Multi-Objective GFlowNets

Anonymous Author(s)
Affiliation
Address
email

Abstract

In many applications of machine learning, like drug discovery and material design, the goal is to generate candidates that simultaneously maximize a set of objectives. As these objectives are often conflicting, there is no single candidate that simultaneously maximizes all objectives, but rather a set of Pareto-optimal candidates where one objective cannot be improved without worsening another. Moreover, these objectives, when considered in practice are often under-specified, making diversity of candidates a key consideration. The existing multi-objective optimization methods focus predominantly on covering the Pareto front, failing to capture diversity in the space of candidates. Motivated by the success of GFlowNets for generation of diverse candidates in a single objective setting, in this paper we consider Multi-Objective GFlowNets (MOGFNs). MOGFNs consist of a Conditional GFlowNet which models a family of single-objective sub-problems derived by decomposing the multi-objective optimization problem. Our work is the first to empirically demonstrate conditional GFlowNets. Through a series of experiments on synthetic as well as practically relevant material design and drug discovery tasks, we empirically demonstrate that MOGFNs outperform existing methods in terms of hypervolume, R2-distance and candidate diversity. We also demonstrate the effectiveness of MOGFNs over existing methods in active learning settings.

1 Introduction

Decision making in practical applications often involves reasoning about multiple, often conflicting, objectives [21]. For instance, in the design of materials for photovoltaic cells, the goal is to generate *novel* candidate materials (molecules) which have high energy-conversion efficiency, stability, ease of synthesizability and robustness to the elements [27]. These objectives are conflicting – molecules with high energy-conversion efficiency can be less stable and harder to synthesize – so there is no single candidate that maximizes all the objectives simultaneously. Such problems fall under the umbrella of *Multi-Objective Optimization* (MOO) [12, 29, MOO], wherein one is interested in identifying *Pareto-optimal* candidates. The set of Pareto-optimal candidates cover all the best tradeoffs among the objectives, i.e., the Pareto front, where each point on that front corresponds to a different set of weights associated with each of the objectives.

In-silico design of materials is typically driven by proxies that only approximate the true objectives. Since these proxies are generally models trained using finite data, there is epistemic uncertainty associated with their predictions. This makes it critical to generate *diverse* candidates to ensure that at least some candidates end up working in downstream evaluations [19]. Generative Flow Networks [4, 5, GFlowNets] are a recently proposed family of probabilistic models which tackle the problem of diverse candidate generation. Contrary to the reward maximization view of reinforcement learning (RL) and Bayesian optimization (BO), GFlowNets generate candidates with probability proportional to the reward. Sampling candidates, as opposed to greedily generating them, implicitly

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

encourages diversity in candidate generation. GFlowNets have shown promising results in single objective problems of molecule generation [4] and biological sequence design [19].

In this paper, we study *Multi-Objective GFlowNets* (MOGFNs), extensions of GFlowNets to tackle multi-objective optimization problems. We consider two variants of MOGFNs – (a) Preference-Conditional GFlowNets (MOGFN-PC) combining Reward-Conditional GFlowNets [5] with Weighted Sum Scalarization [12] and (b) MOGFN-AL an extension of GFlowNet-AL [19] for multi-objective active learning settings. We empirically demonstrate the advantage of these MOGFNs over existing approaches on a wide variety of tasks relevant for material design: the generation of small molecules, as well as the design of DNA Aptamers and fluorescent proteins. Our contributions are the following

C1 We introduce two new GFlowNet variants for multi-objective optimization: MOGFN-PC and MOGFN-AL.

C2 We empirically demonstrate the efficacy of MOGFN-PC in high-dimensional MOO problems. Our work is the first successful empirical validation of Reward-Conditional GFlowNets [5].

C3 Through a series of experiments on practically relevant tasks of molecule generation and sequence generation, we demonstrate that MOGFN-PC generates *diverse* Pareto-optimal candidates.

C4 In a challenging multi-objective active learning task for protein design, we show that MOGFN-AL results in significant improvements in sample-efficiency.

2 Related Work

Evolutionary Algorithms: Traditionally, evolutionary algorithms such as NSGA-II have been widely used in various multi-objective optimization problems [12, 24, 6]. More recently, [30] incorporated graph neural networks into evolutionary algorithms enabling them to tackle large combinatorial spaces. Unlike MOGFNs, evolutionary algorithms do not leverage any type of data, including any past experience, and therefore are required to solve each instance of a MOO from scratch rather than amortize computation during training in order to quickly generate solutions at run-time.

Multi-Objective Reinforcement Learning: MOO problems have also received significant interest in the reinforcement learning (RL) literature [17]. Traditional approaches broadly consist of learning sets of Pareto-dominant policies [35, 38, 34]. Recent work has focused on extending Deep RL algorithms for multi-objective settings such as Envelope-MOQ [41], MO-MPO [1, 2] , and MOReinforce [26]. A general shortcoming of RL based approaches is that they only discover a single mode of the reward function, and thus cannot generate diverse candidates, which also persists in the multi-objective setting.

Multi-Objective Bayesian Optimization (MOBO): Bayesian optimization (BO) has been used in the context of MOO when the objectives are expensive to evaluate and sample-efficiency is a key consideration. MOBO approaches consist of learning a surrogate model of the true objective function, which is used to define an acquisition function such as expected hypervolume improvement [13, 9, 10] and max-value entropy search [3], as well as scalarization-based approaches [32, 45]. The key drawbacks of MOBO approaches are that they do not consider the need for a diversity of generated candidates, and they mainly consider continuous state spaces. [36] proposed LaMBO, which uses language models in conjunction with BO for sequence design.

