GLOBAL-ORDER GFLOWNETS

Lluís Pastor Pérez *
Department of Computer Science
ETH Zürich
lpastor@ethz.ch

Javier Alonso García
Sony Europe B.V.
Stuttgart, Germany
javier.alonso@sony.com

Lukas Mauch
Sony Europe B.V.
Stuttgart, Germany
lukas.mauch@sony.com

ABSTRACT

Order-Preserving (OP) GFlowNets have demonstrated remarkable success in tackling complex multi-objective (MOO) black-box optimization problems using stochastic optimization techniques. Specifically, they can be trained online to efficiently sample diverse candidates near the Pareto front. A key advantage of OP GFlowNets is their ability to impose a local order on training samples based on Pareto dominance, eliminating the need for scalarization – a common requirement in other approaches like Preference-Conditional GFlowNets. However, we identify an important limitation of OP GFlowNets: imposing this local order on training samples can lead to conflicting optimization objectives. To address this issue, we introduce Global-Order GFlowNets, which transform the local order into a global one, thereby resolving these conflicts. Our experimental evaluations on various benchmarks demonstrate a small but consistent performance improvement.

1 INTRODUCTION

Multi-objective optimization is a complex problem that arises in many fields, where multiple competing objectives must be optimized simultaneously. One of the biggest challenges lies in the black-box scenario, where the objectives can only be evaluated indirectly, without explicit knowledge of their underlying functions. In such cases, we often rely on Bayesian or stochastic optimization methods to navigate the solution space and to sample diverse candidates near the Pareto front Gunantara (2018).

Recently, GFlowNets have been introduced to solve challenging black-box optimization methods with great success. Numerous methods have been developed to adapt GFlowNets to MOO problems, addressing the challenge of multi-dimensional rewards and Pareto dominance. In these problems, rewards are often conflicting, meaning that improving one objective may worsen another. For example, in Neural Architecture Search (NAS), higher training accuracy often comes with a larger number of parameters, leading to increased latency and computational cost.

One such approach are OP GFlowNets (OP-GFNs) Chen and Mauch (2024), which use new reward mechanism based on Pareto dominance across subsets. In this approach, the probability of choosing a sample is either uniformly distributed across the Pareto set or zero otherwise. The learned reward is designed to preserve the ordering of utility, minimizing the Kullback-Leibler (KL) divergence Joyce (2011) between the true Pareto distribution and the learned distribution. This ensures that the generated samples respect the Pareto dominance structure.

We identified one major shortcoming of OP-GFNs, namely: "The way how OP GFNs impose the local order on the training samples can lead to conflicting training objectives."

We summarize the contributions of this paper: 1) We propose two new approaches to convert the local order into a global one, defining a new training paradigm for GFlowNets, namely, Global-Order GFlowNets. These methods differ in computational complexity and offer distinct orders, yet both are

*Work done in an internship at Sony Europe B.V. Main author.

consistent with the local relation of Pareto dominance. 2) Using a simple example, we demonstrate how the restrictive local order used in OP-GFNs can lead to inconsistencies. 3) We test our methods on different benchmarks, achieving comparable results and, in some cases, improving the performance.

2 Related Work

2.1 Multi-Objective Optimization

Multi-Objective Optimization (MOO), also known as Pareto optimization, deals with problems involving more than one objective function to be optimized simultaneously. Without loss of generality, we consider the problem of maximizing a function. Formally, a MOO problem is defined as follows:

Definition 1 (MOO problem)**.** *Let $\mathcal{X}$ be the data (solution) space, $\mathcal{Y}$ the d-dimensional objective space and $F = (f_1, f_2, ..., f_d) : \mathcal{X} \to \mathcal{Y}$. Then, the MOO problem associated with this setup is*

$$\text{argmax}_{x \in \mathcal{X}} (f_1(x), f_2(x), ..., f_d(x))$$

The essence of MOO is to optimize multiple objectives simultaneously, with no objective being more important than another. Unlike single-objective optimization, MOO aims to find a set of optimal solutions, known as the Pareto front, which represents trade-offs among the objectives.

