DEEPSETS GFLOWNETS FOR SUBSET SELECTION AND AN APPLICATION TO COMBINATORIAL CHEMISTRY

Anonymous authors
Paper under double-blind review

ABSTRACT

DNA-encoded libraries (DELs) are combinatorial chemical libraries screened against a protein target in a single pooled experiment, avoiding the high costs associated with evaluating molecules one-by-one. We formulate DEL design as a combinatorial optimization problem which involves choosing sets of building blocks which produce many high-quality combinations. Generative Flow Networks (GFlowNets; Bengio et al. (2023)), which learn to sample compositional objects proportional to a reward, are well-suited to this problem, generating diverse, high-reward libraries rather than converging to a single optimum. However, applying GFlowNets to real-world DELs presents two open challenges: 1) DELs must be represented in a way that preserves the chemical structure of the underlying building blocks for efficient learning; 2) the cost of scoring candidate libraries must remain bearable at real-world scales. To address these obstacles, we leverage DeepSets representations to amortize the computation cost of combinatorial rewards and obtain a structure-aware GFlowNet architecture suitable for the optimization of combinatorial set functions on large structured environments. Applied to DEL design, our approach scales to libraries containing 125 billion elements, three orders of magnitude beyond prior work.

1 INTRODUCTION

The design of small molecules with desired properties is a central problem in drug discovery. The goal is rarely to find a single optimal candidate, but rather a diverse pool of candidates, as 10^5 to $> 10^6$ molecules are typically screened in an industrial drug discovery project to yield one or two candidate molecules (Hughes et al., 2011). Generative Flow Networks (GFlowNets; Bengio et al. (2021)) are a natural fit for this setting, sampling compositional objects in proportion to some reward rather than simply maximizing it, which avoids mode collapse and supports the multi-modal objective landscape of molecular generation. In recent years, GFlowNets have been successfully applied to the generation of diverse, high-reward small molecules (Kim et al., 2026; Koziarski et al., 2024b; Bengio et al., 2021). However, generating and evaluating individual molecules is slow and costly, as each candidate must be synthesized and assayed one at a time, limiting such pipelines to screening collections that cover only a fraction of drug-like chemical space (Goodnow et al., 2017).

Combinatorial chemistry addresses the constraints of sequential molecule-by-molecule design, allowing for the synthesis of libraries containing up to trillions of compounds through simple split-and-pool procedures (Gerry et al., 2019). DNA-Encoded Libraries (DELs) extend this idea by tagging each molecule with a unique DNA barcode, allowing large libraries to be screened simultaneously. In a DEL pulldown experiment, the library is exposed to an immobilized protein; non-binding molecules are washed away, leaving high-affinity binders enriched in the remaining pool, which can then be identified through sequencing (Neri & Lerner, 2018). A challenge of this approach is that pulldown data is noisy, and grows noisier with library size. A fixed sequencing depth spread across more compounds results in fewer reads per molecule (Kalinic & Kómár, 2020; Lim et al., 2022). As a result, targeted applications often favor smaller, purpose-built libraries, which typically yield a cleaner signal at the cost of scale. This tradeoff is well-documented. Library enrichment quality has been shown to degrade with scale, with larger libraries exhibiting lower signal-to-noise ratios and higher false-negative rates than smaller focused ones (Satz, 2018; Satz et al., 2017).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Designing such a library is a combinatorial generation problem: DELs are typically constructed by combining *building blocks* drawn from several predefined collections, called *pools*. It is therefore a natural candidate problem for GFlowNets (Koziarski et al., 2024a). Scaling this approach to real-world DELs, however, presents two open challenges: 1) DELs must be represented in a way that preserves the chemical structure of the underlying building blocks for efficient learning; 2) the cost of scoring candidate libraries must remain bearable at real-world scales. Koziarski et al. (2024a) address the first by grouping building blocks by chemical similarity, and the second by replacing the oracle with an inexpensive learned proxy, evaluated over the full library ahead of training. However, scoring a single library draws its cost from two sources: the expense of evaluating the element-level oracle on an individual compound, and the number of compounds that must be evaluated (which grows as the product of the pool sizes). While the proposed approach is feasible for small libraries, it leaves the second source of evaluation cost unaddressed, as DELs typically range from hundreds of millions to trillions of compounds (Goodnow et al., 2017).

These obstacles point to a common question, which motivates the rest of this paper: "*How do we generate diverse, high-reward subsets from pools of building blocks without exhaustive scoring?*" We answer with a structure-aware GFlowNet and an online learning framework that makes expensive reward proxies tractable at scale, and use it to design targeted DELs drawn from a top-level library containing over **125 billion** candidates, three orders of magnitude beyond previous work.

In addressing the general problem of combinatorial subset optimization, of which DEL design is a prime example, we leverage DeepSets (Zaheer et al., 2017), an architecture that represents a set through a permutation-invariant aggregation over embeddings of its elements. Despite its success in supervised settings, DeepSets has not, to our knowledge, been used in set-generating GFlowNets. Prior work has instead relied on multi-hot representations, including the GFlowNet of Koziarski et al. (2024a) (DEL-GFN) and its hierarchical variant (H-DEL-GFN), as well as the set-selection GFlowNet of Evmorfos et al. (2023) (GFLOW-SS) that discard the structural information DeepSets preserves. Our contributions are as follows:

① **A DeepSets-based scheme for the amortization of combinatorial rewards, and its application to DEL scoring.** When evaluating rewards which are expressed in terms of combinatorial structures on sets of building blocks, the cost of combinatorial explosion imposes a large cost on exact reward evaluation, as the set of building blocks grows. We show how DeepSets can be used to amortize this cost, providing a source of non-enumerative proxies for expensive combinatorial rewards.

② **DeepSets-GFN: A structure-aware flow model for set GFlowNets.** A DeepSets-based GFlowNet architecture for subset selection that remains sensitive to the features of individual building blocks in both its state and action representations.

③ **A target-oriented DEL generation pipeline.** Combining the above contributions, we introduce a three-variable DeepSets-GFN for DEL construction, rewarded by the predictive multivariate DeepSets model from our first contribution. We use it to generate high-quality DELs targeted towards four different protein targets, including ClpP, Mpro, sEH, and TBLR1.

Empirically, our proposed framework outperforms existing set-generating GFlowNets on DEL design tasks. Across four target proteins and three environment sizes, we discover sub-libraries with top-1 docking rewards that match or exceed both GFlowNet baselines in 11 of 12 settings, and the advantage grows significantly with environment size, as both GFlowNet baselines degrade in the largest environment. We further demonstrate that pre-training across a range of reward thresholds provides a way to rapidly adjust the threshold on-the-fly, helping to capture a greater range of favorable score distributions. Finally, we show that online learning of combinatorial rewards with DeepSets-GFN results in an increasingly accurate reward proxy, which is able to rapidly and accurately score sets of building blocks even well outside of the initial training distribution.

2 Problem Setting

We introduce the general problem of combinatorial subset selection and show that DEL design can be formulated under that setting. We then show how this problem can be formulated as sequential decision-making and highlight the specific challenges.

Let $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K$ denote K *pools* of available items, called *building blocks*. Let $(B_k \subseteq \mathcal{B}_k)_{k=1}^{K}$ denote subsets of the corresponding pools. Let c denote a combinatorial function that takes subsets

$(B_k \subseteq \mathcal{B}_k)_{k=1}^K$ as input and produces a set of elements. The elements of $c(B_1, \dots, B_K)$ are called *c-structures* on $B_1, \dots, B_K$ (Bergeron et al., 1998). Let $\mathcal{U}$ denote the space of c-structures and let $f : \mathcal{U} \to \mathbb{R}$ denote a real-valued *scoring function* of c-structures. Given constraints $(\ell_k \leq |\mathcal{B}_k|)_{k=1}^K$, the objective is to find subsets $(B_k \subseteq \mathcal{B}_k : |B_k| \leq \ell_k)_{k=1}^K$ such that the resulting set $c(B_1, \dots, B_K) \subseteq \mathcal{U}$ contains many c-structures u with low score $f(u)$ [*].

Application to DEL Design DELs are combinatorial libraries in which small-molecule building blocks are joined across several cycles and tagged with unique DNA barcodes, producing large numbers of candidate molecules that can be synthesized and screened simultaneously (Satz et al., 2022). Let $\mathcal{M}$ denote the molecular space. DELs are equipped with a *reaction map* $\mathfrak{R} : \mathcal{B}_1 \times \cdots \times \mathcal{B}_K \to \mathcal{M}$ which combines K building blocks $(b_k \in \mathcal{B}_k)_{k=1}^K$, one from each pool, into an element $\mathfrak{R}(b_1, \dots, b_K)$ of the molecular space $\mathcal{M}$. The c function corresponds to the K-Cartesian product $c(B_1, \dots, B_K) := B_1 \times \cdots \times B_K$, and a c-structure is a tuple $u = (b_1, \dots, b_K)$ with $b_k \in B_k$ for all $k \in [K]$. Applying the reaction map to every element $u \in c(B_1, \dots, B_K)$ results in a DEL $L = \mathfrak{R}(c(B_1, \dots, B_K))$. The scoring function f reflects some desirable property. For example, $f(u)$ can capture the docking affinity of the molecule $\mathfrak{R}(u)$ on a specific target, with a lower score indicating higher affinity.

Sequential decision-making perspective The combinatorial subset selection problem can be formulated as a sequential decision-making problem, where a compositional object is built by constructing item-by-item sets of building blocks $B_k \subseteq \mathcal{B}_k$ from each pool k. This environment can be represented as a directed acyclic graph (DAG) described by a state-space $\mathcal{S}$ and an action-space $\mathcal{A}$. A state $s \in \mathcal{S}$ denotes a specific collection of sets $s = (B_k)_{k=1}^K$, and an action $a \in \mathcal{A}$ consists in adding a new building block from any pool k to the corresponding set B_k. We denote $\mathcal{A}_k(s) := \mathcal{B}_k \setminus B_k$ the building blocks of the k-th pool that are not yet contained in the set B_k of state s. The complete available action space in state s is $\mathcal{A}(s) = \sqcup_k \mathcal{A}_k(s)$. The initial state $s_0 := (\emptyset)_{k=1}^K$ contains K empty sets. The action $a \in \mathcal{A}_k(s)$ causes a transition from state $s = (B_1, \dots, B_k, \dots, B_K)$ to $s' = (B_1, \dots, B_k \cup \{a\}, \dots, B_K)$. A terminating state $x \in \mathcal{X}$ is reached only once every set of building block has reached its predefined target size, i.e., $\mathcal{X} := \{(B_1, \dots, B_K) : B_k \subseteq \mathcal{B}_k, |B_k| = \ell_k \; \forall k \in [K]\}$. A *trajectory* $\tau = (s_0, s_1, \dots, s_T)$ corresponds to the sequence of states encountered from the initial state to a terminating state. All trajectories have length $T = \ell_1 + \cdots + \ell_K$. The objective is to discover terminating states $x \in \mathcal{X}$ such that $c(x)$ contains many c-structures $u \in \mathcal{U}$ with low score $f(u)$.