Other Works: [47] introduced LaMOO, which tackles the MOO problem by iteratively splitting the candidate space into smaller regions, whereas [11] introduce MORBO, which performs BO in parallel on multiple local regions of the candidate space. Both these methods, however, are limited to continuous candidate spaces.

3 Background

3.1 Multi-Objective Optimization

Multi-objective optimization (MOO) involves finding a set of feasible candidates $x^\star \in \mathcal{X}$ which all simultaneously maximize a set of objectives.

$$\max_{x \in \mathcal{X}} \left(R_1(x), \ldots, R_d(x)\right). \tag{1}$$

In general the objectives being optimized can be conflicting such that there is no single $x^\star$ which simultaneously maximizes all objectives. Consequently, the concept of *Pareto optimality* is adopted as an alternative solution in MOO.

Given $x_1, x_2 \in \mathcal{X}$, x_1 is said to *dominate* x_2 written ($x_1 \succ x_2$) *iff* $R_i(x_1) \geq R_i(x_2)\ \ \forall i \in \{1, \ldots, d\}$ and $\exists k \in \{1, \ldots, d\}$ such that $R_k(x_1) > R_k(x_2)$. A candidate $x^\star$ is *Pareto optimal* if there exists no other solution $x' \in \mathcal{X}$ which dominates $x^\star$. In other words, for a Pareto optimal candidate it is impossible to improve one objective without sacrificing another. The *Pareto set* is the set of all Pareto optimal candidates in $\mathcal{X}$, and the *Pareto front* is defined as the image of the Pareto set in objective-space. It is important to note that since the objectives being optimized in general might not be *injective*, any point on the Pareto front can be the image of several candidates in the Pareto set.

In practice, MOO problems can be challenging when they have high-dimensional objectives and candidates. A standard strategy is to decompose the multi-objective optimization problem into simpler single objective optimization problems. *Weighted sum scalarization* [12, 29] is a widely used method for this. We can assign a set of convex weights (preferences) ω_i to the objectives R_i, such that $\omega_i \geq 0$ and $\sum_{i=1}^{d} \omega_i = 1$. The MOO problem in Equation 1 is then decomposed into solving single objective sub-problems of the form

$$\max_{x \in \mathcal{X}} \sum_{i=1}^{d} \omega_i R_i(x). \tag{2}$$

Solutions to these sub-problems are Pareto optimal and, under certain assumptions, each Pareto optimal candidate for Equation 1 is a solution to a sub-problem for some ω [12]. Thus, in principle, solving the MOO problem can be viewed as solving a family of single objective optimization problems, each defined by the preference vector ω.

3.2 GFlowNets

Generative Flow Networks [4, 5, GFlowNets] are a family of probabilistic models which generate *compositional* objects $x \in \mathcal{X}$ with probability proportional to a given reward $R : \mathcal{X} \to \mathbb{R}^+$. The sequential construction of $x \in \mathcal{X}$ can be described as a trajectory $\tau \in \mathcal{T}$ in a weighted directed acyclic graph (DAG)[1] $\mathcal{G} = (\mathcal{S}, \mathcal{E})$, starting from an empty object s_0, using building blocks (actions) $\mathcal{A}$. The nodes $\mathcal{S}$ of this graph (states) correspond to the set of all possible objects (including partially constructed objects, i.e. $\mathcal{X} \subset \mathcal{S}$), that can be constructed using sequences of actions in $\mathcal{A}$. An edge $s \xrightarrow{a} s' \in \mathcal{E}$ indicates that action a at state s leads to state s'. The set of terminal nodes in $\mathcal{G}$ is the set of fully constructed objects $\mathcal{X}$, i.e. there is no outgoing edge from the fully constructed objects. A *complete* trajectory in $\mathcal{G}$, $\tau = (s_0 \to s_1 \cdots \to x)$ results in the generation of an object x. [2]

A forward policy $P_F(\cdot \mid s)$ is a distribution over the children of state s. Objects can be constructed by sampling trajectories using P_F starting from s_0. A backward policy $P_B(\cdot \mid s)$ defines a distribution over the parents of state s which can generate backward trajectories starting at any state, e.g., iteratively sampling P_B from terminal state x shows a way to construct x. Let $\pi(x)$ be the marginal likelihood of sampling trajectories following P_F that terminate in x, and partition function $Z = \sum_{x \in \mathcal{X}} R(x)$. The learning problem solved by GFlowNets is to estimate P_F such that $\pi(x) \propto R(x)$. We focus on the trajectory balance objective [28] to learn $P_F(\cdot \mid s; \theta), P_B(\cdot \mid s; \theta), Z_\theta$ which approximate the forward and backward policies and partition function, parameterized by θ:

$$\mathcal{L}(\tau; \theta) = \left(\log \frac{Z_\theta \prod_{s \to s' \in \tau} P_F(s'|s; \theta)}{R(x) \prod_{s \to s' \in \tau} P_B(s|s'; \theta)} \right)^2. \tag{3}$$

During training, trajectories are sampled from a mixture of the current forward policy P_F and a uniform random policy U, with a parameter δ which controls the fraction of actions sampled from U. In many problem settings, one would like to focus on the modes, which can be achieved by exponentiation of the reward with a constant β, resulting in $\pi(x) \propto R(x)^\beta$.

[1] If the object is constructed in a canonical order (say a string constructed from left to right), $\mathcal{G}$ is a tree.

[2] In a DAG $\mathcal{G}$ there can be several trajectories that result in the generation of the same object.