Definition 2 (Pareto Dominance and Pareto Set)**.** *Let $\mathcal{X}$, $\mathcal{Y}$, and $f_1, f_2, \ldots, f_d$ be as defined above. For $x, x' \in \mathcal{X}$, x Pareto dominates (or simply dominates) x' if:*

1. *For every $i \in \{1, \ldots, d\}$, $f_i(x) \geq f_i(x')$.*
2. *There is a $j \in \{1, \ldots, d\}$ such that $f_j(x) > f_j(x')$.*

An $x \in \mathcal{X}$ is non-dominated if $\nexists x' \in \mathcal{X}$ such that x' dominates x. The set $\mathcal{P}_\mathcal{X}$, consisting of non-dominated samples, is the Pareto set. The images under $f_1, ...f_d$ of all $x \in \mathcal{P}_\mathcal{X}$ is the Pareto front.

We consider a Multi-Objective Optimization (MOO) in a black-box scenario, which means that the objective functions $f_i(x)$ are not known analytically and can only be evaluated through stochastic optimization. Our primary interest lies in sampling diverse candidates near the Pareto front with high probability. This requires efficient stochastic optimization methods to navigate the uncertainty inherent in this black-box setup.

2.2 Performance measures for Multi-Objective Optimization

MOO algorithms must address two critical aspects: 1) **Fidelity**: Ensuring that sampled solutions are close to the true Pareto front, indicating a good approximation of the optimal trade-offs among the objectives. This is typically measured by distance-based metrics such as the *Inverted Generational Distance (IGD^+)* and the *Averaged Hausdorff Distance (d_H)* Ishibuchi et al. (2015); Coello Coello and Reyes Sierra (2004). 2) **Coverage**: Guaranteeing that we cover a significant portion of the entire Pareto front, providing a comprehensive understanding of the optimal solution space. This is typically measured by diversity and spread metrics such as the *Hypervolume Indicator (HV)*, *Pareto Coverage*, and *Pareto-Clusters Entropy (PC-ent)* Zitzler et al. (2007); Roy et al. (2023). We employ more metrics to compare the different methods, with details available in Appendix B.

2.3 GFlowNets for multi-objective optimization

Generative Flow Networks (GFlowNets) Bengio et al. (2021; 2023) have emerged as a novel class of probabilistic generative models. GFlowNets are controllable, in the sense that they can generate samples proportional to a given reward function $x \propto R(x)$. Compared to MCMC methods, they amortize the sampling in a training step and, hence, are more efficient Andrieu et al. (2003).

More specifically, GFlowNets decompose the sampling process into sequences of actions that are applied to, and that modify, an object s_0, forming a trajectory of partial objects $\tau = (s_0, s_1, ..., s_n)$ that terminate in an observation $x = s_n$. GFlowNets are typically parametrized by a triplet $P_F(s_t|s_{t-1}; \theta)$, $P_B(s_{t-1}|s_t; \theta)$ and Z_θ, namely the forward policy, the backward policy and the partition function.

By forcing P_F, P_B and Z_θ into balance over a set of training trajectories, i.e. by enforcing for each trajectory that

$$R(x)\prod_{t=1}^{n} P_B(s_{t-1}|s_t;\theta) = Z_\theta \prod_{t=1}^{n} P_F(s_t|s_{t-1};\theta), \tag{1}$$

we assure that objects x are generated from s_0 with $P(x;\theta) = \frac{R(x)}{\sum_{\forall x' \in \mathcal{X}} R(x')}$, when adhering to the forward policy P_F.

Similar to MCMC methods, GFlowNets can be easily applied for stochastic optimization. Let $f(x)$ be an objective function we want to maximize, we can choose $R(x) = \exp(\beta f(x))$ to sample candidates that maximize $f(x)$ proportionally more often Kirkpatrick et al. (1983). Here, $\beta \in \mathbb{R}^+$ is a temperature term that controls the variance.