Challenges In addition to the typical challenges of combinatorial optimization, combinatorial subset selection adds a layer of considerable computational cost when $|c(B_1, \dots, B_K)|$ grows much faster than $\ell_1 + \cdots + \ell_K$, as is the case for most commonly encountered combinatorial species such as permutations, partitions, or colorings (Garey & Johnson, 1979). As a result, even the *evaluation*, let alone the optimization, of a score function on every c-structure rapidly becomes problematic.

3 Background and Related Work

3.1 Generative Flow Networks

GFlowNets (or GFNs) are a class of probabilistic models trained to sample compositional objects over a discrete sample space $\mathcal{X}$. Let $\mathcal{R} : \mathcal{X} \to \mathbb{R}$ denote a *reward function*, the objective of GFlowNets si to sample each object $x \in \mathcal{X}$ with probability $p(x)$ proportional to its reward $\mathcal{R}(x)$ (Bengio et al., 2021). GFlowNets operate on a DAG over a set of states $\mathcal{S}$, and objects are constructed step-by-step, starting from state $s_0 = \emptyset$ and following the edges of the DAG until a complete object is reached. Given current state $s \in \mathcal{S}$ and next state $s' \in \mathcal{S}$, GFlowNets parameterize a forward policy $P_F(s'|s)$ and a backward policy $P_B(s|s')$. The forward policy is a distribution over the children of each state s; sampling from it, starting from s_0, generates a trajectory $\tau = (s_0, s_1, \dots, s_T)$ ending in some terminal state $x \in \mathcal{X}$. The backward policy is a distribution over the parents of each state s'; starting from x, it generates trajectories in reverse, back to the initial state s_0. Both factorize over the transitions (or edges) along a trajectory τ:

[*] If the desired structures are instead those with high value of f, we may replace f by $-f$. In this work, the desirable structures have low value of f.

$P_F(\tau) = \prod_{t=0}^{T-1} P_F(s_{t+1}|s_t)$ and $P_B(\tau|x) = \prod_{t=1}^{T} P_B(s_{t-1}|s_t)$. As a state may be reached by several unique paths in the DAG, many trajectories can end in the same terminal state x, and the probability of sampling x is the sum of $P_F(\tau)$ over all of those paths. Training consists of learning both P_F and P_B, parameterized by a neural network θ, such that a consistency constraint holds across trajectories. Several such constraints have been proposed; we adopt Trajectory Balance (TB), which enforces a global constraint over trajectories (Malkin et al., 2022):

$$\mathcal{L}_{TB}(\tau) = (\log Z_\theta + \log P_F(\tau;\theta) - \log R(x) - \log P_B(\tau \mid x;\theta))^2. \tag{1}$$

Minimizing $\mathcal{L}_{TB}$ to zero over all trajectories guarantees $p(x) \propto \mathcal{R}(x)$. When P_B is deterministic, TB reduces to path consistency learning (Nachum et al., 2017), as shown by Deleu et al. (2024).

GFlowNets for subset selection When the objective is submodular, greedy selection is the natural choice, as it is provably within a constant factor of the optimal solution (Nemhauser et al., 1978). These guarantees weaken as the objective departs from submodularity, as measured by the *submodularity ratio* formalized by Das & Kempe (2011). GFlowNets offer an alternative that avoids such structural assumptions, which makes them a natural candidate for tackling subset selection problems where no such decomposition exists.

An extensive application of GFlowNets to subset selection targets graph combinatorial problems whose solutions are vertex subsets such as maximum cut and the maximum independent set (Zhang et al., 2023). There, the objective is either the size of the selected set (where each element contributes equally), or a submodular function, in which training also depends on this decomposability as it scores partial solutions with intermediate rewards. In contrast, limited work considers objectives that depend on how selected elements interact. In sparse sensor array selection, the performance of a subarray depends on how the selected sensors jointly sample the array aperture rather than on each sensor's isolated contribution, requiring selection methods that avoid submodularity-based greedy heuristics (Evmorfos et al., 2023). DEL design shares this structure; the quality of a library depends on how building blocks combine across synthesis cycles rather than on their individual properties. We examine GFlowNet-based approaches to DEL design below.

GFlowNets for DEL design GFlowNets have been widely applied to the task of molecular generation, where trajectories correspond to sequential graph- or fragment-based construction of a single compound (Bengio et al., 2021; Jain et al., 2022; Shen et al., 2023). Our setting differs in that rather than sequentially building one molecule atom-by-atom, we construct a *subset* of pre-defined building blocks, and the reward is defined jointly over combinations drawn from multiple pools. This subset-generation setting has received comparatively little attention in the GFlowNet literature. Evmorfos et al. (2023) introduce GFLOW-SS, representing candidate subsets as binary selection vectors and trains a GFlowNet whose reward measures how well the aggregate output of a subset matches a target signal, applying the method to sparse antenna array design. Most similar to our work, Koziarski et al. (2024a) propose DEL-GFN, a GFlowNet for generating DELs with high average PPI score, along with a hierarchical variant (H-DEL-GFN) that clusters building blocks to approximate chemical structure. Both GFLOW-SS and DEL-GFN represent states using multi-hot vectors over the building block pool, ignoring the identity and features of individual elements; H-DEL-GFN's clustering-based policy only loosely recovers this structure. Further, these methods are demonstrated only on libraries of at most 1.58 million molecules, small enough to compute and hold in memory the base proxy's values over the entire library, an infeasible strategy at larger scales.

3.2 DeepSets representation

A central characteristic in learning over sets is that sets have no inherent ordering, so the output of a model should not depend on any arrangement of elements at the input. Given a pool $\mathcal{B}$ and a subset $B \subseteq \mathcal{B}$ from that pool. DeepSets (Zaheer et al., 2017) is an elegant machine learning framework for sets, which relies on the universality property of the functional form:

$$g(B) = \rho\left(\sum_{b \in B} \phi(b)\right), \tag{2}$$

which can represent any set function g on a countable pool $\mathcal{B}$. In other words, any set function can be decomposed as a d'-dimensional summed embedding $\phi : \mathcal{B} \to \mathbb{R}^{d'}$ followed by a d'-dimensional

scalar output head $\rho : \mathbb{R}^{d'} \to \mathbb{R}$. When the set B has no additional structure, $\phi(b)$ can be learned as an embedding table; when B has pre-existing embeddings $\varphi(b) = \mathbf{b}$, ϕ can be learned as with a multi-layer perceptron (MLP) $\phi(\mathbf{b})$. We sometimes write $\phi(B)$ for $\sum_{b \in B} \phi(B)$, when there is no confusion.

DeepSets has been widely adopted across domains in which inputs are naturally unordered. Zaheer et al. (2017) demonstrate the applicability of DeepSets in point-cloud classification, estimation of population statistics, anomaly detection within sets of images, and set expansion for text concepts. DeepSets is closely related to PointNet (Qi et al., 2017), which uses the same encode-then-pool structure with max pooling (but using a different pooling operation). Energy Flow Networks (Komiske et al., 2019) adopted the DeepSets decomposition to classify jets from variable-length sets of particles. In reinforcement learning, DeepSets has been used to handle observations containing a variable number of objects; Huegle et al. (2019) used a DeepSets encoder within a Q-network for high-level decision-making decisions in autonomous driving environments. Lee et al. (2018) similarly used a DeepSets encoder in a DQN agent for a sim-to-real robotic sorting task. DeepSets has also served as the foundation for more expressive set architectures, such as the Set Transformer (Lee et al., 2019), designed to model interactions among elements in the input set.

4 GFlowNets for combinatorial subset optimization

In this section, we provide the methods to make GFlowNets efficiently applicable to the combinatorial subset selection problem (Section 2). Instantiating GFlowNets on the corresponding DAG requires defining a reward function that can operate at the level of sets of c-structures. Therefore, we begin by introducing an aggregator function to score sets of c-structures, from our function f which scores individual c-structures. Then, since our aggregation still requires computing the scores of all c-structures in $c(x)$, we show how to leverage DeepSets to amortize this effect at the cost of some overhead. As a result, we obtain an affordable proxy for our expensive aggregator set function. Next, on the generative side, we describe how DeepSets can also be used as a set-friendly, structure-aware GFlowNet architecture suitable for the optimization of a combinatorial set function on a large structured environment.

4.1 Soft-count aggregation for combinatorial subset selection problems

Recall that for every $x = (B_1, \ldots, B_K) \in \mathcal{X}$, we have a set of c-structures $c(x)$, and a scoring function f for individual c-structures $u \in c(x)$. We need to derive a reward function $\mathcal{R} : \mathcal{X} \to \mathbb{R}$ from the scores $f(u)$ of all c-structures $u \in c(x)$. A naive way would be to count c-structures whose f-score falls below a specified threshold λ. However, integer-valued signals provide poor signal for learning; for sufficiently small values of λ, one might even remain stuck in purgatory forever, encountering only libraries with $\mathcal{R} = 0$. In order to address this, we propose to use a *soft-counting function*, which provides a richer learning signal than the hard count:

$$r_{\lambda,h}(x) = 1 + \sum_{u \in c(x)} \sigma\left(\frac{\lambda - f(u)}{h}\right), \qquad h > 0, \quad \lambda \in \mathbb{R} \tag{3}$$

where $\sigma(y) = (1 + e^{-y})^{-1}$ is the sigmoid function, λ is a threshold value, and $h > 0$. Up to scaling factor and Laplace smoothing, the soft count is the cumulative density function (CDF) of a smoothed kernel estimator of the empirical distribution $|c(x)|^{-1} \sum_{u \in c(x)} \delta_{f(u)}$, using a logistic kernel of width h. The following Propositions justifies the use of $r_{\lambda,h}(x)$ as a drop-in replacement for the hard count. The proof can be found in Appendices A.1 and A.2.

Proposition 1 (Sandwich inequality) *For every $\delta > 0$, with $N(\lambda) = \{u \in c(x) | f(u) \leq \lambda\}$ and $\epsilon_\delta = \sigma(-\delta/h) = (1 + e^{-\delta/h})^{-1}$,*

$$(1 - \epsilon_\delta) N(\lambda - \delta) \;\leq\; r_{\lambda,h}(x) \;\leq\; N(\lambda + \delta) + n\epsilon_\delta.$$

Indeed, the sandwich inequality implies that by optimizing the soft count, we are indirectly optimizing the hard count, with both objectives coinciding as $h \to 0$. The next proposition shows that, as a function of λ, the soft count captures the distribution of F scores.

Proposition 2 (Faithfulness of soft-count) *The map* $\sum_{u \in c(x)} \delta_{f(u)} \mapsto \sum_{u \in c(x)} \sigma\left(\frac{\lambda - f(u)}{h}\right)$ *from distributions to continuous functions of* λ *is injective.*

The analogous proposition for the hard count is the trivial statement that the density function can be recovered from the cumulative density function; this proposition shows that soft-counts are also sufficiently informative to capture the distribution of f scores.