4 Multi-Objective GFlowNets

We broadly categorize *Multi-Objective GFlowNets* (MOGFNs) as GFlowNets that solve a family of subproblems derived from a MOO problem. In this section we consider two instantiations: (a) Preference-Conditional GFlowNets (MOGFN-PC), which leverage weighted-sum scalarization and (b) MOGFN-AL, which leverages multi-objective acquisition functions for active learning.

4.1 Preference-Conditional GFlowNets

Reward-conditional GFlowNets [5] are a generalization of GFlowNets that *simultaneously* model a family of reward functions, rather than a single reward function. Let $\mathcal{C}$ denote a set of conditioning information. Each $c \in \mathcal{C}$ induces a unique reward function $R(x|c)$. We can define a *family* of weighted DAGs $\{\mathcal{G}_c = (\mathcal{S}_c, \mathcal{E}),\ c \in \mathcal{C}\}$, which describe the construction of $x \in \mathcal{X}$, with conditioning information c available at all states.

A *Reward-Conditional GFlowNet* consists of the tuple $(P_F(s' \mid s, c), P_B(s \mid s', c), Z(c))$, parameterized by θ. $P_F(s' \mid s, c)$ and $P_B(s \mid s', c)$ are the *conditional* forward and backward policies and $Z(c)$ is the conditional partition function. In other words, Reward-conditional GFlowNets model the family of rewards $R(x|c),\ \ \forall c \in \mathcal{C}$ simultaneously. We refer the reader to [5] for a more thorough discussion on conditional GFlowNets.

Recall from Section 3.1 that multi-objective optimization problems can be decomposed into a family of single-objective problems each defined by a preference ω over the objectives. We can thus employ reward-conditional GFlowNets to model the family of reward functions, by using Δ^d (the d-simplex) as the conditioning set $\mathcal{C}$, i.e., possible values of the convex preference weights ω.

Preference-Conditional GFlowNets (MOGFN-PC) are reward-conditional GFlowNets, conditioned on the preferences $\omega \in \Delta^d$ over a set of objectives $\{R_1(x), \ldots, R_d(x)\}$. MOGFN-PC models the family of reward functions $R(x|\omega)$. $R(x|\omega)$ can be specified using the decomposition of the MOO problems. We consider three choices:

- Weighted-sum (WS) [12]: $R(x \mid \omega) = \sum_{i=1}^{d} \omega_i R_i(x)$
- Weighted-log-sum (WL) [5]: $R(x \mid \omega) = \sum_{i=1}^{d} R_i(x)^{\omega_i}$
- Weighted-Tchebycheff (WT) [7]: $R(x \mid \omega) = \min_{1 \le i \le d} \omega_i |R_i(x) - z_i^\star|$, where $z_i^\star$ denotes some ideal value for objective R_i.

Training MOGFN-PC: Training MOGFN-PC (or any reward-conditional GFlowNet) closely follows that of a standard GFlowNet and is described in Algorithm 1. The objective is to learn the parameters θ that parameterize the conditional policies $P_F(s' \mid s, \omega)$ and $P_B(s \mid s', \omega)$, and the log-partition function $\log Z(\omega)$. To this end, we consider an extension of the trajectory balance objective for reward-conditional GFlowNets:

$$\mathcal{L}_{TB}(\tau, \omega; \theta) = \left(\log \frac{Z(\omega) \prod_{s \to s' \in \tau} P_F(s'|s, \omega)}{R(x \mid \omega) \prod_{s \to s' \in \tau} P_B(s|s', \omega)} \right)^2 . \tag{4}$$

Another important consideration is the distribution $p(\omega)$ used to sample preferences during training. $p(\omega)$ influences the regions of the Pareto front that are captured by MOGFN-PC. In our experiments we primarily use a Dirichlet(α) to sample preferences. Note that $\alpha = 1$ corresponds to a uniform distribution over Δ^d and results in uniform coverage of the entire Pareto front. Finally, following Section 3.2, we use reward exponent β to focus on the modes of the conditional distribution $R(x \mid \omega)$.

MOGFN-PC and MOReinforce: MOGFN-PC is closely related to MOReinforce [26]. Both learn a preference-conditional policy to sample Pareto-optimal candidates. The key difference is the learning objective – MOReinforce uses a multi-objective version of REINFORCE [39], whereas MOGFN-PC uses a preference-conditional GFlowNet objective. As discussed in Section 3.1, each point on the Pareto front (corresponding to a unique ω) can correspond to multiple candidates in the Pareto set. MOReinforce, given a preference ω will converge to sampling the single candidate that maximizes $R(x|\omega)$. MOGFN-PC on the other hand, samples from $R(x|\omega)$ which enables generation of diverse candidates from the Pareto set for a give ω. We demonstrate this empirically in Section 5.

Algorithm 1: Training Multi-Objective GFlowNets

Input:
$p(\omega)$: Distribution for sampling preferences;
β: Reward Exponent;
δ: Mixing Coefficient for uniform actions in sampling policy;
N: Number of training steps;
Initialize:
$(P_F(s'|s,\omega), P_B(s|s',\omega), \log Z(\omega))$: Conditional GFlowNet with parameters θ;
for $i = 1$ *to* N **do**
  Sample preference $\omega \sim p(\omega)$;
  Sample trajectory τ following policy $\hat{\pi} = (1-\delta)P_F \ + \delta$Uniform ;
  Compute reward $R(x)^\beta$ for generated samples and corresponding loss $\mathcal{L}(\tau, \omega; \theta)$ using Equation 4;
  Update parameters θ with gradients from the loss, $\nabla_\theta \mathcal{L}(\tau, \omega)$;
end

4.2 MOGFN-AL

In many practical scenarios, the objective functions are also often computationally or economically expensive to evaluate. The goal here is to generate Pareto-optimal candidates with the fewest evaluations of the objective function. Traditionally, these scenarios are tackled broadly with active learning algorithms [49]. We focus on the task of biological sequence design with multiple objectives, building upon initial work on active learning with GFlowNets [19].