There are different methods to adapt GFlowNets to MOO problems, One approach is Preference-Conditional GFlowNets (PC-GFNs) Jain et al. (2023), which use scalarization on the rewards. More specifically, scalarization means choosing preferences $w : R_w(x) = w^\top F(x)$, where $\mathbb{1}^\top w = 1, w_k \geq 0$. The scalarized objective is then used as a reward signal for a conditional GFlowNet. Of course, the choice of w strongly influences the shape of $R_w(x)$ and therefore the regions in which the GFlowNet will generate the most samples. Goal-Conditioned GFlowNets (GC-GFNs) Roy et al. (2023) tried to control the generation and introduced different focus regions that can steer the generation process.

The Order-Preserving (OP) GFlowNet Chen and Mauch (2024) is a variant that is particularly useful for MOO. It does not rely on a predefined reward signal $R(x)$ as it introduces a new reward mechanism based on Pareto dominance across subsets of $\mathcal{X}$. More specifically, let $\mathcal{X}' \subset \mathcal{X}$, an OP GFlowNet defines the probability of a sample $x \in \mathcal{X}'$ to be in the Pareto set $\mathcal{P}_{\mathcal{X}'}$ of the given subset $\mathcal{X}'$ as

$$P(x|\mathcal{X}') = \frac{\mathbf{1}[x \in \mathcal{P}_{\mathcal{X}'}]}{|\mathcal{P}_{\mathcal{X}'}|}, \tag{2}$$

where $\mathbf{1}[\cdot]$ is the indicator function.

An OP GFlowNet is trained to approximate $P(x|\mathcal{X}')$ for any subset $\mathcal{X}'$, i.e., to sample uniformly from the Pareto set of any subset $\mathcal{X}'$. Let $R(x;\theta)$ be the reward that is implicitly defined by Eq.1 for given P_F, P_B and Z_θ, OP GFlowNets minimize the Kullback-Leibler divergence

$$\mathcal{L}_{\text{OP}}(\mathcal{X}') = \text{KL}(P(x|\mathcal{X}')\|P(x|\mathcal{X}';\theta)), \tag{3}$$

across the subsets $\mathcal{X}' \subset \mathcal{X}$, where the conditional $P(x|\mathcal{X}';\theta) = \frac{R(x;\theta)}{\sum_{x' \in \mathcal{X}'} R(x';\theta)}$, $\forall x \in \mathcal{X}'$. The OP GFlowNet then learns a distribution $P(x;\theta)$ over the full set $\mathcal{X}$, that is consistent with all marginals, i.e.

$$P(x;\theta) = P(x|\mathcal{X}';\theta)P(\mathcal{X}';\theta),\ \forall \mathcal{X}' \subseteq \mathcal{X}. \tag{4}$$

Note, that for all subsets $\mathcal{X}' : \mathcal{P}_{\mathcal{X}} \cap \mathcal{X}' = \emptyset \quad \Rightarrow \quad P(\mathcal{X}';\theta) = 0$ such that the conditionals $P(x|\mathcal{X}';\theta)$ are consistent with $P(x;\theta)$. For scalar optimization problems with a single optimum, a possible $P(x;\theta)$ with the required properties exists and is the limit of the softmax $P(x;\theta) = \lim_{\gamma\to\infty} \frac{e^{\gamma g(f(x))}}{\sum_{x' \in \mathcal{X}} e^{\gamma g(f(x'))}}$, where $g(\cdot)$ is a monotonically increasing function that preserves the order on the image of f, that is, $f(x_1) \leq f(x_2) \Rightarrow g(f(x_1)) \leq g(f(x_2))$.

OP-GFNs' main advantage is that their loss function does not need the definition of a scalar reward. They are trained to sample proportional to given target distributions in the subset $\mathcal{X}'$ of the solution space. These target distributions are uniform over the Pareto set of these subsets, and 0 otherwise.

3 METHOD

3.1 LOCAL ORDER DILEMMA OF OP GFNS

We noticed that different to the scalar case, training with the target distribution across subsets that has been proposed for OP GFNs can lead to inconsistencies that cannot be resolved.