4.2 Amortizing combinatorial explosion with DeepSets

Attempting to use the soft-count (Equation 3) directly as the reward function $\mathcal{R}$ would require computing the score of all individual c-structures in $c(x)$ for every visited terminating state $x \in \mathcal{X}$, which requires $|c(x)| \gg \ell_1 + \cdots + \ell_K$ function calls. We propose to amortize this cost in two steps. Following Koziarski et al. (2024a), we first train a less expensive learned proxy $\hat{f}$ to query in replacement of the true score function f. We denote $\hat{r}_{\lambda,h}$ the resulting approximation of the soft-count function (Equation 3). Although $\hat{r}_{\lambda,h}$ is much less expensive than $r_{\lambda,h}$, it is still a combinatorial set function whose complexity scales in $|c(x)|$. We tackle the cost of combinatorial explosion by introducing an amortization scheme for combinatorial set functions based on DeepSets. We begin by generalizing DeepSets to the multi-pool setting.

Proposition 3 (Multi-variate DeepSets) *Given countable pools* $\mathcal{B}_1, \ldots, \mathcal{B}_K$ *and a multi-variate set function* g*, there exist embeddings* $\phi_k : \mathcal{B}_K \to \mathbb{R}^d$ *and a scalar output head* $\rho : \mathbb{R}^d \to \mathbb{R}$*, such that*

$$g(x) = \rho\left(\sum_{b \in B_1} \phi_1(b), \ldots, \sum_{b \in B_K} \phi_K(b)\right), \tag{4}$$

for every $x = (B_1, \ldots, B_K)$.

The proof can be found in Appendix A.3. Proposition 3 justifies the endeavor of jointly *learning* the $K+1$ functions ρ and ϕ_k, each with its own set of parameters θ_k. We write θ for the disjoint union of these $K+1$ sets of parameters, and $\hat{g}_\theta = (\rho_\theta, \phi_{1;\theta}, \ldots, \phi_{K;\theta})$ for the DeepSets model.

Our combinatorial amortization scheme is based on the observation that evaluating g using the form 4 requires only $\ell_1 + \cdots + \ell_K$ calls to the ϕ_k's. Amortizing the computation of a combinatorial set function g with multi-variate DeepSets therefore takes the form of a two-step process:

Step 1. Construct a dataset $\mathcal{D}_g$ of values $(x, g(x))$, for example by sampling $x = (B_1, \ldots, B_K)$ at random, and evaluating g. All else being equal, this step incurs a cost which scales with $|c(x)|$.

Step 2. Train a DeepSets predictive model $\widehat{g}_\theta \approx g$ on $\mathcal{D}_g$, with parameters $\theta = (\theta_\rho, \theta_{\phi,1}, \ldots, \theta_{\phi,K})$. The cost of the resulting function now scales in $\sum_k \ell_k$, all else being equal.

We note that, despite being in $O(|c(x)|)$, the overhead step is highly parallelizable.

4.3 DeepSets-GFNs

We instantiate GFlowNets on the DAG that describes the combinatorial subset selection problem (Section 2). Suppose that $\mathcal{R}(x) = \mathcal{R}(B_1, \ldots, B_K)$ is a reward function. In order to capture its structure , we leverage the multi-variate DeepSets representation (Proposition 3). More specifically, in order to scale to large number of available of actions (here corresponding to all building blocks contained in the pools), we adopt a state-action flow model (Seo et al., 2025). We suppose that building blocks are equipped with a representation $\mathbf{b}$. The generative model thus takes as inputs representations of both a state s and action a, and outputs a scalar, which is the estimated log-flow through action a in state s. The state-action flow model makes it possible to restrict flow evaluation to selected subsets of actions; in turn, this action subsampling scheme makes it possible to work with arbitrarily large action spaces without significant memory or compute requirements. Since any given action may not be sampled often, action subsampling in a large action space also acts as a form of regularization or forced exploration.

Recall that in the combinatorial subset selection problem, a state is given by a tuple $s = (B_1, \ldots, B_K)$ of partially constructed sets, and $a \in \mathcal{A}_k(s)$ denotes an available actions from the

k-th pool in state s, provided that $|B_k| < \ell_k$. The DeepSets-GFN architecture expresses the log-flow through action a in state s as

$$\log \mathcal{F}_\psi(s, a) = \rho_\psi \left(\sum_{b \in B_1} \phi_{1;\psi}(\mathbf{b}), \ldots, \sum_{b \in B_K} \phi_{K;\psi}(\mathbf{b}), \tilde{\phi}_{k;\psi}(\mathbf{a}) \right), \tag{5}$$

Like θ in Section 4.2, ψ is the disjoint union of the $K+1$ sets of parameters, plus the learnable constant $\mathrm{Log}Z$. The embedding $\tilde{\phi}_{k;\psi}(\mathbf{a})$ of action a is simply $\phi_{k;\psi}$ padded with $(K-1)$ d-dimensional zero vectors: $\tilde{\phi}_{k;\psi}(\mathbf{a}) = (\mathbf{0}_d, \ldots, \underbrace{\phi_{k;\psi}(\mathbf{a})}_{k^{\text{th}}\text{pool}}, \ldots, \mathbf{0}_d)$.

By taking into account not only the representation of the $(B_1, \ldots, B_K)$ selected up to now, but also the representation of the considered action a, this architecture is well-placed to leverage the pre-existing representations when modeling the flow.

5 Application to DEL design

We showcase the benefits of the proposed methods on the specific problem of designing DELs optimized for docking on four different protein targets (Mpro, TBLR1, ClpP, and sEH; see Appendix B for motivation). More specifically, we consider a top-level library constructed with $K = 3$ subsets taken from a single pool $\mathcal{B} = \mathcal{B}_1 = \mathcal{B}_2 = \mathcal{B}_3$. Applying the reaction map $\mathfrak{R}$ to a c-structure (a triple (b_1, b_2, b_3) results in a molecule called a *trimer*. The score function $f(u)$ of some c-structure u corresponds to the docking score of the associated trimer $\mathfrak{R}(u)$ according to Autodock Vina (v1.2.7).

We apply the proposed amortization scheme (Section 4.2) to the logarithm of the soft-count function (Equation 3). First, for each target, we train a proxy $\hat{f}$ on a dataset of $500K$ randomly selected trimers docked with Autodock Vina. Then, as in Koziarski et al. (2024a), we replace f with $\hat{f}$ in the definition of r, obtaining a function that we call $\tilde{r}$, which is much less expensive but still a combinatorial set function. We then apply the combinatorial amortization scheme to the function $\tilde{G} := \log \tilde{r}$, obtaining a 3-variable DeepSets proxy $\hat{G}$. Thus, we have

$$\hat{G} \approx \tilde{G} = \log \tilde{r} \approx \log r.$$

To train $\hat{G}$, we sample triplets of building blocks uniformly at random and $\tilde{G} = \log \tilde{r}_{\lambda,h}$, constructing a dataset $\mathcal{D}_{\hat{G}}$, on which we train $\hat{G} = \hat{G}_\theta$. This final proxy is used to provide the signal throughout all experiments, with the GFlowNet reward being $\mathcal{R}(x) = \exp(\hat{G}(x)) \approx \log r(x)$.

5.1 DEL Optimization

Our first experiment compares the performance of DeepSets-GFN applied to DEL optimization against existing methods, namely the DEL-GFN and H-DEL-GFN of Koziarski et al. (2024a), Markov Chain Monte Carlo (MCMC), and random sampling. We train two versions of DeepSets-GFN: one which uses a subsampling rate of 1, and the other a subsampling rate of 0.02. Each method selects $10 \times 10 \times 10$ DELs, had a budget of $15K$ reward calls, and the results are averaged over 3 seeds. The environments increase in difficulty with the size of each building block pool: 200 (small), 1000 (medium), and 5000 (large), with building blocks held constant across targets. Parameter counts for the neural models were equalized within 1% tolerance across each environment and objective. We used a threshold $\lambda = -9$ and width $h = 1$ across all targets and environments.

Figure 1 reports the top-1 docking scores and diversity across all four targets and all three environment sizes. We observe that DeepSets-GFN outperforms the other approaches in terms of rewards in 11 of 12 environments, especially as the environment grows; in all cases it lies on the Pareto front of the (diversity, reward) metrics. As observed by (Seo et al., 2025), subsampling improved diversity at the cost of comparatively little reward. These results show that DeepSets-GFN is able to leverage building block structure effectively. Detailed results can be found in Table E.2 and Table E.2. Without surprise, maximally exploiting the structure with a subsampling rate of 1 also leads to a decrease in diversity, as sampling is now better informed. This motivates further work to improve diversity in a meaningful way, but shows that the subsampling rate provides a useful knob for adjusting the reward/diversity tradeoff.

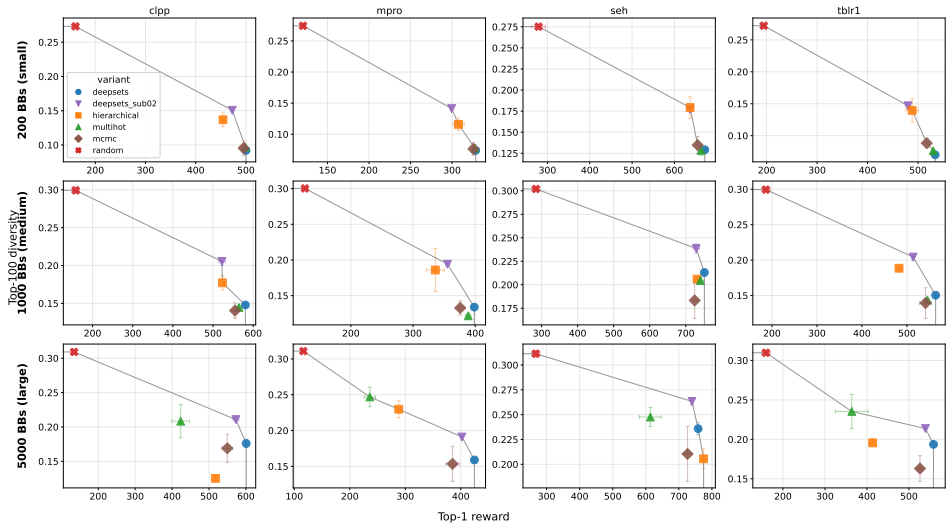


Figure 1: Top-1 DeepDEL reward discovered and top-100 diversity across 4 targets and 3 environment sizes (small, medium, large). DeepDEL reward models used a threshold value of $\lambda_0 = -9$ and width $h = 1$. The base DeepSets-GFN model was trained with full subsampling and the other DeepSets-GFN was trained with a subsampling rate of 0.02.

5.2 Reward Threshold Conditioning

Much like a probability distribution cannot be reconstructed from a single value of its CDF, the soft-count $r_{\lambda,H}(x)$ (Equation 3) cannot capture the full distribution of docking scores. The threshold λ selects which part of the score distribution the reward favors, down to resolution h, with large λ penalizing right-tail mass. For this reason, we study a version of the predictive model $\hat{g}_\theta$ in which the thresholding variable λ is itself a parameter.