We consider the sequence-design framework presented in [36]. To simplify the notation, we use $\mathbf{R}(x) : \mathcal{X} \mapsto \mathbb{R}^d$ to denote a function which returns a vector with all the objectives $\mathbf{R}(x) = (R_1(x), \ldots, R_d(x))$. We start with an initial dataset of $\mathcal{D}_0 = (x_i, y_i)_{i=1}^N$ of candidates $x_i \in \mathcal{X}$ and their corresponding objective values $y_i = \mathbf{R}(x)$. Let $\hat{\mathcal{P}}_0$ denote the set of non-dominated points in $\mathcal{D}_0$. The MOO problem is solved sequentially over a number of rounds, each consisting of maximizing a scalar acquisition function $f : \mathcal{X} \mapsto \mathbb{R}$ to find candidates x that improve the current approximation of the Pareto front $\hat{\mathcal{P}}_i$. At the end of each round the generated candidates are evaluated with $\mathbf{R}$ and incorporated in the dataset for the next round $\mathcal{D}_i$. The acquisition function f is defined using an approximate Bayesian model $\hat{\mathbf{R}}$ of the objectives, estimated using $\mathcal{D}$. f accounts for the epistemic uncertainty in $\hat{\mathbf{R}}$, using it as a signal for exploration. We use *MOGFN-AL* to broadly define methods that tackle the MOO problem using GFlowNets to solve the sequence of sub-problems specified by the acquisition function f.

LaMBO [36] leverages denoising autoencoders to propose mutations to candidates $x \in \hat{\mathcal{P}}_i$ to generate $x' \in \mathcal{X}$, such that $x' \succ x$. LaMBO optimizes the acquisition function in the latent space of the denoising autoencoder. Instead, we propose using GFlowNets to generate the mutations to optimize the acquisition function. Given a sequence x, at each step, P_F selects a position in $l \in \{1, \ldots, |x|\}$ and a token $v \in \mathcal{A}$ to replace $x[l]$. We ensure acyclicity of $\mathcal{G}$ by only allowing a single mutation at any position within a trajectory. Thus, we generate mutations proportionally to the acquisition function, resulting in a diverse set of candidates to improve the Pareto front.

5 Empirical Results

In this section, we present our empirical findings across a wide range of tasks ranging from sequence design to molecule generation. More specifically, we consider two classes of tasks as summarized in Figure 1 - *(a)* Sequence generation where the trajectories τ induce a tree. *(a)* Graph generation where the trajectories τ induce a DAG $\mathcal{G}$. Through our experiments we answer the following questions:

Q1 *Can MOGFNs model the preference-conditional reward distribution?*

Q2 *Can MOGFNs sample Pareto-optimal candidates?*

Q3 *Are candidates sampled by MOGFNs* diverse*?*

Q4 *Do MOGFNs scale to high-dimensional problems relevant in practice?*

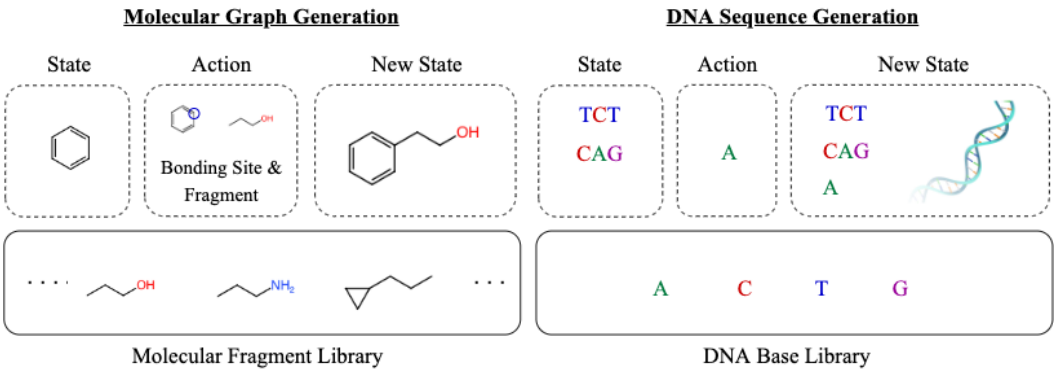

Figure 1: **Tasks Considered**: We consider two classes of tasks where the $\mathcal{G}$ is a DAG or a Tree. In this figure we represent the one step transition from the current *State* to a *New State*. *Left* All the molecular tasks considered in this paper (QM9 and Fragments) presume a DAG structure. *Right*: All the sequence modelling tasks (Strings and DNA generation) presume a tree structure.

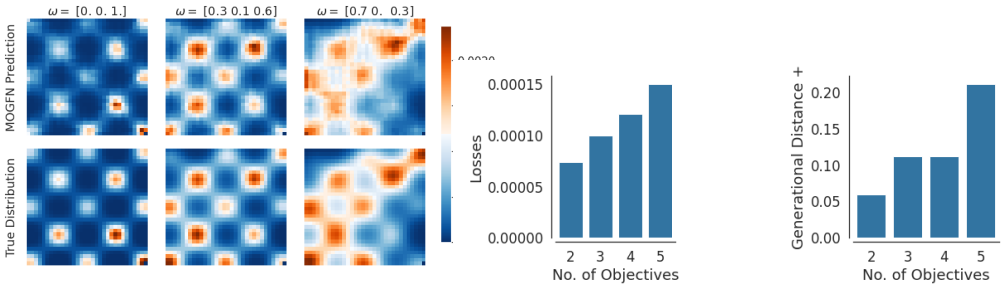

Figure 2: *(Left)* Visualization of the distribution learned by MOGFN-PC (Top) and ground truth (Bottom) on hypergrid of size 32×32 with 3 objectives. *(Middle)* Average test loss between the MOGFN distribution and the true distribution across objectives. *(Right)* Generational Distance $+$ metrics of MOGFN-PC across objectives.