Figure 1 shows a simple failure case, where we want to maximize two objective functions $F = (f_1, f_2)$ over the set $\mathcal{X} = \{x_1, x_2, x_3, x_4\}$. Ideally, we want to learn a $P(x; \theta)$ that is uniform over the Pareto set $\mathcal{P}_\mathcal{X} = \{x_3, x_4\}$ and zero otherwise, by fitting the conditionals over the subsets $\mathcal{X}_1 = \{x_2, x_4\}$, $\mathcal{X}_2 = \{x_2, x_3\}$, $\mathcal{X}_3 = \{x_3, x_4\}$, using the target distributions from Eq. 2.

However, for this specific example we cannot construct a $P(x)$ that is consistent with all possible conditional reference distributions. More specifically, we recognize that each subset contains at least one Pareto optimal point, meaning that none of the marginals $P(\mathcal{X}_k),\ k = 1, 2, 3$ are allowed to vanish. Further, Eq. 2 imposes the conditions: 1) $P(x_2) = P(x_4) = 0.5$, 2) $P(x_3) = P(x_4) = 0.5$ and 3) $P(x_2|\mathcal{X}_2) = 0$. The last condition leads to a contradiction, because $P(\mathcal{X}_2) \neq 0$, yielding $P(x_2) = 0$.

This problem arises, because we do not allow the probability of a point within the Pareto set of $\mathcal{X}_k$ to vanish. However, this is necessary in order to resolve this contradiction. In the following, we resolve this issue by: 1) Defining a global order over the elements of $\mathcal{X}$ that is consistent with the Pareto dominance. 2) Training the GFlowNet to sample proportional to this global order.

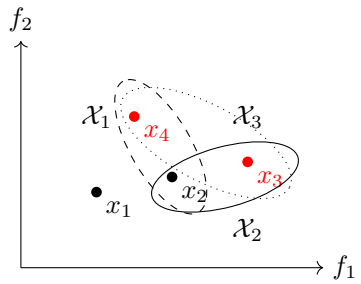

Figure 1: An example for the local order dilemma of OP GFNs. We can construct three different subsets $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{X}_3$, for which we can no longer construct $P(x; \theta)$ that is consistent with all conditionals.

3.2 Global Orders

This contradiction proves the need for a global ordering perspective. By assigning a global $\hat{R}$ that respects Pareto's dominance universally, rather than subset-by-subset basis, we completely avoid such inconsistencies. Our goal is to define a global ordering function $\hat{R}$ that assigns a rank or score to each point $x \in X$, consistently reflecting Pareto dominance, so that it does not depend on any subset, but on the seen data $\mathcal{X}$. Specifically, if x_1 dominates x_2, then we must have $\hat{R}(x_1) > \hat{R}(x_2)$. Contrarily to OP-GFNs, if x_1 and x_2 are not comparable, it is up to the global order to decide whether $\hat{R}(x_1)$ is greater, equal, or lower than $\hat{R}(x_2)$. By introducing a global perspective, we avoid the contradictions that arise when partial orders are combined from multiple subsets. We propose two main methods for defining such global order, and we name Global-Order GFlowNets to GFlowNets with a global ordering.

3.2.1 Global Rank

The Global Rank method (Algorithm 1) iteratively identifies Pareto fronts within the dataset D, assigning integer ranks that show how close each point is to the Pareto front. We first find the Pareto front of D and assign it the lowest temporary rank. After removing these points, we find the next Pareto front from the remaining set and assign it the next integer rank, and so on, until all points have been assigned a rank (or earlier if we decide to stop the process). In the end, we invert the ranking so that points on the first Pareto front receive the highest $\hat{R}$ values.

Although Global Rank guarantees consistency with Pareto dominance, it can be computationally expensive since it repeatedly computes Pareto fronts. For large-scale problems, the number of iterations may be capped, assigning a default minimal rank to all remaining points after a cutoff. This slightly relaxes theoretical guarantees for very poor solutions, but provides substantial computational savings.

