoichiometry incorporated in the node and graph embeddings (*no_stoich*) do not achieve a Score_{av} higher than 0.35 (Tab 4). In contrast, this threshold is exceeded multiple times by other agents that do incorporate the stoichiometry, except for agents trained solely on the atom symbol as node features. A closer examination of the highest-scoring agents reveals that when we artificially evaluate using a different stoichiometry than the one used during molecular generation, there are significant drops in Score_{av} (Fig. 7a). This suggests that the agent adapts the graph design to the given stoichiometry, which is randomly sampled during each epoch. Conversely, the best solution of *no_stoich* does not exhibit the same magnitude of drop in Score_{av} when we evaluate with a different stoichiometry, suggesting that it converges to a local optimum where the monomer units fit independently of the stoichiometry.

Notably, for the highest-scoring agent, the size of the monomer unit appears to correlate with its relative frequency (Fig. 7a), a trend not observed in the *no_stoich* solution. Considering again that the distinct monomer units across all generated graphs appear chemically similar, we anticipate that nearly identical copolymers will form when simulating macromolecules across all sets of graphs and stoichiometry values. Therefore, we expect minimal differences in predicted properties based on these chemical similarities, despite notable variations in graph size, stoichiometry, and chain architecture.

However, in a toy example featuring two different monomer units, we demonstrate that the employed property predictors are sensitive to changes in these chemical characteristics (Fig. 8), despite not resulting in a different copolymer being represented. Building on our previous observations, we hypothesize that high-scoring agents exploit this lack of robustness from the property predictor, scaling the monomer size with its relative frequency to 'trick' the property predictor, even though this might not necessarily be favorable for the underlying goal of maximizing the HERs of copolymers.