We construct a dataset $\mathcal{D}_{\hat{G}}$ by sampling pairs (x, λ), where $x = (B_1, B_2, B_3)$ contains three random subsets of size ℓ and λ is uniformly sampled from a given interval $I = [\lambda_{\min}, \lambda_{\max}]$. We evaluate (x, λ) with $\tilde{G}_{\lambda,h}(x)$. On this dataset, we train a λ-conditioned predictor, where λ is appended to $\phi_{1,2,3}$ using a Fourier encoding $\phi_{\text{Fourier}}(\lambda)$:

$$\widehat{G}(B_1, B_2, B_3, \lambda) = \rho\left(\phi_1(B_1), \phi_2(B_2), \phi_3(B_3), \phi_{\text{Fourier}}(\lambda)\right). \tag{6}$$

On the generative side, we similarly condition DeepSets-GFN on λ. We train a conditional GFlowNet by sampling λ uniformly from I, which is a form of pre-training in which the GFlowNet learns the effect of λ on the reward function. Instead of the learnable partition constant $\text{Log}Z$, we learn a small neural network $(\text{Log}Z)(\phi_{\text{Fourier}}(\lambda))$.

In the following experiment, we measure the effect of this pretraining. For each target, we train a conditional predictive model $\hat{G}$ and a conditional DeepSets-GFN using $\lambda \in [-12, -6]$. This covers much of the range of docking scores across all four targets, as can be seen in Appendix C. We specialize the conditioned predictive model at various weights $\lambda = \lambda_0$ and fold the Fourier encoder values into the model weights, in order to obtain pretrained, non-conditioned models $\mathcal{F}^{(\lambda_0)}$, and similarly specialize $\text{Log}Z$ to a constant. Table 5.2 shows the top-1 reward obtained with and without pre-training, on each target, for three different λ thresholds. As the results show, pretraining is highly effective across threshold values in the conditioning interval.

5.3 Online Learning Improvement

We explore the generalization ability of the predictive DeepDEL model. Like any learned model, $\widehat{G}$ is only as good as its dataset; trained on randomly sampled objects, it is unlikely to be accurate out-of-the-box on exceptional objects. The two-level learning loop (discussed in Appendix D.2), in which an outer loop periodically corrects $\hat{f}$ against the oracle f while a nested inner loop retrains $\widehat{G}$ on the GFlowNet's own samples, is designed to improve the accuracy of $\widehat{G}$ over inner loop iterations.

Target	Pre-training	$\lambda_0 = -7$	$\lambda_0 = -9$	$\lambda_0 = -11$
ClpP	Yes	**0.749 (±0.019)**	**0.454 (±0.021)**	**0.103 (±0.031)**
	No	0.666 (±0.012)	0.292 (±0.014)	0.055 (±0.002)
Mpro	Yes	**0.740 (±0.002)**	**0.285 (± 0.0012)**	**0.058 (±0.002)**
	No	0.651 (±0.017)	0.189 (± 0.003)	0.045 (±0.002)
sEH	Yes	**0.890 (±0.010)**	**0.647 (±0.011)**	**0.250 (±0.014)**
	No	0.822 (±0.007)	0.541 (±0.027)	0.154 (±0.008)
TBLR1	Yes	**0.803 (±0.001)**	**0.389 (±0.013)**	**0.058 (±0.004)**
	No	0.725 (±0.040)	0.300 (±0.002)	0.049 (±0.001)

Table 1: Top-1 reward (higher is better, normalized $\times 10^{-3}$) across all targets and three fixed thresholds λ. Bold = better of the two pre-training variants. Mean $\pm$ standard deviation over 3 runs.

For this experiment, we picked sEH as our target, trained the proxy $\hat{f}$, and used it as our ground truth. We carried out a first online learning campaign, consisting of 5 inner loops of the online learning pipeline. From the final training set $\mathcal{H}_{\tilde{G}}$, we selected a stratified validation set V containing 10^4 libraries, spread over the entire range of values.

Figure 2 shows the validation error of the DeepDEL model against this external validation set, over the course of 5 online learning runs. For this experiment, we generated a fresh $\mathcal{H}'_{\tilde{G}}$ which we used as the initial dataset for a new online learning campaign, which we carried out over 3 seeds; $\mathcal{H}'_{\tilde{G}}$ was enriched with $5K$ selected libraries at each step. As a baseline, we compared to an online learning campaign during which we enriched $\mathcal{H}'_{\tilde{G}}$ at each step with $5K$ randomly-selected libraries instead of GFlowNet-selected libraries. We observe that online learning with GFlowNet results in a consistently decreasing validation error, while the random baseline does not. This indicates that the generalization capability of $\widehat{G}$ improves over online learning iterations, even on libraries selected during a previous independent campaign.

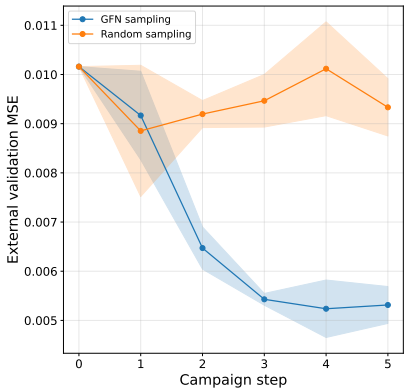


Figure 2: Validation error of DeepDEL on stratified validation set, online learning with GFlowNet vs. random training set enrichment.

6 Discussion and Future Works

In this work, we introduce DeepSets-GFN, a structure-aware flow model for subset selection tasks, together with a DeepSets-based method for amortizing combinatorial rewards, which we have applied directly to the task of DEL design (though neither is limited to this domain). Although we have used these methods to generate high-quality DELs up to size $20 \times 20 \times 20$, there are a number of challenges involved in generating much larger libraries. There comes a point where even the overhead in $O(\ell^3)$ becomes too expensive. In previous work, such as the original (Zaheer et al., 2017), DeepSets has shown an ability to generalize *across set cardinality*. In this work, we considered libraries of a predetermined size, but we did not investigate or make use of this property, except perhaps indirectly in DeepSets-GFN. In future work, we intend to explore this avenue further, as it could prove useful for scaling to larger libraries.

It is likely that our DeepSets-based combinatorial amortization scheme could find use in applications involving other types of combinatorial structures. DELs are proof that combinatorial explosion can be a good thing, as it is the very source of their usefulness; but in machine learning, combinatorial explosion remains a beast to be fully tamed.

7 AI USE STATEMENT

LLMS were used to generate code, for which open weight models were used almost exclusively. The authors report using AI for brainstorming and for polishing writing. The statement and proof of 1 was suggested by Claude.

8 ETHICS STATEMENT

This study did not involve human participants, animal subjects, or sensitive personal data. The authors declare that the research was carried out in accordance with established academic and ethical standards. No potential ethical, environmental, or social harms are anticipated as a result of this work.

9 REPRODUCIBILITY STATEMENT

Source code and building blocks used for this research is available at the anonymous repository: `https://anonymous.4open.science/r/DeepDEL`. The code is also included in the supplementary material.

ACKNOWLEDGMENTS

We thank Enamine for allowing us to use and share the set of 5056 building blocks which were used in this work.

REFERENCES

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 27381–27394. Curran Associates, Inc., 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

François Bergeron, Gilbert Labelle, and Pierre Leroux. *Combinatorial Species and Tree-like Structures*. Number 67 in Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1998. ISBN 978-0521573238. Translated by Margaret Readdy.

Vaibhav Bhandari, Keith S Wong, Jin Lin Zhou, Mark F Mabanglo, Robert A Batey, and Walid A Houry. The role of clpp protease in bacterial pathogenesis and human diseases. *ACS chemical biology*, 13(6):1413–1425, 2018.

Abhimanyu Das and David Kempe. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 1057–1064. Omnipress, 2011.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. *arXiv preprint arXiv:2402.10309*, 2024.

Thomas Duflot, Clothilde Roche, Fabien Lamoureux, Dominique Guerrot, and Jeremy Bellien. Design and discovery of soluble epoxide hydrolase inhibitors for the treatment of cardiovascular diseases. *Expert opinion on drug discovery*, 9(3):229–243, 2014.

Enamine. Enamine REAL Space. `https://enamine.net/building-blocks/building-blocks-catalog`, 2026.

Spilios Evmorfos, Zhaoyi Xu, and Athina Petropulu. Gflownets for Sensor Selection. In *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Rome, Italy, September 2023. IEEE. ISBN 979-8-3503-2411-2. doi: 10.1109/MLSP55844.2023.10285914.

Dorte Frees, Saara NA Qazi, Philip J Hill, and Hanne Ingmer. Alternative roles of clpx and clpp in staphylococcus aureus stress tolerance and virulence. *Molecular microbiology*, 48(6):1565–1578, 2003.

Olivier Gaillot, Elisabeth Pellegrini, Søren Bregenholt, Shamila Nair, and Patrick Berche. The clpp serine protease is essential for the intracellular parasitism and virulence of listeria monocytogenes. *Molecular microbiology*, 35(6):1286–1294, 2000.

Miguel García-Ortegón, Gregor NC Simm, Austin J Tripp, José Miguel Hernández-Lobato, Andreas Bender, and Sergio Bacallado. Dockstring: easy molecular docking yields better benchmarks for ligand design. *Journal of chemical information and modeling*, 62(15):3486, 2022.

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979. ISBN 0-7167-1045-5.

Christopher J. Gerry, Mathias J. Wawer, Paul A. Clemons, and Stuart L. Schreiber. DNA Barcoding a Complete Matrix of Stereoisomeric Small Molecules. *Journal of the American Chemical Society*, 141(26):10225–10235, July 2019. ISSN 0002-7863. doi: 10.1021/jacs.9b01203.

Robert A Goodnow, Jr, Christoph E Dumelin, and Anthony D Keefe. DNA-encoded chemistry: enabling the deeper sampling of chemical space. *Nat. Rev. Drug Discov.*, 16(2):131–147, February 2017.

David Hsieh, Eric Xueyu Chen, Joseph Wang Franses, Lynn G Feun, Zishuo Ian Hu, Gentry Teng King, Jimmy J Hwang, David D Stenehjem, Casey Cunningham, and Daneng Li. Tegavivint, a downstream wnt/β-catenin inhibitor: Dose-finding results from a phase 1/2 trial in advanced hepatocellular carcinoma (ahcc). *Journal of Clinical Oncology*, 44:4015–4015, 2026.

Maria Huegle, Gabriel Kalweit, Branka Mirchevska, Moritz Werling, and Joschka Boedecker. Dynamic input for deep reinforcement learning in autonomous driving. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 7566–7573. IEEE, 2019.

James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6):1239–1249, 2011.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International conference on machine learning*, pp. 9786–9801. PMLR, 2022.

Zhenming Jin, Xiaoyu Du, Yechun Xu, Yongqiang Deng, Meiqin Liu, Yao Zhao, Bing Zhang, Xiaofeng Li, Leike Zhang, Chao Peng, et al. Structure of mpro from sars-cov-2 and discovery of its inhibitors. *Nature*, 582(7811):289–293, 2020.

M Kalinic and P Kómár. Denoising DNA encoded library screens with sparse learning. *ACS Combinatorial Science*, 22(8):410–421, 2020.

Hyeonah Kim, Minsu Kim, Celine Roget, Dionessa Biton, Louis Vaillancourt, Yves V Brun, Yoshua Bengio, and Alex Hernandez-Garcia. Synthesizable molecular generation via soft-constrained gflownets with rich chemical priors. *arXiv preprint arXiv:2602.04119*, 2026.