Algorithm 1 GLOBAL RANK

Require: Dataset D, objective functions $f_1, \ldots, f_d$

$i \leftarrow 0$
$D_{\text{orig}} \leftarrow D$
while $|D| > 0$ **do**
  $P_{\text{front}} \leftarrow \texttt{Compute\_Pareto}(D, f_1, \ldots, f_d)$
  for $p \in P_{\text{front}}$ **do**
    $rank(p) \leftarrow i$ ▷ Assign current rank to Pareto front points
  end for
  $D \leftarrow D \setminus P_{\text{front}}$
  $i \leftarrow i + 1$
end while
$max\_rank \leftarrow i$ ▷ Maximum assigned rank
for $p \in D_{\text{orig}}$ **do**
  $\hat{R}(p) \leftarrow max\_rank - rank(p)$ ▷ Invert ranks
end for
return $\hat{R}$ ▷ Return the global ranks for all points in D_{orig}

Algorithm 2 NEAREST NEIGHBOR ORDER

Require: Dataset D, objective functions $f_1, \ldots, f_d$, distance metric $d(\cdot, \cdot)$

$P \leftarrow \texttt{Compute\_Pareto}(D, f_1, \ldots, f_d)$
for $p \in P$ **do**
  $\hat{R}(p) \leftarrow 0$ ▷ Points on the Pareto front have $\hat{R} = 0$
end for
for $x \in D \setminus P$ **do**
  $distances \leftarrow []$ ▷ Collect distances to all Pareto front points
  for $p \in P$ **do**
    $distances$.append$(d(x, p))$
  end for
  $\hat{R}(x) \leftarrow -\min(distances)$ ▷ Assign rank as negative min. distance to Pareto front
end for
return $\hat{R}$ ▷ Return $\hat{R}$ assignments for all points in D

Once $\hat{R}$ is defined through global ranking, we can train the GFlowNet such that $R(X; \theta)$ approximates either $\hat{R}(X) = (\hat{R}(x_1), \ldots, \hat{R}(x_B))$, its softmax transformation, or the indicator function $\mathbf{1}_{u=\max(\hat{R}(X))}[\hat{R}(X)]$. The choice depends on whether we prioritize uniform exploration along Pareto fronts (for example, by using softmax) or place a stricter focus on top-ranked points.

3.2.2 NEAREST NEIGHBOR ORDER

The Nearest Neighbor Order (Algorithm 2) is an alternative global ranking approach that bases a point's score on its distance to the Pareto front. Points on the Pareto front receive $\hat{R} = 0$, and all other points receive negative values proportional to their minimal distance to any Pareto optimal point.

We first compute the Pareto front P of D. For each $x \notin P$, we measure the distance $d(x, P) = \min_{p \in P} d(x, p)$. We then set $\hat{R}(x) = -d(x, P)$, while $\hat{R}(p) = 0$ for $p \in P$. Alternatively, the distance can also be computed by interpolating along the Pareto front.

Normalization of reward scales is critical here to avoid bias in the distance computations. Additionally, this approach may be less suitable if rewards differ significantly in their scales or distributions.

We show an example on how these orders, although defined under the same principle, can result in largely different reward distributions. Consider the discretization of the square with (target) rewards $r : [0, 1] \times [0, 1] \to \mathbb{R}^2$, $r(x, y) = (\pi x \cos(\pi x),\ \pi y \sin(\pi y))$. After applying the two orders, we obtain completely different values of $\hat{R}$, as shown in Figure 2.

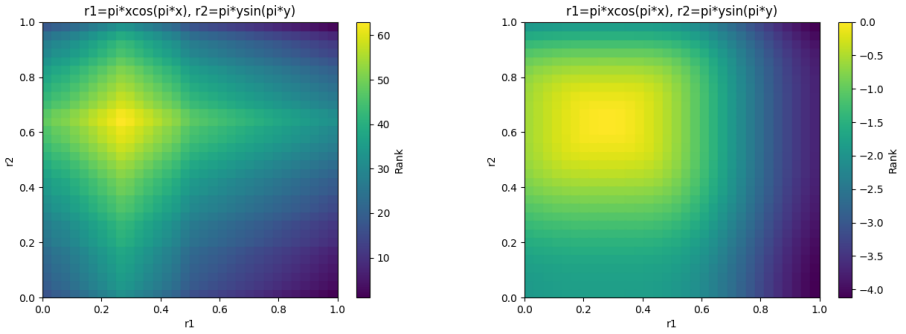

Figure 2: $\hat{R}$ computed with Global Rank (left) and Nearest Neighbor (right) algorithms