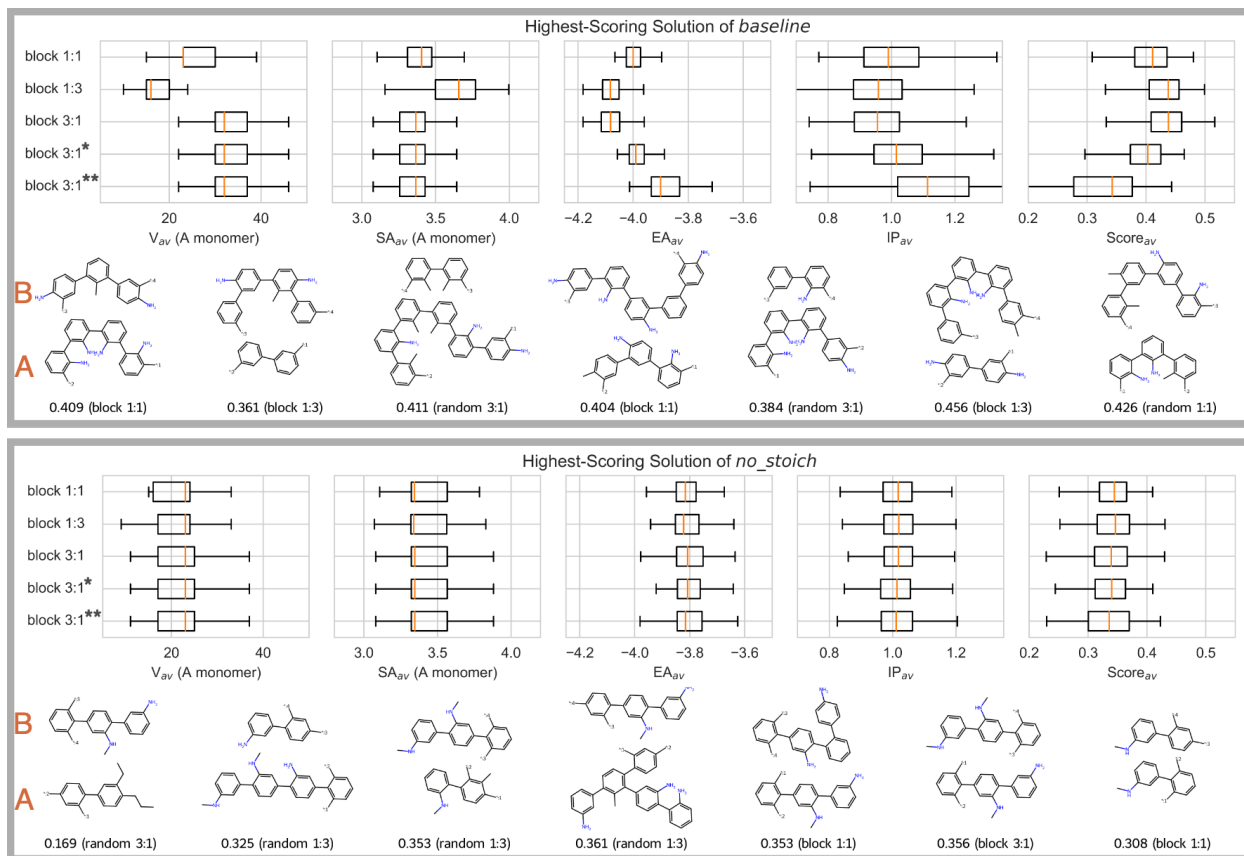


Figure 7: Reported metrics across subsets of polymers generated with specific chain architectures and stoichiometry values for the agent types *baseline* and *no_stoich*, where *no_stoich* does not incorporate stoichiometry in the node and graph embeddings, and *baseline* does. In the figure, we use a notation such as '0.353 (random 1:3)' to indicate that the polymer, with the depicted monomer units above, has a score of 0.353 and a chain architecture 'random' with stoichiometry 1:3. For the selected polymers subsets, the reported metrics include the average number of nodes (V_{av}) and average SA score (SA_{av}) for all A monomers, alongside the average electron affinity (EA_{av}), the average ionization potential (IP_{av}), and average score ($Score_{av}$) for all polymers. The figure showcases a selection of generated polymers for both agents, consistently depicting the B monomers at the top and the A monomers on the bottom. * Evaluated with a 1:1 stoichiometry. ** Evaluated with a 1:3 stoichiometry.

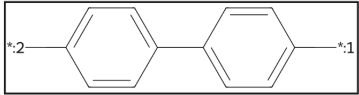
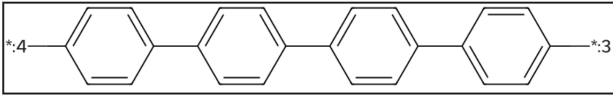
<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"> A Monomer  </div> <div style="text-align: center;"> B Monomer  </div> </div>			
(a)			
Chain Architecture	Stoichiometry	EA	IP
Random	1:1	-2.98	1.34
Block	1:1	-2.94	1.32
Block	3:1	-3.01	1.37
Block	1:3	-2.89	1.28
(b)			
Chain Architecture	Stoichiometry	EA	IP
Random	1:1 (A:A)	-2.92	1.30
Block	1:1 (A:A)	-2.87	1.28
Block	3:1 (A:A)	-2.87	1.28
Block	1:3 (A:A)	-2.87	1.28
(c)			
Subgraphs		SA (A)	SA (B)
As displayed		3.45	2.22
Wildcard atoms (*) turned into Bromine (Br)		1.24	1.06
Wildcard atoms (*) turned into Hydrogen (H)		1.00	1.00

Figure 8: Toy example of a copolymer, featuring two distinct monomer units, that remains invariant when simulating macromolecules with different chain architectures or stoichiometry values, yet exhibits sensitivity in property prediction and SA score calculation in the case study. **(a)** Property prediction varies with chain architecture and stoichiometry. **(b)** This sensitivity is observed only with chain architecture in homopolymer cases. **(c)** SA score calculation shows sensitivity to subtle differences in monomer unit representation, particularly with wildcard atoms influencing scores favorably for larger monomers. Notably, substituting wildcard atoms in the subgraphs with hydrogen or bromine, used for polymerization in the case study, reduces the overall SA scores and observed disparities between monomer units.

One explanation could be that the policy takes advantage of the method used by the property predictors to form the graph embedding. Specifically, the property predictors scale the node embeddings, formed by the wD-MPNN, based on the relative frequency of the monomer units to which the nodes belong, before aggregating them into the complete graph embedding (Fig. 4a). When more frequently occurring monomer units are also larger, the average scaling factor for all node embeddings increases. In scenarios where node embeddings are nearly identical, as expected when monomer units exhibit minimal diversity, this tends to predominantly amplify the magnitude of the graph embedding. As a multi-layer perceptron operates on this graph embedding to predict a chemical property, variations in its structure can lead to different outcomes. To address this issue, we propose that future work should use a property predictor that normalizes the scaling factor across all polymer graphs to a fixed mean value.