Daniel W Kneller, Gwyndalyn Phillips, Hugh M O'Neill, Robert Jedrzejczak, Lucy Stols, Paul Langan, Andrzej Joachimiak, Leighton Coates, and Andrey Kovalevsky. Structural plasticity of sars-cov-2 3cl mpro active site cavity revealed by room temperature x-ray crystallography. *Nature communications*, 11(1):3202, 2020.

Patrick T Komiske, Eric M Metodiev, and Jesse Thaler. Energy flow networks: deep sets for particle jets. *Journal of High Energy Physics*, 2019(1):1–46, 2019.

Michał Koziarski, Mohammed Abukalam, Vedant Shah, Louis Vaillancourt, Doris Alexandra Schuetz, Moksh Jain, Almer van der Sloot, Mathieu Bourgey, Anne Marinier, and Yoshua Bengio. Towards DNA-encoded library generation with GFlowNets. In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024a.

Michał Koziarski, Andrei Rekesh, Dmytro Shevchuk, Almer van der Sloot, Piotr Gaiński, Yoshua Bengio, Cheng-Hao Liu, Mike Tyers, and Robert A Batey. Rgfn: Synthesizable molecular generation using gflownets. *Advances in Neural Information Processing Systems*, 37:46908–46955, 2024b.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 2019.

Robert Lee, Serena Mou, Vibhavari Dasagi, Jake Bruce, Jürgen Leitner, and Niko Sünderhauf. Zero-shot sim-to-real transfer with modular priors. *CoRR*, 2018.

Jiong Li and Cun-Yu Wang. TBL1–TBLR1 and β-catenin recruit each other to wnt target-gene promoter for transcription activation and oncogenesis. *Nature cell biology*, 10(2):160–169, 2008.

Xinghua Li, Weijiang Liang, Junling Liu, Chuyong Lin, Shu Wu, Libing Song, and Zhongyu Yuan. Transducin (β)-like 1 x-linked receptor 1 promotes proliferation and tumorigenicity in human breast cancer via activation of beta-catenin signaling. *Breast Cancer Research*, 16(5):465, 2014.

Katherine S. Lim, Andrew G. Reidenbach, Bruce K. Hua, Jeremy W. Mason, Christopher J. Gerry, Paul A. Clemons, and Connor W. Coley. Machine Learning on DNA-Encoded Library Count Data Using an Uncertainty-Aware Probabilistic Loss Function. *Journal of Chemical Information and Modeling*, 62(10), May 2022. ISSN 1549-9596. doi: 10.1021/acs.jcim.2c00041.

Alexander Litovchick, Christoph E Dumelin, Sevan Habeshian, Diana Gikunju, Marie-Aude Guié, Paolo Centrella, Ying Zhang, Eric A Sigel, John W Cuozzo, Anthony D Keefe, et al. Encoded library synthesis using chemical ligation and the discovery of sEH inhibitors from a 334-million member library. *Scientific reports*, 5(1):10916, 2015.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

Kevin McCloskey, Eric A Sigel, Steven Kearnes, Ling Xue, Xia Tian, Dennis Moccia, Diana Gikunju, Sana Bazzaz, Betty Chan, Matthew A Clark, et al. Machine learning on DNA-encoded libraries: a new paradigm for hit finding. *Journal of Medicinal Chemistry*, 63(16):8857–8866, 2020.

Carlos Moreno-Cinos, Kenneth Goossens, Irene G Salado, Pieter Van Der Veken, Hans De Winter, and Koen Augustyns. Clpp protease, a promising antimicrobial target. *International journal of molecular sciences*, 20(9):2232, 2019.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265–294, 1978. doi: 10.1007/BF01588971.

Dario Neri and Richard A Lerner. DNA-encoded chemical libraries: A selection system based on endowing organic compounds with amplifiable information. *Annu. Rev. Biochem.*, 87(1):479–502, June 2018.

Valentina Perissi, Claudio Scafoglio, Jie Zhang, Kenneth A Ohgi, David W Rose, Christopher K Glass, and Michael G Rosenfeld. TBL1 and TBLR1 phosphorylation on regulated gene promoters overcomes dual ctbp and ncor/smrt transcriptional repression checkpoints. *Molecular cell*, 29(6): 755–766, 2008.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Alexander L Satz. What do you get from DNA-encoded libraries? *ACS medicinal chemistry letters*, 9(5):408, 2018.

Alexander L Satz, Remo Hochstrasser, and Ann C Petersen. Analysis of current DNA encoded library screening data indicates higher false negative rates for numerically larger libraries. *ACS Combinatorial Science*, 19(4):234–238, 2017.

Alexander L Satz, Andreas Brunschweiger, Mark E Flanagan, Andreas Gloger, Nils JV Hansen, Letian Kuai, Verena BK Kunig, Xiaojie Lu, Daniel Madsen, Lisa A Marcaurelle, et al. DNA-encoded chemical libraries. *Nature Reviews Methods Primers*, 2(1):3, 2022.

Seonghwan Seo, Minsu Kim, Tony Shen, Martin Ester, Jinkyoo Park, Sungsoo Ahn, and Woo Youn Kim. Generative flows on synthetic pathway for drug design. In *International Conference on Learning Representations*, volume 2025, pp. 14029–14060, 2025.

Tony Shen, Seonghwan Seo, Grayson Lee, Mohit Pandey, Jason R Smith, Artem Cherkasov, Woo Youn Kim, and Martin Ester. Tacogfn: Target-conditioned gflownet for structure-based drug design. *arXiv preprint arXiv:2310.03223*, 2023.

Tomoko Yamamoto, Hiroshi Sashinami, Akiko Takaya, Toshifumi Tomoyasu, Hidenori Matsui, Yuji Kikuchi, Tomoko Hanawa, Shigeru Kamiya, and Akio Nakane. Disruption of the genes for ClpXP protease in salmonella enterica serovar typhimurium results in persistent infection in mice, and development of persistence requires endogenous gamma interferon and tumor necrosis factor alpha. *Infection and immunity*, 69(5):3164–3174, 2001.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. *Advances in neural information processing systems*, 36:11952–11969, 2023.

A PROOFS

A.1 PROOF OF PROPOSITION 1

We fix w and H and $D = (B_1, B_2, B_3)$; let $x_1, \ldots, x_n$ be the elements of the multiset $\{f(m) : m \in D\}$. Let $r(\lambda) = r_{\lambda,H}(x)$ and $N(\lambda) = |\{i | x_i \leq \lambda\}|$ be the soft and hard counts respectively. Define $\epsilon_\delta = \sigma(-\delta/H) = (1 + e^{\delta/H})^{-1} \leq e^{-\delta/H}$. We prove that

$$(1 - \epsilon_\delta)N(\lambda - \delta) \leq r(\lambda) \leq N(\lambda + \delta) + n\epsilon_\delta.$$

We partition the indices $[1 : n]$ into three subsets:

$$A = \{i : x_i \leq \lambda - \delta\}, \quad B = \{i : \lambda - \delta < x_i \leq \lambda + \delta\}, \quad C = \{i : \lambda + \delta < x_i\}$$

so that $|A| = N(\lambda - \delta), |A| + |B| = N(\lambda + \delta), |C| = n = N(\lambda + \delta)$. For $i \in A$, we have $\frac{\lambda - x_i}{h} \geq \delta/h$, so each element in A contributes at least $\sigma(\delta/h) = 1 - \epsilon_\delta$. This gives the lower bound. For the second bound, we bound above each contributions coming from A or B by 1, giving $N(\lambda + \delta)$, and the third by $\sigma(-\delta/h) = \epsilon_\delta$, giving the upper bound.

A.2 PROOF OF PROPOSITION 2

Consider the sum $r(\lambda) = \sum \sigma\left(\frac{\lambda - x_i}{H}\right)$. If we put $z = e^{\lambda/H}$ and $a_i = e^{x_i/H}$, then $\sigma(\lambda - x_i) = \frac{z}{z + a_i}$. The multi-set $\{-a_i\}$ can be recovered as the set of poles of $\sum \frac{z}{z+a_i}$, with the residue at the pole providing the multiplicity.

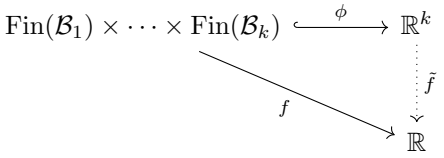
A.3 Proof of Proposition3

In fact, we can even take $h = 1$. Order each $\mathcal{B}_i$ and map $b_{i,n}$ to $\phi_i(b_{i,n}) = 2^n$, which is a binary one-hot vector representation of $b_{i,n}$. If $\mathrm{Fin}(X)$ denotes the set of finite subsets of X, then the map

$$\mathrm{Fin}(\mathcal{B}_1) \times \cdots \times \mathrm{Fin}(\mathcal{B}_k) \xrightarrow{\phi} \mathbb{R}^k \tag{7}$$

$$(B_1, \ldots, B_K) \mapsto \left(\sum_{b \in B_1} \phi(b), \ldots, \sum_{b \in B_K} \phi(b) \right) \tag{8}$$

is injective, and its image I is a countable subset of $\mathbb{R}^k$ without accumulation point, and hence there are infinitely many continuous extensions $\tilde{f} : \mathbb{R}^k \to \mathbb{R}$ to the values of f on I:



Then, with $\rho = \tilde{f}$, we have written f in the multi-variate DeepSets form.

B Target Selection

We selected the same four targets (sEH, Mpro, ClpP, and TBLR1) as Koziarski et al. (2024b), which were originally chosen for their diverse functions and binding sites.

Soluble Epoxide Hydrolase (sEH) is a key enzyme in eicosanoid-mediated inflammation and a target for cardiovascular disease and other conditions, with sEH inhibitors having been shown to improve vascular and cardiac function in animal models of cardiovascular diseases (Duflot et al., 2014). sEH has been used to evaluate machine learning based hit finding, with high experimental hit rates reported from models trained on DEL selection data (McCloskey et al., 2020), and is supported by a well-validated biochemical activity assay (Litovchick et al., 2015).

The main protease of SARS-CoV-2, (M^{pro}, also known as $3CL^{pro}$ or NSP5; hereafter *Mpro* for ease of use) plays a critical role in mediating viral replication and transcription (Jin et al., 2020). The Mpro catalytic site contains several sub-pockets ($S1, S1', S3, S4$) that recognize distinct amino acid motifs in substrate cleavage sites, and it shows structural plasticity (Kneller et al., 2020), making it a challenging test case for structure-based design.