4 EXPERIMENTS

In this section, we evaluate our different Global-Order GFlowNets across various benchmarks to demonstrate improvements over previous results. The definition of the evaluation metrics can be found in Appendix B. We begin this section explaining the usage of replay buffers. Then, we present results for key benchmarks such as Fragment-based molecule generation, QM9, and DNA, detailing each benchmark's importance, methods, parameters, and comparisons with other algorithms. For the synthetic benchmarks, HyperGrid and N-Grams, we have included the results and detailed analysis in Appendix C. Our methods are marked with an asterisk (*) in all tables.

4.0.1 REPLAY BUFFER AND CHEAP GFLOWNETS

In MOO, traditional training methods often rely solely on the samples generated at each step, leading to unstable training and convergence issues. To address this, most experiments incorporate a replay buffer, where newly generated trajectories are stored and randomly sampled for training, stabilizing the learning process.

In single-objective scenarios, it is common to sample from both high- and low-reward experiences, while in MOO, samples are drawn from the Pareto front of all observed samples. We implement a warm-up phase to avoid the network getting stuck in a local optima.

The training process involves drawing new samples, updating the Pareto front, and drawing batches from both the buffer and the Pareto front, ensuring at least $k \in \mathbb{N}$ Pareto optimal samples are included. This approach allows for an efficient and cost-effective variant of Global Rank GFlowNets, named Cheap-GR-GFNs, by focusing only on the actual Pareto front, significantly reducing computational costs.

From now on, we refer to our methods as follows: Global-Rank GFlowNets (GR-GFNs), Trimmed Global-Rank GFlowNets with a maximum rank k (GR-GFNs (k)), Cheap Global-Rank GFlowNets (Cheap GR-GFNs), Nearest Neighbor GFlowNets (NN-GFNs), and their variant with linear interpolation of the Pareto Front (NN-int-GFNs). Due to the amount of different experiments, most of the plots and the tables can be found in Appendix C.

4.1 DNA SEQUENCE GENERATION

GFlowNets function as samplers of trajectories, making them naturally suited for addressing various problems in biology and chemistry. In this task, we generate DNA sequences by adding one nucleobase at a time: adenine (A), cytosine (C), guanine (G), or thymine (T) Zhou et al. (2017); Corey et al. (2022); Yesselman et al. (2019); Kilgour et al. (2021). In this work, we evaluate `energy-pairs` and `energy-pins-pairs`, whose details can be found in Appendix C.3. For consistency and due to the similarity of the problem structure, we compare our approach using the same algorithms (PC-GFNs and OP-GFNs) and parameters as in the N-Grams task found in Appendix C.2. Following the advice of Jain et al. (2023); Chen and Mauch (2024), we set $\beta = 80$ for PC-GFNs.

4.1.1 RESULTS

The results are presented in Table 1. For the two rewards case we see that the three methods perform optimally, but for the three rewards, ours clearly outperform the other two, except for the Top-k diversity. In Chen and Mauch (2024), authors claim that their method does not sample as close to the Pareto front as PC-GFNs, but their method explores more. This is repeated here with our method, now exploring more, yet getting a slightly worse Pareto front in `Energy-Pairs`, as shown in Figure 3.

Table 1: Results for DNA benchmark

Energy - Pairs	HV (↑)	R_2 (↓)	PC-ent (↑)	$d_H(P', P)$ (↓)	Diversity (↑)
OP-GFNs	**0.35**	**2.65**	**0.00**	**0.53**	3.98
PC-GFNs	**0.35**	**2.65**	**0.00**	**0.53**	5.27
Cheap-GR-GFNs*	**0.35**	**2.65**	**0.00**	**0.53**	**6.97**
Energy - Pins - Pairs					
OP-GFNs	0.07	17.32	0.69	0.74	**11.5**
PC-GFNs	0.08	17.08	0.69	0.75	6.34
Cheap-GR-GFNs*	**0.11**	**14.44**	**1.39**	**0.67**	6.49