Large Molecule Size In both the best solution and the best solution of *no_stoich*, the generated molecules exhibit a notably high average number of nodes (V_{av}) (Fig. 7). This characteristic correlates

with higher average scores (Score_{av}) on the Pareto front. We can think of two possible reasons that could explain this observation. Firstly, consider that our model assigns a zero score to all non-unique polymers, penalizing deterministic outcomes (Eq. 26). It is likely that by generating larger polymer graphs, more actions introduce stochasticity, thereby potentially enhancing uniqueness. This is also what we see in Figure 6, where Pareto-optimal solutions with high Score_{av} and large graph size V_{av} tend to have higher uniqueness values. Moreover, small changes to a larger molecule likely have less impact on its properties, thereby preserving high scores.

An alternative explanation revolves around the SA score calculation for individual monomers. In this calculation, we use the representation frequently used for represented repeating units, using wildcard atoms to denote connection points with other monomers. However, in an illustrative example (Fig. 8c), we notice a significant increase in the SA score when wildcard atoms are included. Interestingly, substituting these wildcards with bromide atoms, integral parts of the dibromo monomers that disappear post-polymerization [36], results in a significant SA score reduction. Upon closer inspection, we attribute these differences in SA score to fragment scoring, a key

component of the SA scoring algorithm.

As outlined in Chapter 4, molecules are fragmented, and each fragment is assigned a score based on its frequency in PubChem. Since wildcard atoms are likely infrequent in this dataset, fragments containing them receive higher scores. In our toy example, the impact of wildcard atoms on the SA score of the B monomer is less pronounced compared to the A monomer, due to the relatively larger molecular size of monomer B, which dilutes the influence of the high-scoring fragments (Fig. 8c). We suspect a similar phenomenon in our findings, supported by Figure 7, where larger monomer units exhibit lower SA scores despite their chemical similarity to smaller ones. With the generated chemical structures, polyphenyl chains with frequent amino groups, appearing synthetically accessible, the results imply that the higher average number of nodes in generated polymers might provide a strategic advantage, mitigating the adverse effect of wildcard atoms on SA score calculations.

Selected Agent Figure 9 highlights an agent situated on the Pareto frontier with a Score_{av} of 0.27. This figure effectively illustrates the impact of fine-tuning on the generated polymers by the agent, presenting density functions among molecules generated by the agent for three molecular properties used in scoring polymers. It compares these distributions both before and after pretraining, showcasing the shift in property distributions. We selected this particular agent to balance scoring and diversity. Moreover, as previously discussed, this agent with lower Score_{av} is less likely to take advantage of weaknesses in the scoring function.