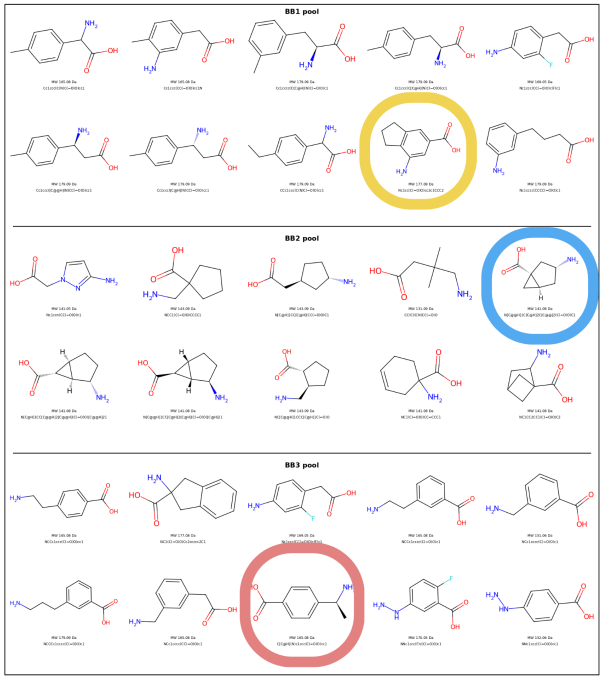
Well characterized in many species, ClpP is involved in the proteolysis of defective and misfolded proteins (Moreno-Cinos et al., 2019). Together with Lon, it mediates an estimated 80% of intracellular proteolysis in bacteria (Bhandari et al., 2018), and its deletion attenuates or reduces virulence across diverse pathogens (Gaillot et al., 2000; Yamamoto et al., 2001; Frees et al., 2003).

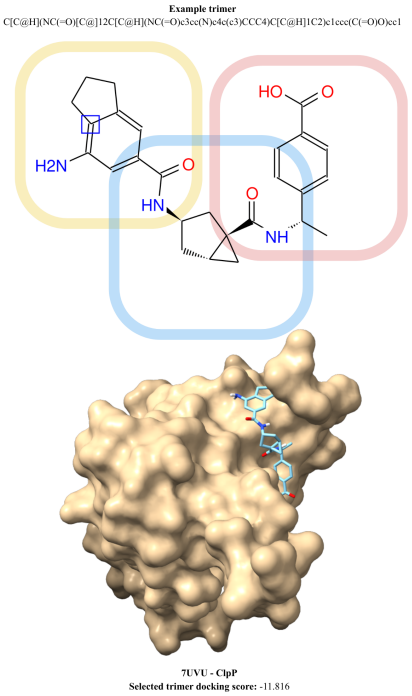
TBL1 and its paralog TBLR1 are WD40-domain proteins that form part of the NCoR corepressor complex, where they work as exchange factors converting repressed promoters to an active state (Perissi et al., 2008). TBLR1 and β-catenin reciprocally recruit each other to Wnt target genes (Li & Wang, 2008), and TBLR1 drives breast cancer proliferation through cyclin D1 in a TCH4-dependent manner (Li et al., 2014). The TBL1 inhibitor tegavivint (which disrupts the TBL1/β-catenin complex) has shown clinical activity in Wnt-mutant hepatocellular carcinoma (aHCC) (Hsieh et al., 2026), confirming that the target is druggable.

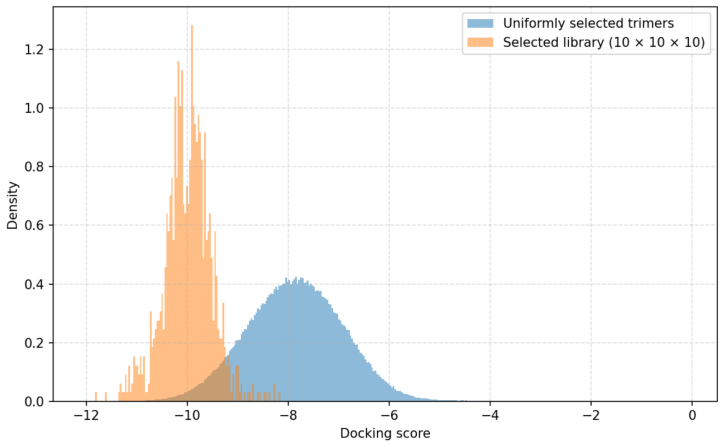
C Generated libraries

We provide samples of $10 \times 10 \times 10$ generated libraries for each of the 4 targets. Each was discovered over a single online learning run of 10 inner loops and 1 outer loop. At the end of each inner loop, 5 libraries were selected for docking on the basis of their high $\tilde{G}$ score. From each library, a trimer was selected at random and docked on the target.

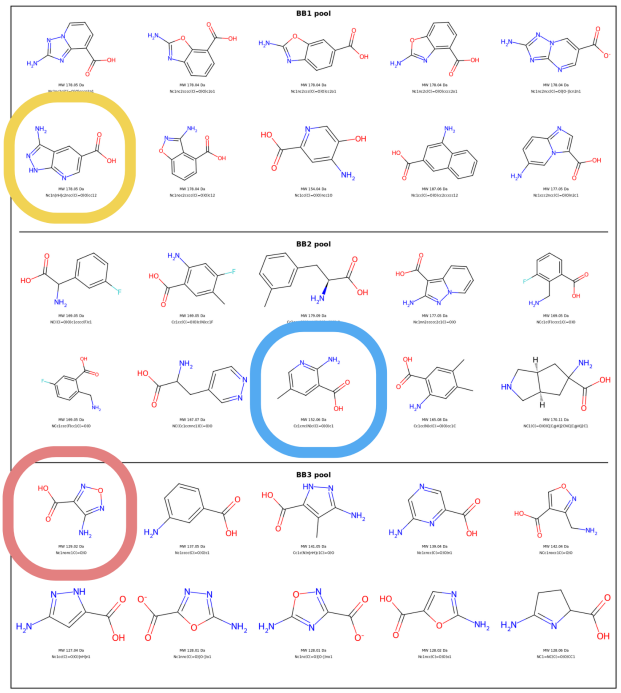
C.1 CLPP

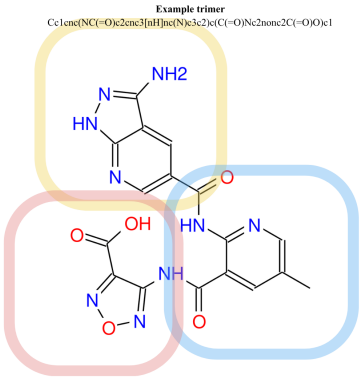

BB1 pool
BB2 pool
BB3 pool

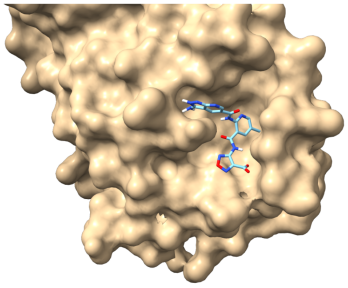

Example trimer
C[C@H](NC(=O)[C@]12C[C@H](NC(=O)c3cc(N)c4c(c3)CCC4)C[C@H]1C2)c1ccc(C(=O)O)cc1
H2N
HO
O
HN
NH
7UVU - ClpP
Selected trimer docking score: -11.816


Uniformly selected trimers
Selected library (10 × 10 × 10)
Density
Docking score
0.0
0.2
0.4
0.6
0.8
1.0
1.2
−12
−10
−8
−6
−4
−2
0