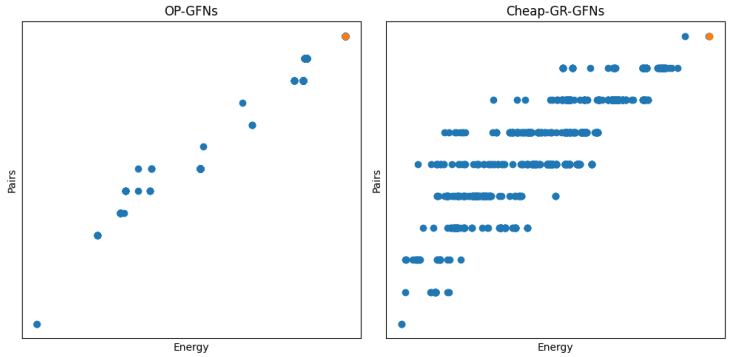

Figure 3: The 1280 generated samples, with 18 unique samples for OP-GFNs compared to 356 of Cheap-GR-GFNs

4.2 FRAGMENT-BASED MOLECULE GENERATION

We continue with the chemistry-themed benchmarks, particularly with the Fragment-based molecule generation from Kumar et al. (2012); Bengio et al. (2021), based on computational chemistry and Machine Learning, aimed at designing new molecules with desired properties. This task is significant in fields like pharmaceuticals, material science, and biochemistry, as new molecules can lead to new drugs, materials, and chemical processes. The objective functions are `SEH`,`QED`,`SA`, and `MW`, whose details can be found in Appendix C.4. We compare our methods Cheap-GR-GFNs and GR-GFNs with OP-GFNs, PC-GFNs and the Goal-Conditioned GFNs (GC-GFNs).

4.2.1 RESULTS

We show the 3200 generated candidates in Figure 10, and the evaluation metrics in Table 6. As with the metrics it is not clear to determine which method performs better; we also added a column indicating the following metric. We select all the Pareto fronts from each method as candidates, and we recompute the new Pareto front. Then, we count how many points for each method are not dominated. We show this in Figure 11. We see that for the cases of `QED-SA` and `SEH-SA`, our methods explore the `SA` function more than OP-GFNs, which tend to explore the other objective function. For `SEH-QED` and `SEH-MW`, our methods sample closer from the Pareto front. For the other two pairs, we do not find significant differences among the different methods because the Pareto front is easier to find.

4.3 QM9

A related challenge to Fragmentation-based molecule generation is the QM9 environment Ramakrishnan et al. (2014), where molecules are generated by sequentially adding atoms and bonds, with a maximum of 9 atoms. We explore objective functions such as `MXMNet` (HOMO-LUMO gap prediction), `logP`, `SA`, and a modified `MW` function, detailed in the Appendix C.5

4.3.1 RESULTS

The plots containing the 3200 generated candidates are presented in Figure 12, while the metrics used to evaluate them are shown in Table 7. Again we compute the number of non-dominated samples as in the previous benchmark. Remarkably, these metrics show improvements in most of the objective functions. To support this point, we plot the different Pareto fronts in Figure 13.

5 CONCLUSION

We introduced a new approach to adapt GFlowNets for MOO tasks, developing PC-GFNs and OP-GFNs, which create weighted rewards and local orders among samples, respectively. We combined these methods to define a global reward that imposes a global order, hence defining Global-Order GFlowNets.

In our experiments, Cheap-GR-GFNs and GR-GFNs performed on par with or better than PC-GFNs and OP-GFNs, especially in the HyperGrid, DNA, and QM9 benchmarks, achieving results closer to the ideal Pareto front. Cheap-GR-GFNs outperformed PC-GFNs in several N-Grams tests, particularly in challenging 4-unigram and 4-bigram cases. In the Fragmentation-based molecule generation benchmark, our methods showed strong performance, with GR-GFNs and OP-GFNs sampling closer to the Pareto front in several instances.

While NN-GFNs excelled in the HyperGrid problem, capturing the full Pareto front in specific cases, their performance declined in other benchmarks. Although no algorithm proved better performance across all tests, our proposed methods represent a competitive alternative, excelling in specific scenarios.