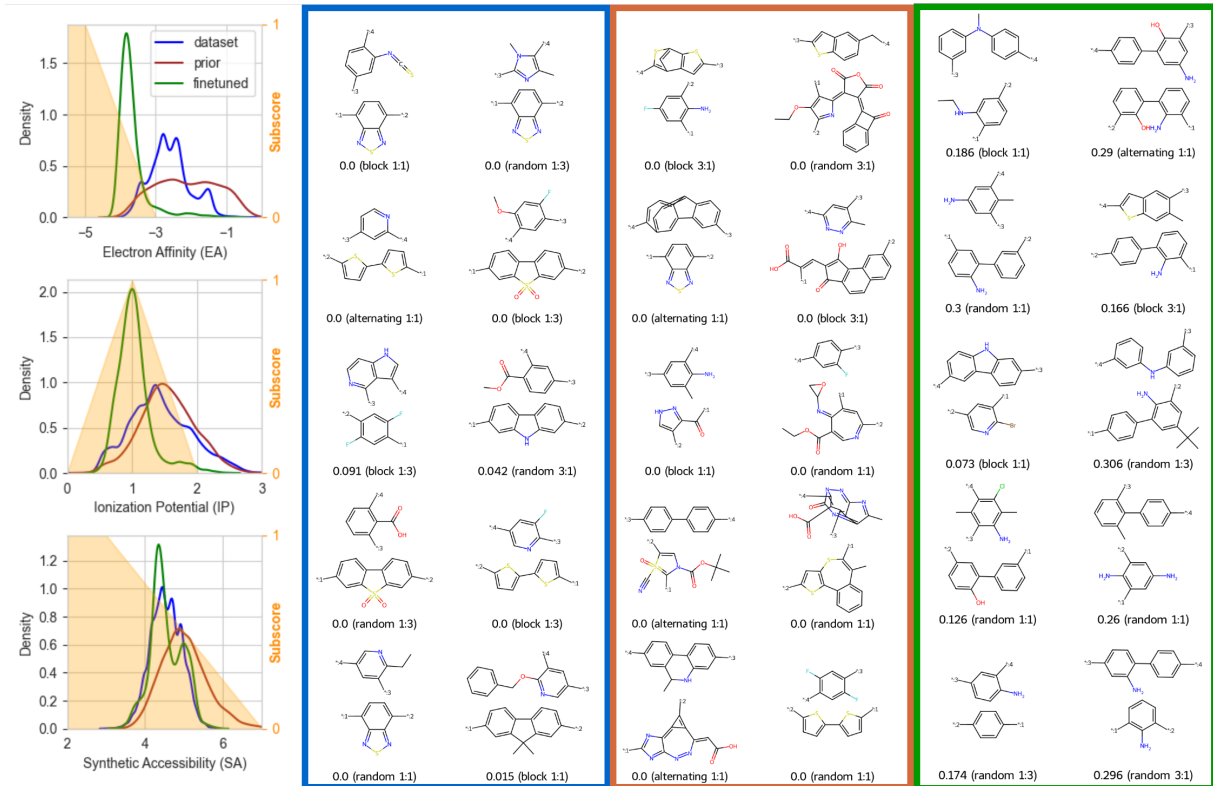


Figure 9: Density functions calculated across three sets of polymers for all three molecular properties used in the scoring of polymers. The three sets of polymers include, with samples visualized from left to right, the $\sim 39K$ polymers in the dataset (Fig. 4b), 2100 molecules generated by the prior agent of *baseline* (Tab. 2), and 2100 molecules generated by the agent after finetuning on the case study objective. In the figure, we use a notation such as '0.353 (random 1:3)' to indicate that the polymer, with the depicted monomer units above, has a score of 0.353 and a chain architecture 'random' with stoichiometry 1:3.

6 Conclusion and Outlook

In conclusion, we have developed a graph-based generative model that can be pretrained to generate molecules similar to those in an expert dataset and finetuned using reinforcement learning to explore more desired regions of the chemical space. This model achieves high validity rates by excluding actions that result in invalid molecules, a key advantage of its graph representation, which also facilitates the incorporation of more detailed information. To validate the effectiveness of our approach, we conducted a case study focused on designing polymers optimized for Hydrogen Evolution Reaction (HER) activity, which measures the production of hydrogen gas in photocatalytic water splitting. Our model successfully navigated the chemical landscape, optimizing polymer designs based on key properties like electron affinity and ionization potential. It also considers the synthetic accessibility of the monomer units, a practical consideration when producing synthetic polymers.

However, our experimental results indicate some nuances in our findings. It appears that high-scoring agents in our case study might be manipulating the optimization process. They tend to generate larger molecules to artificially reduce the predicted synthetic accessibility score of monomer units, which may not align with the true objective. Future work could look into forming more robust predictions on the synthetic accessibility of monomer units to mitigate this issue. Additionally, exploring the integration of polymerization types and resulting functional groups into the scoring system would be advantageous, particularly if the generative model is afforded flexibility in selecting these factors.

Furthermore, the high-scoring agents in our case study adjust the size of monomer units based on their relative abundance in the polymer stoichiometry, likely misleading the used property predictors. These findings underscore the difficulty of developing reliable property predictors for polymer design, a largely unexplored field hampered by a scarcity of empirical data. Future efforts should aim to enhance the robustness of property predictors for polymers, integrating the adjustment proposed in Chapter 5.2.