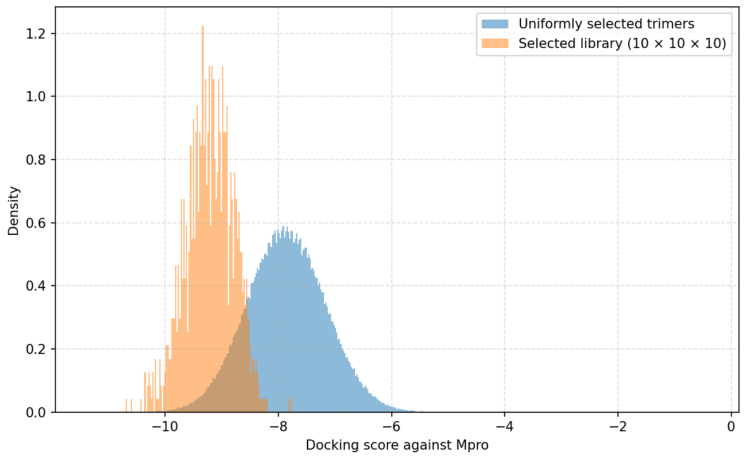
C.2 MPRO



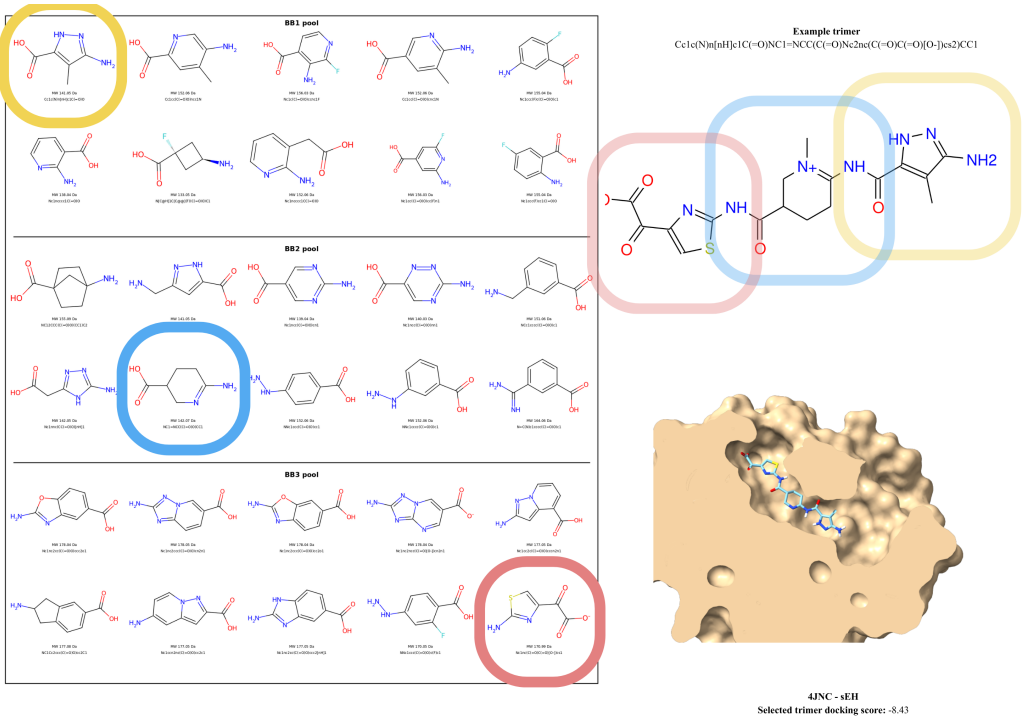


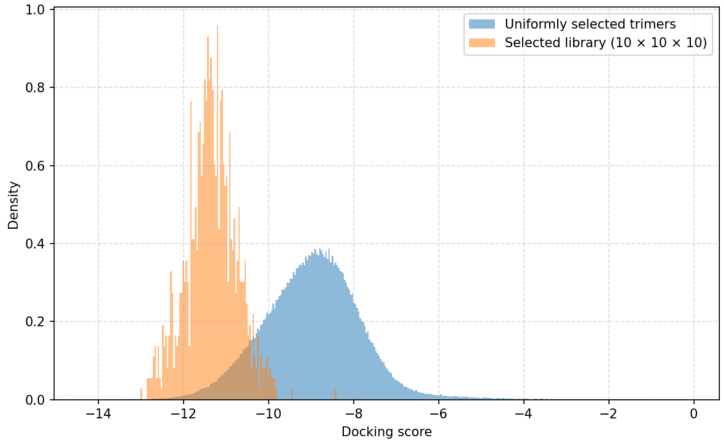


6W63 - Mpro
Selected trimer docking score: -10.69

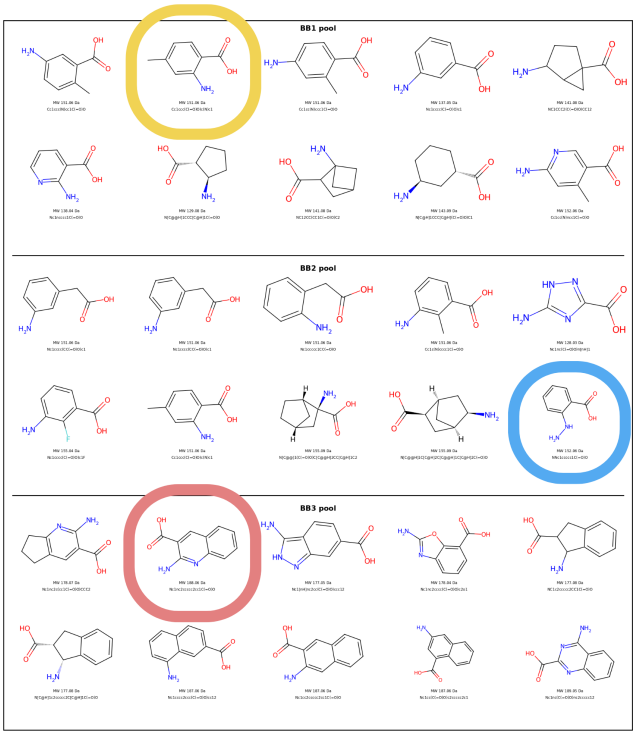


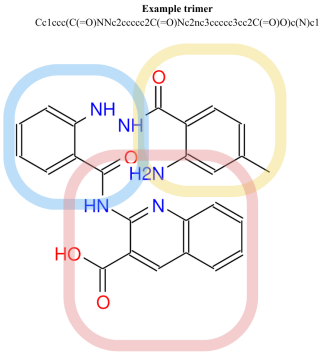
C.3 sEH

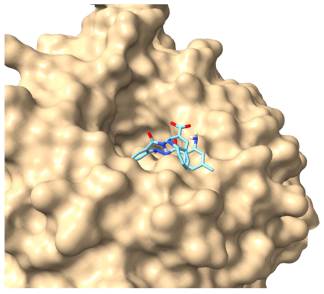

BB1 pool
BB2 pool
BB3 pool
Example trimer
Cc1c(N)n[nH]c1C(=O)NC1=NCC(C(=O)Nc2nc(C(=O)C(=O)[O-])cs2)CC1
4JNC - sEH
Selected trimer docking score: -8.43


Uniformly selected trimers
Selected library (10 × 10 × 10)
Density
Docking score
1.0
0.8
0.6
0.4
0.2
0.0
−14
−12
−10
−8
−6
−4
−2
0

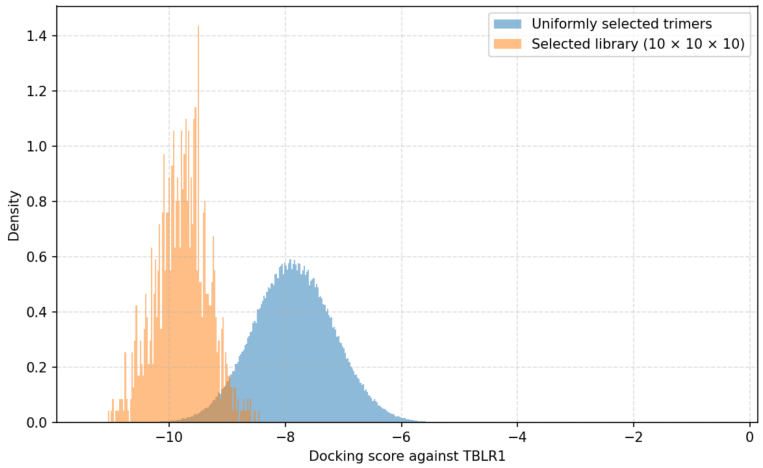
C.4 TBLR1







5NAF - tblr1
Selected trimer docking score: -11.03



D Implementation Details

To support reproducibility, we provide implementation details that were omitted from the main text due to space limitations. D.1 provides details on the environments used within our experimental setup. D.2 discusses the two-level online learning loop used to update the first-order and second-order proxy models. D.3 discusses the architecture and training details of the DeepDEL model, the first-order proxy $\hat{f}$, and GFlowNets, in addition to details regarding our computing infrastructure.

D.1 Environments

All experiments took place in an environment defined by a top-level DEL of the form $\mathcal{B} \times \mathcal{B} \times \mathcal{B}$, where $\mathcal{B}$ is a collection of 5056 bi-functional building blocks from the Enamine catalog (Enamine, 2026), having both a unique amine group and a unique acid group. The reaction was amide coupling in both steps, resulting in a simple click chemistry environment in which molecules are assembled amine-to-acid in chains of three. Building blocks and assembled molecules were both represented using Morgan fingerprints of radius 2 and length 2048. The environments of experiment 1 were obtained by restricting $\mathcal{B}$ to subsets of $200, 1000$ and 5000 building blocks. Experiments 2 used 5000 building blocks and Experiment 3 used the full environment of 5056 building blocks.

D.2 Two-Level Online Learning Loop

Figure 3 provides an overview of the full two-level online learning loop, which we describe step-by-step below.

We begin by sampling trimers $m \in \mathcal{L}$, uniformly at random and scoring them with the oracle, providing us with a fixed dataset $\mathcal{H}_f = (m, f(m))$ of true oracle scores. After this, $\hat{f}$ is initially trained on $\mathcal{H}_f$. With $\hat{f}$ trained, we construct a second dataset, $\mathcal{H}_{\tilde{G}}$, by sampling a set of DELs of size $\ell \times \ell \times \ell$ uniformly at random and scoring each with $\tilde{G}$.

It is here that the inner loop begins. Using the current $\tilde{H}_{\tilde{G}}$, we train $\widehat{G}$. We train the GFlowNet for λ trajectories against the reward $r(x) = \exp(\widehat{G}(x))$. We apply a selection rule to evaluate some of the sampled trajectories with $\tilde{G}$, appending the values to $\mathcal{H}_{\tilde{G}}$. From those, we again apply a selection rule to select a number for evaluation by the oracle F, and for inclusion in the dataset $\mathcal{H}_F$ We apply a selection rule (i.e., top-m or m-leaders) to $\mathcal{H}$ and add the selected DELs to $H_{\tilde{G}}$, growing the dataset that $\widehat{G}$ will be retrained on. This cycle (retrain $\widehat{G}$, generate with GFlowNet, select and fold samples back in) repeats for n iterations, with $\widehat{G}$ improving each time as it learns from the GFlowNet's own increasingly targeted samples.

After n inner loop iterations, control returns to the outer loop where $\hat{f}$ is retrained on $\mathcal{H}_f$, and because $\widehat{G}$'s targets depend on $\hat{f}$, a fresh $\mathcal{H}_{\tilde{G}}$ is constructed using the newly updated $\hat{f}$ before the inner loop resumes. In this way, the outer loop periodically corrects $\hat{f}$ against the true oracle, while the inner loop, nested within each outer step, lets $\widehat{G}$ and the GFlowNet co-evolve against whatever the current proxy $\hat{f}$ defines as the target.

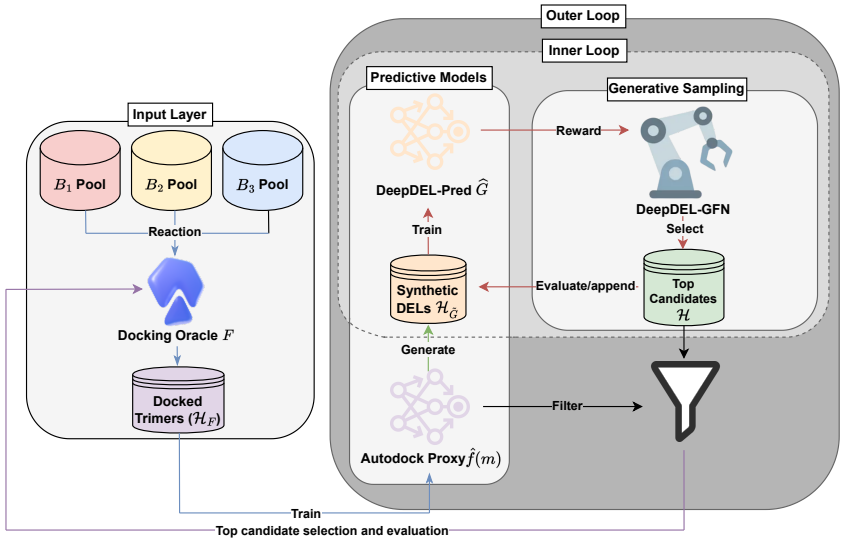


Figure 3: The high-level online learning loop, alternating between an outer loop that retrains the first-order proxy $\hat{f}$ and an inner loop that trains the GFlowNet against a second-order proxy $\widehat{G}$.

D.3 Experimental Details

D.3.1 DeepDEL models

The DeepDEL models which were used as log-rewards in the experiments used Φ_i models of width 6144 and depth 2, while ρ had width 2048 and depth 2. The entire architecture used approximately 189 million parameters. The output head used a scaled sigmoid activation, and the models were trained using soft binary cross-entropy, on batches of 256 elements, using a learning rate of 10^{-4} and a ridge regularization coefficient of 10^{-6}, for 30 epochs, using early stopping with a 4 epoch criterion.

Weight constraint In our implementation of the the soft count, we implemented a weight constraint as large molecules are more likely to result in artificially good docking scores that do not reflect true binding affinity (García-Ortegón et al., 2022). Given a molecular weight threshold w, in lieu of Eq. 3, we used

$$r_{\lambda,H,w}(x) = 1 + \sum_{m \in D_{\leq x}} \sigma\left(\frac{\lambda - f(m)}{H}\right), \qquad H > 0, \quad \lambda \in \mathbb{R}, \tag{9}$$

where $D = c(x)$, $D_{\leq w} = \{m \in D \mid \text{MolWeight}(m) \leq w\}$ for the subset of D consisting of molecules with weight $\leq w$.

D.3.2 Docking proxies $\hat{f}$

To train the docking proxies, for each target $500K$ trimers were selected at random from $\mathcal{L}$ and docked using Autodock Vina. The proxies were trained as plain MLPs that map a molecule's 2048-bit ECFP/Morgan fingerprint (radius 2) to a single scalar docking score, with three hidden layers of 1024 units, ReLU activations, 0.5 dropout, and a linear output head. The proxies were trained on the MSE loss, with target clipping to ≤ 0 and y-standardization that is undone at inference.

D.3.3 GFlowNets

The set selection environments $\mathcal{G}(\{\mathcal{B}_i\}, \ell)$ considered have no stop action, and, since terminal objects are determined by their cardinality, all complete trajectories have the same length, and all terminal objects receive equal amounts of a uniform flow. As a consequence, the uniform backwards policy attributes the same log-flow to all complete backwards trajectories, so that the P_B term in the TB loss can be taken as a constant, which we absorb in $\text{Log}Z$.