To answer these questions, we rely on standard performance indicators used in the multi-objective optimization literature that quantify the Pareto front approximation such as the **Hypervolume Indicator** (HV), **Generational Distance+** (GD+), and the $\boldsymbol{R_2}$ **indicator**. To quantify diversity we rely on the **Top-K Diversity** and **Top-K Reward** metrics from [4]. The metrics are described in detail in Section A. For all our empirical evaluations we sample a set of preferences which are fixed for all the baselines and MOGFN-PC. For each preference, we sample 128 candidates from which we pick the top 10, and we report their scalarized reward and diversity averaged over preferences. We use these sets of samples to compute the HV and R_2 indicators. We pick the best hyperparameters for both MOGFN-PC and baselines based on the HV indicator, and report the mean and standard deviation across three seeds for all quantities.

5.1 Synthetic Tasks

5.1.1 Hyper-Grid

We first study the ability of MOGFN-PC to capture the preference-conditional reward distribution, in a multi-objective version of the HyperGrid task from [4]. Consider an n-dimensional hypercube gridworld where each cell in the grid corresponds to a state. The agent starts at the top left coordinate marked as $(0, 0, \dots)$ and is allowed to move only towards the right, down, or stop. When the agent performs the *stop* action, the trajectory terminates and the agent receives a non-zero reward. In this work, we consider the following reward functions - `brannin(x)`, `currin(x)`, `sphere(x)`, `shubert(x)`, `beale(x)`. See Appendix B.1 for more details on the reward functions.

Table 1: Results for the `regex_3` and `regex_3_conf` synthetic string tasks.

Algorithm	`regex_3`				`regex_3_conf`			
	Reward (↑)	Diversity (↑)	HV (↑)	R_2 (↓)	Reward (↑)	Diversity (↑)	HV (↑)	R_2 (↓)
Envelope-MOQ	$0.05_{\pm 0.04}$	$0_{\pm 0}$	$0.012_{\pm 0.013}$	$19.66_{\pm 0.66}$	$0.08_{\pm 0.015}$	$0_{\pm 0}$	$0.023_{\pm 0.011}$	$21.18_{\pm 0.72}$
MOReinforce	$0.12_{\pm 0.02}$	$0_{\pm 0}$	$0.015_{\pm 0.021}$	$20.32_{\pm 0.93}$	$0.031_{\pm 0.001}$	$0_{\pm 0}$	$0.036_{\pm 0.009}$	$21.04_{\pm 0.51}$
MOSoftQL	$0.28_{\pm 0.03}$	$21.09_{\pm 0.65}$	$0.093_{\pm 0.025}$	$15.79_{\pm 0.23}$	$0.3639_{\pm 0.011}$	$23.131_{\pm 0.6736}$	$0.105_{\pm 0.014}$	$12.803_{\pm 0.2617}$
MOGFN-PC	$0.44_{\pm 0.01}$	$19.79_{\pm 0.08}$	$0.220_{\pm 0.017}$	$9.97_{\pm 0.45}$	$0.3773_{\pm 0.001}$	$22.712_{\pm 0.2576}$	$0.121_{\pm 0.015}$	$11.388_{\pm 0.1653}$

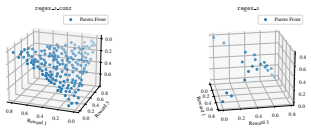

Figure 3: Pareto front of candidates generated by MOGFN-PC on the `regex_3_conf` (*Left*) and `regex_3` (*Right*) tasks

We present our qualitative and quantitative results in Figure 2. Since this is a toy setting, we can compute the *Multi-Objective GFlowNets* (MOGFN)-PC learnt distribution in closed form. To answer **Q1**, in Section 5.1.1, we visualize the learned distribution of MOGFN conditioned on a fixed preference vector ω and contrast it with the true distribution. We can clearly see that the learned distribution of MOGFN-PC is almost the same as the true distribution. Moreover, MOGFN-PC is able to capture all the modes in the distribution, which suggests that diversity is captured, answering **Q3**. We answer **Q2** and **Q4** quantitatively; more specifically, we present the average L_1-loss between the MOGFN-PC distribution and the true distribution averaged across a set of fixed test preference vectors covering the simplex in Section 5.1.1. We can see that for all objectives considered the loss is consistently low. Finally, to test the Pareto performance, we measure the **Generational Distance** + (GD+) across different numbers of objectives. We observe that the MOGFN-PC not only learns a distribution that is close to the true distribution but also learns to sample from the Pareto front.

5.1.2 String

We consider variants of the synthetic sequence design task from [36]. Given an alphabet of available actions, the task is to generate sequences (of maximum length 32) which maximize the set of objectives $\{R_1, \ldots, R_d\}$. Each reward R_i is defined as the number of occurrences of a given pattern in the sequence. We consider two tasks: `regex_3` with patterns `["AC", "CV", "VA"]` (correlated objectives, i.e. multiple objectives can be maximized simultaneously) and `regex_3_conf` with patterns `["A", "C", "V"]` (conflicting rewards, i.e. one objective cannot be improved without making another worse). Note that as the reward depends only on the number of occurrences of the patterns, there can be multiple sequences corresponding to the same reward, making the notion of diversity important. We use the *edit distance* as the measure of distance between two candidates. We present further details in Appendix B.2. We use MOReinforce [26], Envelope-MOQ [41] and MOSoftQL (same formulation as MOReinforce and MOGFN-PC, but trained with the Soft Q-Learning [15] objective) as baselines.