The design goal in this work seemed to favor smaller repetitive chemical structures, primarily producing polyphenyl chains with limited variations between the monomer units. This approach seemingly made the incorporation of global information across the entire polymer redundant for decision-making. However, the prior was trained on a small and non-

diverse dataset, which might have forced the agent to stay too closely aligned with the dataset, limiting its ability to explore more diverse structures. Additionally, our architecture and cost function potentially influence the optima reached, constraining the agent’s ability to leverage global information effectively.

Future work could explore diverse design tasks to assess the agent’s ability to utilize global information, considering not only the relative abundance of chemical motifs but also their spatial distribution within the polymer. Furthermore, future efforts could look at integrating deep exploration [21], or the combination of curriculum learning and diversity filters [20, 44] to more efficiently explore the chemical space, potentially leading to high-scoring optima that emphasize larger chemical structures. Alternatively, as in the approach by Zhou et al. [24], which does not involve pretraining on a specific dataset or incorporating prior likelihood in the cost function, omitting these constraints could increase flexibility. Considering the aforementioned limitations of copolymer datasets in terms of size and diversity, this approach may facilitate the discovery of a broader range of solutions.

In this study, copolymer design was constrained to manipulating monomer units, with stoichiometry and chain architectures sampled from a pretraining dataset. Another constraint was that the environment only allowed actions that augmented molecular graphs solely with zero-charged atoms, thus limiting the representation of a significant portion of the chemical space. Additionally, the study focused exclusively on linear copolymers, leaving a gap for exploring more complex copolymer structures, such as graft or network copolymers.

Moving forward, future research could enhance the agent’s flexibility by introducing a wider variety of building blocks, potentially including relevant motifs like functional groups to guide molecular generation. Furthermore, expanding the methodology to enable the creation of more complex copolymers from the ground up, including those with high branch factors, presents an opportunity for exploration. However, given the previously described challenges in developing reliable property predictors in this field, we advocate for the design and evaluation of robust scoring functions before progressing to more advanced and flexible generative models.

References

- [1] Aaron M. Virshup, Julia Contreras-García, Peter Wipf, Weitao Yang, and David N. Beratan. Stochastic Voyages into Uncharted Chemical Space Produce a Representative Library of All Possible Drug-Like Compounds. *Journal of the American Chemical Society*, 135(19):7296–7303, May 2013.
- [2] Daniel C. Elton, Zois Boukouvalas, Mark D. Fuge, and Peter W. Chung. Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design & Engineering*, 4(4):828–849, August 2019.
- [3] Stephen Gow, Mahesan Niranjana, Samantha Kanza, and Jeremy G Frey. A review of reinforcement learning in chemistry. *Digital Discovery*, 1(5):551–567, 2022.
- [4] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, July 2018.
- [5] Kianoosh Sattari, Yunchao Xie, and Jian Lin. Data-driven algorithms for inverse design of polymers. *Soft Matter*, 17(33):7607–7622, 2021.
- [6] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes, December 2022.
- [7] Seonghwan Kim, Charles M. Schroeder, and Nicholas E. Jackson. Open Macromolecular Genome: Generative Design of Synthetically Accessible Polymers. *ACS Polymers Au*, 3(4):318–330, August 2023.
- [8] Gabriel Vogel, Paolo Sortino, and Jana Weber. Graph-to-String Variational Autoencoder for Synthetic Polymer Design. In *AI for Accelerated Materials Design - NeurIPS 2023 Workshop*, November 2023.
- [9] Kazuya Hiraide, Kenta Hirayama, Katsuhiko Endo, and Mayu Muramatsu. Application of deep learning to inverse design of phase separation structure in polymer alloy. *Computational Materials Science*, 190:110278, April 2021.
- [10] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction Tree Variational Autoencoder for Molecular Graph Generation, March 2019.
- [11] Miriam Nnadili, Andrew Okafor, Teslim Olayiwola, David Akinpelu, and Jose Romagnoli. Generative AI-Driven Molecular Design: Combining Predictive Models and Reinforcement Learning for Tailored Molecule Generation, November 2023.
- [12] Chiho Kim, Rohit Batra, Lihua Chen, Huan Tran, and Rampi Ramprasad. Polymer design using genetic algorithm and machine learning. *Computational Materials Science*, 186:110067, January 2021.
- [13] Ruimin Ma, Hanfeng Zhang, and Tengfei Luo. Exploring High Thermal Conductivity Amorphous Polymers Using Reinforcement Learning. *ACS Applied Materials & Interfaces*, 14(13):15587–15598, April 2022.
- [14] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):48, September 2017.
- [15] Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W Coley. Sample Efficiency Matters: A Benchmark for Practical Molecular Optimization.
- [16] 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, February 1988. doi: 10.1021/ci00057a005.
- [17] 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, 2018.
- [18] Tzzy-Shyang Lin, Connor W. Coley, Hide-nobu Mochigase, Haley K. Beech, Wencong Wang, Zi Wang, Eliot Woods, Stephen L. Craig, Jeremiah A. Johnson, Julia A. Kalow, Klavs F. Jensen, and Bradley D. Olsen. BigSMILES: A Structurally-Based Line Notation for Describing Macromolecules. *ACS Central Science*, 5(9):1523–1531, September 2019.
- [19] Ludwig Schneider, Dylan Walsh, Bradley Olsen, and Juan de Pablo. Generative BigSMILES: An Extension for Polymer Informatics, Computer Simulations & ML/AI, August 2023.