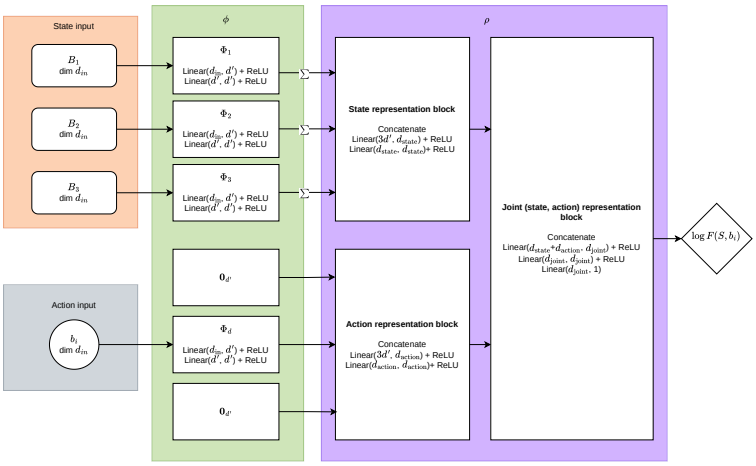


Figure 4: Architecture used for the DeepSets-GFN in the paper.

Experiments 2 & 3 used a subsampling rate of 0.02. The GFlowNets in Experiment 1 all used a subsampling rate of 1, except DeepSets. In all experiments, an inverse-temperature reward exponent of $\beta = 200$ was applied. All DeepSets-GFlowNets featured in the experiments used the architecture laid out in Figure 4. In Experiment 1, we used $d_h = 1024$, $d_{joint} = d_{state} = d_{action} = 1000$ for DeepSets-GFN, while DEL-GFN and H-DEL-GFN were implemented using 5-layer MLPs with hidden size of 1536. For H-DEL-GFN, we clustered $\mathcal{B}$ in 20 clusters using agglomerative clustering with Jaccard distance. A batch size of 1 was used for methods using stochastic gradient descent. MCMC was run with a single chain using 5000 burn-in steps.

Note on dynamic programming. We remark that the DeepDEL state representation $\Phi(S) = \sum_{s \in S} \Phi(\mathbf{b})$ is additive with respect to actions, i.e. $\Phi(S \sqcup \{b\}) = \Phi(S) + \Phi(\mathbf{b})$. As such, when sampling a trajectory from DeepSets-GFN, the state representation can be carried forward, and does not need to be re-computed at every step.

D.3.4 Compute Details

All experiments were carried out on clusters of the Digital Research Alliance of Canada, using A100 and H100 GPUs.

E Supplemental Experimental Data

E.1 Performance of docking proxies

The distributions of docking scores for the four targets is displayed in Figure 5, in which scores above 0 were clipped to 0. Table 2 shows final validation RMSE and Pearson r metrics on a held-out validation set.

Table 2: Docking proxy validation metrics

Target	Validation RMSE	Validation Pearson r
ClpP	0.7756	0.8075
Mpro	0.3783	0.9266
sEH	0.7558	0.8529
TBLR1	0.3574	0.9498

E.2 PERFORMANCE OF DEEPDEL

In Table 3, we report validation metrics of the DeepDEL models across the four targets, with $\lambda = -9$, $w = 500$ Da and $H = 1$, on $10 \times 10 \times 10$ libraries. Each dataset consisted of $500K$ randomly-generated $10 \times 10 \times 10$ libraries, 20% of which was held out for validation.

Table 3: DeepDEL validation metrics

Target	Validation Kendall τ	Validation MSE
ClpP	0.925853	0.006180
Mpro	0.931713	0.004757
sEH	0.924641	0.007398
TBLR1	0.925136	0.007113

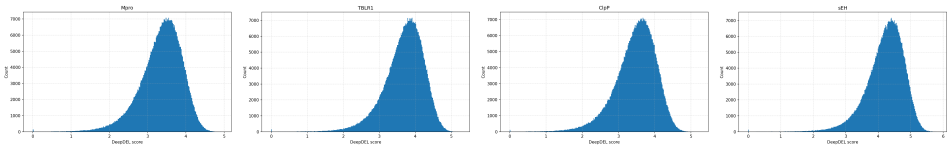


Figure 5: Histograms of scores in the training sets.

Table 4: **Top-1 DeepDEL reward** (higher is better, normalized $\times 10^{-3}$) across four unique targets (ClpP, Mpro, sEH, TBLR1) and three action space sizes, where $|\text{BB}|$ denotes the size of the building-block library (200 for small; 1000 for medium; 5000 for large). Mean (std) over 3 runs.

Target	$\|$BB$\|$	DS-GFN	DS-GFN (0.02)	DEL-GFN	H-DEL-GFN	MCMC	Random
ClpP	Small	**0.50 (0.002)**	0.47 (0.002)	0.50 (0.002)	0.45 (0.002)	0.50 (0.006)	0.16 (0.007)
	Medium	**0.58 (0.004)**	0.52 (0.008)	0.56 (0.006)	0.52 (0.005)	0.55 (0.006)	0.16 (0.009)
	Large	**0.60 (0.002)**	0.57 (0.008)	0.42 (0.024)	0.52 (0.002)	0.55 (0.015)	0.13 (0.006)
Mpro	Small	**0.33 (0.001)**	0.30 (0.001)	0.33 (0.000)	0.31 (0.007)	0.33 (0.001)	0.12 (0.001)
	Medium	**0.40 (0.004)**	0.36 (0.003)	0.39 (0.005)	0.34 (0.015)	0.38 (0.007)	0.13 (0.005)
	Large	**0.42 (0.004)**	0.40 (0.002)	0.24 (0.010)	0.29 (0.002)	0.39 (0.012)	0.12 (0.003)
sEH	Small	**0.67 (0.004)**	0.64 (0.001)	0.66 (0.004)	0.64 (0.006)	0.65 (0.005)	0.28 (0.016)
	Medium	**0.75 (0.006)**	0.73 (0.009)	0.74 (0.010)	0.73 (0.004)	0.72 (0.005)	0.28 (0.012)
	Large	0.76 (0.005)	0.74 (0.001)	0.61 (0.033)	**0.77 (0.009)**	0.73 (0.011)	0.27 (0.012)
TBLR1	Small	**0.53 (0.003)**	0.48 (0.006)	0.53 (0.002)	0.49 (0.013)	0.52 (0.004)	0.19 (0.004)
	Medium	**0.56 (0.002)**	0.51 (0.003)	0.55 (0.003)	0.48 (0.008)	0.54 (0.013)	0.19 (0.010)
	Large	**0.56 (0.007)**	0.54 (0.000)	0.36 (0.039)	0.41 (0.007)	0.53 (0.008)	0.16 (0.010)

Table 5: **Top-100 diversity** (higher is better, normalized $\times 10^{-3}$) across four unique targets (ClpP, Mpro, sEH, TBLR1) and three action space sizes, where $|\text{BB}|$ denotes the size of the building-block library (200 for small; 1000 for medium; 5000 for large). Mean (std) over 3 runs. Random sampling was excluded from the comparison, as it maximizes diversity in all cases without surprise.

Target	$\lvert\text{BB}\rvert$	DS-GFN	DS-GFN (0.02)	DEL-GFN	H-DEL-GFN	MCMC
ClpP	Small	0.09 (0.001)	**0.15 (0.000)**	0.10 (0.002)	0.14 (0.010)	0.10 (0.010)
	Medium	0.15 (0.003)	**0.21 (0.005)**	0.14 (0.002)	0.18 (0.009)	0.14 (0.010)
	Large	0.18 (0.006)	**0.21 (0.002)**	0.21 (0.024)	0.13 (0.004)	0.17 (0.020)
Mpro	Small	0.07 (0.005)	**0.14 (0.003)**	0.08 (0.006)	0.12 (0.010)	0.08 (0.010)
	Medium	0.13 (0.003)	**0.19 (0.003)**	0.12 (0.003)	0.19 (0.030)	0.13 (0.010)
	Large	0.16 (0.004)	0.19 (0.003)	**0.25 (0.014)**	0.23 (0.012)	0.15 (0.024)
sEH	Small	0.13 (0.002)	0.18 (0.002)	0.13 (0.006)	**0.18 (0.013)**	0.13 (0.010)
	Medium	0.21 (0.004)	**0.24 (0.005)**	0.20 (0.003)	0.21 (0.003)	0.18 (0.019)
	Large	0.24 (0.006)	**0.26 (0.004)**	0.25 (0.010)	0.21 (0.010)	0.21 (0.028)
TBLR1	Small	0.07 (0.001)	**0.15 (0.004)**	0.08 (0.002)	0.14 (0.018)	0.09 (0.007)
	Medium	0.15 (0.002)	**0.20 (0.005)**	0.14 (0.001)	0.19 (0.004)	0.14 (0.022)
	Large	0.19 (0.004)	0.21 (0.003)	**0.24 (0.022)**	0.20 (0.005)	0.16 (0.016)

F NOTATION

Table 6: Notation.

Symbol	Meaning
Combinatorics	
$g(B), g(B_1, \ldots, B_K)$	Set function or combinatorial set function
$u \in c(B_1, \ldots, B_K)$	c-structure on $(B_1, \ldots, B_K)$
$f(u)$	structure scoring function
Chemical space	
$\mathcal{M}$	Molecular universe
$\mathcal{B}$, $\mathcal{B}_i \subseteq \mathcal{M}$	Building-block pools
$\mathfrak{R} : \mathcal{B}_1 \times \mathcal{B}_2 \times \mathcal{B}_3 \to \mathcal{M}$	Two-step DEL reaction (assumed injective)
$\mathcal{L} = \operatorname{Im} \mathfrak{R}$	The DEL as a subset of $\mathcal{M}$; identified with $\mathcal{B}_1 \times \mathcal{B}_2 \times \mathcal{B}_3$ via $\mathfrak{R}$
$B_i \subseteq \mathcal{B}_i$	Set of building blocks
$b \in \mathcal{B}_i$	Set of building blocks
$u \in c(B_1, \ldots, B_K)$	c-structure on $(B_1, \ldots, B_K)$
$\mathbf{b}, \mathbf{m} \in \mathbb{R}^d$	d-dimensional ECFP of a molecule $m \in \mathcal{M}$
Scores and rewards	
$f(m)$	Element-level oracle (e.g. docking score)
$\hat{f}(m)$	Learned proxy for F
$r(x)$	Soft count
$\tilde{r}(x)$	Soft count with $\hat{f}$ instead of f
$\tilde{G} = \tilde{G}(x, \lambda, h)$	$\log \tilde{r}$
$\hat{G} = \hat{G}(x)$	3-variable DeepSets trained on $\tilde{G}(x, \lambda, h)$
Predictive DeepSets (DeepDEL)	
ϕ_i	Element embedding for pool i
ρ	Output function
Generative DeepSets (DeepSets-GFN)	
Φ_i	Element embedding for pool i
$\tilde{\Phi}_i$	Action embedding: Φ_i in slot i, zeros elsewhere, e.g. $\tilde{\Phi}_2 = (\mathbf{0}, \Phi_2, \mathbf{0})$
P	Output function