In Table 1, we can observe that MOGFN-PC significantly outperforms all other methods in terms of the Hypervolume and R_2 distance metrics, indicating that it learns a better approximation of the true Pareto front. We also observe that MOGFN-PC results in significantly more diverse and high reward candidates. Figure 3 visualizes the Pareto front of candidates generated with MOGFN-PC. We observe that for `regex_3_conf`, the Pareto front is a plane as it is not possible to improve any two objectives simultaneously, whereas for `regex_3` MOGFN-PC discovers candidates that maximize multiple objectives simultaneously. We also observe that the MOReinforce and Envelope-MOQ baselines struggle in this task as the rewards are sparse.

5.2 Benchmark Tasks

5.2.1 QM9

In this section, we consider a small-molecule generation task based on the QM9 dataset [33]. We generate molecules atom-by-atom and bond-by-bond with up to 9 atoms and use 4 reward signals. The main reward is obtained via a proxy [46] trained on QM9 to predict the HOMO-LUMO gap. The auxiliary rewards are Synthetic Accessibility (SA score), a molecular weight target, and a molecular logP target. All rewards are normalized so that they are between 0 and 1. [3] We present the results in Table 2. We train MOGFN-PC and the baselines to generate 1M molecules and sample 128 molecules

[3] Since the gap proxy is an approximate model, it can output higher values than those found in its training set, meaning the gap reward can exceed 1; it is clipped at 2.

Table 2: Results for the atom-based QM9 task

Algorithm	Reward (↑)	Diversity (↑)	HV (↑)	R_2 (↓)
MOGFN-PC	$0.76_{\pm 0.00}$	$0.93_{\pm 0.00}$	$1.40_{\pm 0.18}$	$2.44_{\pm 1.88}$
MOA2C	$0.61_{\pm 0.05}$	$0.39_{\pm 0.28}$	$1.16_{\pm 0.08}$	$6.28_{\pm 0.67}$
Envelope QL	$0.65_{\pm 0.06}$	$0.85_{\pm 0.01}$	$1.26_{\pm 0.05}$	$5.80_{\pm 0.20}$
MOREINFORCE	$0.57_{\pm 0.12}$	$0.53_{\pm 0.08}$	$1.35_{\pm 0.01}$	$4.65_{\pm 0.03}$

Table 3: Results for the Fragment-based Molecule Generation Task

Algorithm	Reward (↑)	Diversity (↑)	HV (↑)	R_2 (↓)
MOReinforce [26]	$0.41_{\pm 0.07}$	$0.01_{\pm 0.007}$	$0_{\pm 0}$	$9.88_{\pm 1.06}$
MARS [40]	NA	NA	$0.85_{\pm 0.008}$	$1.94_{\pm 0.03}$
MOA2C [31]	$0.76_{\pm 0.16}$	$0.48_{\pm 0.39}$	$0.75_{\pm 0.01}$	$3.35_{\pm 0.02}$
Envelope QL [41]	$0.70_{\pm 0.10}$	$0.15_{\pm 0.05}$	$0.74_{\pm 0.01}$	$3.51_{\pm 0.10}$
MOGFN-PC	$0.89_{\pm 0.05}$	$0.75_{\pm 0.01}$	$0.90_{\pm 0.01}$	$1.86_{\pm 0.08}$

from a fixed set of test preferences and calculate the HV and R_2 indicators, as well as the Top-K Reward & Diversity score. We can see that MOGFN-PC outperforms all baselines in terms of Pareto performance and diverse candidate generation. We compare MOGFN against MOReinforce [26], MOA2C, which is a preference-conditioned based A2C algorithm [31], and Envelope Q-Learning [41].

5.2.2 Fragment-Based Molecule Generation

In this section, we consider the fragment-based [25] drug design task of [4], where the task is to generate molecules by linking fragments to form a junction tree [20]. The main reward function is obtained via a pretrained proxy, available from [4], trained on molecules docked with AutodockVina [37] for the sEH target. The other rewards are based on Synthetic Accessibility (SA), drug likeliness (QED), and a molecular weight target. We detail the reward construction in Appendix B.4. We perform experiments with these four objectives and report HV, R_2, top-k reward, and top-k diversity. We compare MOGFN against MARS [40], MOReinforce [26], MOA2C [31], and Envelope Q-Learning [41].

Similarly to QM9, we sample 128 molecules per preference by conditioning MOGFN-PC on a fixed set of test preferences and calculate Top-K Rewards, Top-K Diversity, and HV and R_2 indicators. We report our results in Table 3. We observe that MOGFN-PC is consistently outperforming baselines not only in terms of HV and R_2, but also candidate diversity score.

5.2.3 DNA Sequence Generation

We also considered the generation of DNA aptamers, that is single-stranded nucleotide sequences that are popular in biological polymer design due to their specificity and affinity as sensors in crowded biochemical environments [48, 8, 42, 22]. We generated sequences by adding one nucleobase (A, C, T or G) at a time, with a maximum length of 60 bases. For multi-objective optimization, we considered three objectives: the free energy of the secondary structured calculated with the software NUPACK [44], the number of base pairs and the inverse of the sequence length to favour shorter sequences. We use MOReinforce [26] as the baseline for comparison.