- [20] Maranga Mokaya, Fergus Imrie, Willem P. van Hoorn, Aleksandra Kalisz, Anthony R. Bradley, and Charlotte M. Deane. Testing the limits of SMILES-based de novo molecular generation with curriculum and deep reinforcement learning. *Nature Machine Intelligence*, 5(4):386–394, April 2023.
- [21] Luca A. Thiede, Mario Krenn, AkshatKumar Nigam, and Alán Aspuru-Guzik. Curiosity in exploring chemical spaces: Intrinsic rewards for molecular reinforcement learning. *Machine Learning: Science and Technology*, 3(3):035008, July 2022.
- [22] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alán Aspuru-Guzik. Self-Referencing Embedded Strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, December 2020.
- [23] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [24] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports*, 9(1):10752, July 2019.
- [25] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs, 2022.
- [26] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph Networks for Molecular Design, August 2020.
- [27] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010. PMID: 20426451.
- [28] Daniel S. Wigh, Jonathan M. Goodman, and Alexei A. Lapkin. A review of molecular representation in the age of machine learning. *WIREs Computational Molecular Science*, 12(5):e1603, 2022.
- [29] Cheng Yan and Guoqiang Li. The Rise of Machine Learning in Polymer Discovery. *Advanced Intelligent Systems*, 5(4):2200243, 2023.
- [30] 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.
- [31] Matteo Aldeghi and Connor W. Coley. A graph representation of molecular ensembles for polymer property prediction. *Chemical Science*, 13(35):10486–10498, 2022.
- [32] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks, October 2017.
- [33] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation, February 2019.
- [34] Zhaocheng Zhu, Chence Shi, Zuobai Zhang, Shengchao Liu, Minghao Xu, Xinyu Yuan, Yang Zhang, Junkun Chen, Huiyu Cai, Jiarui Lu, Chang Ma, Runcheng Liu, Louis-Pascal Khon-neux, Meng Qu, and Jian Tang. TorchDrug: A Powerful and Flexible Machine Learning Platform for Drug Discovery. *ArXiv*, February 2022.
- [35] Rocío Mercado, Esben J. Bjerrum, and Ola Engkvist. Exploring Graph Traversal Algorithms in Graph-Based Molecular Generation. *Journal of Chemical Information and Modeling*, 62(9):2093–2100, May 2022.
- [36] Yang Bai, Liam Wilbraham, Benjamin J. Slater, Martijn A. Zwiijnenburg, Reiner Sebastian Sprick, and Andrew I. Cooper. Accelerated Discovery of Organic Polymer Photocatalysts for Hydrogen Evolution from Water through the Integration of Experiment and Theory. *Journal of the American Chemical Society*, 141(22):9063–9071, June 2019.
- [37] Greg Landrum, Paolo Tosco, Brian Kelley, Ricardo Rodriguez, David Cosgrove, sriniker, Riccardo Vianello, gedeck, NadineSchneider, Gareth Jones, Eisuke Kawashima, Dan Nealschneider, Andrew Dalke, Brian Cole, Matt Swain, Samo Turk, Aleksandr Savelev, Alain Vaucher, Maciej Wójcikowski, Ichiru Take, Vincent F. Scalfani, Rachel Walker, Kazuya Ujihara, Daniel Probst, guillaume godin, Axel Pahl, Juuso Lehtivarjo, Francois Berenger, jasondbiggs, and strets123. rdkit/rdkit: 2024.03.1 (q1 2024) release, March 2024.