As in the other tasks, we evaluate MOGFN-PC and the baselines by calculating the hypervolume, Top-K rewards, Top-K diversity and R_2 distance. We report the results in Table 4. In this case, the largest hypervolume is obtained by the multi-objective RL algorithm [26]. However, it achieves so by finding a quasi-trivial solution with the pattern `GCGCGC...` for most lengths, yielding very low diversity. While MOGFN does not match the hypervolume of the best baseline, it obtains much higher diversity and Top-K rewards, while achieving a relatively high optimal hypervolume.

Table 4: Metrics obtained in the DNA sequence generation task by MOGFN-PC with different reward exponent β and the reinforcement learning baseline.

Algorithm	Reward (↑)	Diversity (↑)	HV (↑)	R_2 (↓)
MOGFN-PC ($\beta = 80$)	$0.663_{\pm 0.006}$	$18.846_{\pm 0.101}$	$0.490_{\pm 0.013}$	$2.500_{\pm 0.056}$
MOGFN-PC ($\beta = 60$)	$0.558_{\pm 0.004}$	$25.027_{\pm 0.184}$	$0.396_{\pm 0.011}$	$2.905_{\pm 0.176}$
MOReinforce [26]	$0.104_{\pm 0.002}$	$0.826_{\pm 0.003}$	$0.630_{\pm 0.021}$	$1.923_{\pm 0.006}$

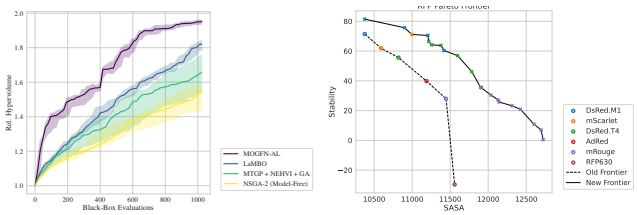

Figure 4: (Left) Relative Hypervolume Improvement and (Right) Pareto front of candidates by MOGFN-AL for the Proxy RFP active learning task.

Table 5: Diversity of candidates generated in Proxy RFP active learning task.

Algorithm	Diversity (↑)
NSGA-2	$0.07_{\pm 0.03}$
MTGP + NEHVI + GA	$0.12_{\pm 0.02}$
LaMBO	$0.18_{\pm 0.03}$
MOGFN	$0.25_{\pm 0.01}$

5.3 Active Learning

Finally, to evaluate MOGFN-AL, we study biological sequence design in an active learning setting. We consider the Proxy RFP task proposed by [36], to generate novel proteins with red fluorescence properties, optimizing for stability and solvent-accessible surface area. As discussed in section 4.2, we generate batches within each round of active learning by generating mutations for sequences in the set of non-dominated candidates in the current dataset. We adopt all the experimental details from [36], only switching the candidate generation with MOGFN-AL. We describe the details in Appendix B.6. We use LaMBO, NSGA-2 and and Model-based EA from [36] as baselines for comparison. We use the improvement in Hypervolume (relative to the initial dataset) and the diversity of generated candidates, measured by the average pairwise BLAST score (e-value).

We observe in Figure 4 (Left) that MOGFN-AL significantly outperforms the baselines in terms of the relative hypervolume improvement. In fact, MOGFN-AL matches the performance of LaMBO within half the number of black-box evaluations (evaluations of the true objective function). Table 5 shows that MOGFN-AL generates candidates that are more diverse than the candidates. We present the Pareto front of candidates generated by MOGFN-AL in Figure 4 (Right).

6 Conclusion

In this work, we extended GFlowNets for multi-objective optimization problems. In particular we considered two instantiations of MOGFN: MOGFN-PC which leverages reward-conditional GFlowNets proposed in [5] to model a family of single objective sub-problems derived from the MOO problem, and MOGFN-AL which sequentially solves a set of single-objective problems defined by multi-objective acquisition functions. Through a series of experiments on sequence and graph generation tasks, we empirically demonstrated the efficacy of MOGFNs in terms of generating diverse Pareto optimal candidates. Future work should focus on the development of preference-conditioned acquisition functions and consider applications of MOGFNs to other novel material discovery objectives.

References

[1] A. Abdolmaleki, S. Huang, L. Hasenclever, M. Neunert, F. Song, M. Zambelli, M. Martins, N. Heess, R. Hadsell, and M. Riedmiller. A distributional view on multi-objective policy optimization. In *International Conference on Machine Learning*, pages 11–22. PMLR, 2020.

[2] A. Abdolmaleki, S. H. Huang, G. Vezzani, B. Shahriari, J. T. Springenberg, S. Mishra, D. TB, A. Byravan, K. Bousmalis, A. Gyorgy, et al. On multi-objective policy optimization as a tool for reinforcement learning. *arXiv preprint arXiv:2106.08199*, 2021.

[3] S. Belakaria, A. Deshwal, and J. R. Doppa. Max-value entropy search for multi-objective bayesian optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[4] E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio. Flow network based generative models for non-iterative diverse candidate generation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[5] Y. Bengio, T. Deleu, E. J. Hu, S. Lahlou, M. Tiwari, and E. Bengio. Gflownet foundations, 2021.

[6] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[7] E. U. Choo and D. R. Atkins. Proper efficiency in nonconvex multicriteria programming. *Mathematics of Operations Research*, 8(3):467–470, 1983.

[8] D. R. Corey, M. J. Damha, and M. Manoharan. Challenges and opportunities for nucleic acid therapeutics. *nucleic acid therapeutics*, 32(1):8–13, 2022.

[9] S. Daulton, M. Balandat, and E. Bakshy. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. *Advances in Neural Information Processing Systems*, 33:9851–9864, 2020.

[10] S. Daulton, M. Balandat, and E. Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. *Advances in Neural Information Processing Systems*, 34:2187–2200, 2021.

[11] S. Daulton, D. Eriksson, M. Balandat, and E. Bakshy. Multi-objective bayesian optimization over high-dimensional search spaces. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[12] M. Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[13] M. T. Emmerich, A. H. Deutz, and J. W. Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2147–2154. IEEE, 2011.

[14] C. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1157–1163, 2006.

[15] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

[16] M. P. Hansen and A. Jaszkiewicz. *Evaluating the quality of approximations to the non-dominated set*. Citeseer, 1994.

[17] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–59, 2022.

[18] H. Ishibuchi, H. Masuda, Y. Tanigaki, and Y. Nojima. Modified distance calculation in generational distance and inverted generational distance. In A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello, editors, *Evolutionary Multi-Criterion Optimization*, pages 110–125, Cham, 2015. Springer International Publishing.

[19] M. Jain, E. Bengio, A. Hernandez-Garcia, J. Rector-Brooks, B. F. Dossou, C. A. Ekbote, J. Fu, T. Zhang, M. Kilgour, D. Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[20] W. Jin, R. Barzilay, and T. Jaakkola. Chapter 11. junction tree variational autoencoder for molecular graph generation. *Drug Discovery*, page 228–249, 2020.

[21] R. Keeney, H. Raiffa, K. L, and R. Meyer. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Wiley series in probability and mathematical statistics. Applied probability and statistics. Cambridge University Press, 1993.

[22] M. Kilgour, T. Liu, B. D. Walker, P. Ren, and L. Simine. E2edna: Simulation protocol for dna aptamers with ligands. *Journal of Chemical Information and Modeling*, 61(9):4139–4144, 2021.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[24] A. Konak, D. W. Coit, and A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety*, 91(9):992–1007, 2006.

[25] A. Kumar, A. Voet, and K. Y. Zhang. Fragment based drug design: from experimental to computational approaches. *Current medicinal chemistry*, 19(30):5128–5147, 2012.

[26] X. Lin, Z. Yang, and Q. Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2021.

[27] J. A. Luceño-Sánchez, A. M. Díez-Pascual, and R. Peña Capilla. Materials for photovoltaics: State of art and recent developments. *International journal of molecular sciences*, 20(4):976, 2019.

[28] N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio. Trajectory balance: Improved credit assignment in gflownets. *Neural Information Processing Systems (NeurIPS)*, 2022. To appear.

[29] K. Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

[30] S. Miret, V. S. Chua, M. Marder, M. Phiellip, N. Jain, and S. Majumdar. Neuroevolution-enhanced multi-objective optimization for mixed-precision quantization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1057–1065, 2022.

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. 2016.

[32] B. Paria, K. Kandasamy, and B. Póczos. A flexible framework for multi-objective bayesian optimization using random scalarizations. In *Uncertainty in Artificial Intelligence*, pages 766–776. PMLR, 2020.

[33] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[34] M. Reymond, E. Bargiacchi, and A. Nowe. Pareto conditioned networks. In *21st International Conference on Autonomous Agents and Multi-agent System*. IFAAMAS, 2022.

[35] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[36] S. Stanton, W. Maddox, N. Gruver, P. Maffettone, E. Delaney, P. Greenside, and A. G. Wilson. Accelerating Bayesian optimization for biological sequence design with denoising autoencoders. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20459–20478. PMLR, 17–23 Jul 2022.

[37] O. Trott and A. J. Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–461, 2010.

[38] K. Van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[39] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[40] Y. Xie, C. Shi, H. Zhou, Y. Yang, W. Zhang, Y. Yu, and L. Li. {MARS}: Markov molecular sampling for multi-objective drug discovery. In *International Conference on Learning Representations*, 2021.

[41] R. Yang, X. Sun, and K. Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in Neural Information Processing Systems*, 32, 2019.

[42] J. D. Yesselman, D. Eiler, E. D. Carlson, M. R. Gotrik, A. E. d'Aquino, A. N. Ooms, W. Kladwang, P. D. Carlson, X. Shi, D. A. Costantino, et al. Computational design of three-dimensional rna structure and function. *Nature nanotechnology*, 14(9):866–873, 2019.

[43] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. *Advances in neural information processing systems*, 32, 2019.

[44] J. N. Zadeh, C. D. Steenberg, J. S. Bois, B. R. Wolfe, M. B. Pierce, A. R. Khan, R. M. Dirks, and N. A. Pierce. Nupack: Analysis and design of nucleic acid systems. *Journal of computational chemistry*, 32(1):170–173, 2011.

[45] R. Zhang and D. Golovin. Random hypervolume scalarizations for provable multi-objective black box optimization. In *International Conference on Machine Learning*, pages 11096–11105. PMLR, 2020.

[46] S. Zhang, Y. Liu, and L. Xie. Molecular mechanics-driven graph neural network with multiplex graph for molecular structures, 2020.

[47] Y. Zhao, L. Wang, K. Yang, T. Zhang, T. Guo, and Y. Tian. Multi-objective optimization by learning space partition. In *International Conference on Learning Representations*, 2022.

[48] W. Zhou, R. Saran, and J. Liu. Metal sensing by dna. *Chemical reviews*, 117(12):8272–8325, 2017.

[49] M. Zuluaga, G. Sergent, A. Krause, and M. Püschel. Active learning for multi-objective optimization. In *International Conference on Machine Learning*, pages 462–470. PMLR, 2013.