- [38] 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):8, June 2009.
- [39] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. PubChem 2023 update. *Nucleic Acids Research*, 51(D1):D1373–D1380, January 2023.
- [40] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, and Regina Barzilay. Analyzing Learned Molecular Representations for Property Prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, August 2019.
- [41] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [42] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.
- [43] Jeff Guo and Philippe Schwaller. Augmented Memory: Capitalizing on Experience Replay to Accelerate De Novo Molecular Design, May 2023.
- [44] Jeff Guo, Vendy Fialková, Juan Diego Arango, Christian Margreitter, Jon Paul Janet, Kostas Papadopoulos, Ola Engkvist, and Atanas Patrónov. Improving de novo molecular design with curriculum learning. *Nature Machine Intelligence*, 4(6):555–563, June 2022.

Appendices

A Supplementary Tables

Table S1: Overview of molecular features included in the node embeddings to form the feature matrix F . These features are extracted using RDKit. For the agent type `only_symbol`, only the atomic symbol feature is used. * The symbol encodes for one of $\{C, N, O, F, S, Cl, Br, I, *\}$

Feature	Size	Type
Symbol*	9	One-hot
Chiral Tag	4	One-hot
Total Degree	8	One-hot
Number of Implicit Hydrogens	7	One-hot
Number of Radical Electrons	8	One-hot
Hybridization Type	9	One-hot
Is Aromatic	1	Boolean
Is in Ring	1	Boolean
	48	

Table S2: Fixed hyperparameter values during pretraining and finetuning.

Parameter	Value
R-GCN hidden layers	[256, 256, 256, 256]
R-GCN activation function	ReLU
m_f (first node MLP) hidden layers	[128, 1]
m_s (second node MLP) hidden layers	[128, 1]
m_e (edge type MLP) hidden layers	[128, 3]
m_t (stop signal MLP) hidden layers	[128, 2]
m_f, m_s, m_e, m_t activation function	Tanh
Agg (aggregation function)	Mean
Adam epsilon	1×10^{-8}
Adam beta parameters	(0.9, 0.999)
Adam AMSGrad	False

Table S3: Mean Negative Log-Likelihood (NLL) training loss for different agent types evaluated at 800, 1600, and 2400 learning steps. Each value represents the average loss across three replications for weight decay values of 0, 0.001, and 0.0001.

Agent Type (Learning Steps)	Mean NLL Loss		
	Weight Decay 0	Weight Decay 0.0001	Weight Decay 0.001
<i>baseline</i> (Step 800)	1.386	1.426	1.624
<i>baseline</i> (Step 1600)	1.258	1.291	1.504
<i>baseline</i> (Step 2400)	1.189	1.253	1.447
<i>only_symbol</i> (Step 800)	1.660	1.713	2.001
<i>only_symbol</i> (Step 1600)	1.478	1.531	1.751
<i>only_symbol</i> (Step 2400)	1.387	1.448	1.689
<i>gta_mon</i> (Step 800)	1.181	1.238	1.377
<i>gta_mon</i> (Step 1600)	1.083	1.127	1.302
<i>gta_mon</i> (Step 2400)	1.034	1.094	1.273
<i>no_stoich</i> (Step 800)	1.357	1.409	1.543
<i>no_stoich</i> (Step 1600)	1.227	1.288	1.458
<i>no_stoich</i> (Step 2400)	1.179	1.223	1.414
<i>obs_focus</i> (Step 800)	1.359	1.407	1.627
<i>obs_focus</i> (Step 1600)	1.212	1.259	1.487
<i>obs_focus</i> (Step 2400)	1.140	1.201	1.440
<i>obs_graph</i> (Step 800)	1.491	1.537	1.734
<i>obs_graph</i> (Step 1600)	1.320	1.388	1.571
<i>obs_graph</i> (Step 2400)	1.226	1.305	1.535