The choice of the global order depends on the problem. In high-dimensional or real-world cases, we recommend using Cheap-GR-GFNs as they keep the complexity of OP-GFNs. This is especially useful in real-world applications where computational efficiency is important, but also the ability to navigate multiple objectives.

5.1 LIMITATIONS AND FUTURE WORK

This work encompasses the concept of finding different global orders that are consistent with the local order given by the relation of Pareto dominance. Although we have discovered many different global orders, it is mandatory to remark that we have not found them all. More importantly, we have not established the optimality of any of our methods, and it remains possible that alternative global orders could outperform those presented in this work. Possible improvements in this matter are potential future approaches in this direction of research. We note that our methods do not surpass others in every aspect. However, in cases where they do not outperform, they either exhibit enhanced exploration capabilities (as shown in Figure 3) or maintain competitive results. Finally, we encountered a significant challenge with Botorch Balandat et al. (2020), where a bug in the Pareto computation function directly impacted the retrieval of Pareto points, substantially hindering our progress. However, we are pleased to report that, after notifying the developers, the issue has been resolved. Experiments were conducted on a system with 8× RTX 6000 Ada GPUs, and 2× AMD EPYC 7302 CPUs.

REFERENCES

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50:5–43, 2003.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27381–27394. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/e614f646836aaed9f89ce58e837e2310-Paper.pdf.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations, 2023.

G Richard Bickerton, Giorgio V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nat Chem*, 4(2):90–98, 2012. doi: 10.1038/nchem.1243.

Yihang Chen and Lukas Mauch. Order-preserving GFlownets. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VXDPXuq4oG.

Carlos A. Coello Coello and Margarita Reyes Sierra. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, and Humberto Sossa, editors, *MICAI 2004: Advances in Artificial Intelligence*, pages 688–697, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24694-7.

David R Corey, Masad J Damha, and Muthiah Manoharan. Challenges and opportunities for nucleic acid therapeutics. *Nucleic Acid Therapeutics*, 32(1):8–13, 2022.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J Cheminformatics*, 1:8, 2009.

C. Griñán-Ferré, S. Codony, E. Pujol, et al. Pharmacological inhibition of soluble epoxide hydrolase as a new therapy for alzheimer's disease. *Neurotherapeutics*, 17:1825–1835, 2020. doi: 10.1007/s13311-020-00854-1. URL https://doi.org/10.1007/s13311-020-00854-1.

Nyoman Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1):1502242, 2018. doi: 10.1080/23311916.2018.1502242. URL https://doi.org/10.1080/23311916.2018.1502242.

Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. 1998. URL https://api.semanticscholar.org/CorpusID:17531041.

Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified distance calculation in generational distance and inverted generational distance. In António Gaspar-Cunha, Carlos Henggeler Antunes, and Carlos Coello Coello, editors, *Evolutionary Multi-Criterion Optimization*, pages 110–125, Cham, 2015. Springer International Publishing. ISBN 978-3-319-15892-1.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-García, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective GFlownets, 2023. URL https://openreview.net/forum?id=3z1Ws6GEYV4.

James M. Joyce. *Kullback-Leibler Divergence*, pages 720–722. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_327. URL https://doi.org/10.1007/978-3-642-04898-2_327.

Michael Kilgour, Tao Liu, Brandon D Walker, Pengyu Ren, and Lena Simine. E2edna: Simulation protocol for dna aptamers with ligands. *Journal of Chemical Information and Modeling*, 61(9):4139–4144, 2021.

S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983.

Ashutosh Kumar, Arnout Voet, and Kam Y.J. Zhang. Fragment based drug design: from experimental to computational approaches. *Current Medicinal Chemistry*, 19(30):5128–5147, 2012.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):1–7, 2014.

RDKit: Open-source cheminformatics. Rdkit: Open-source cheminformatics. `https://www.rdkit.org`. Accessed: 2024-03-23.

Julien Roy, Pierre-Luc Bacon, Christopher Pal, and Emmanuel Bengio. Goal-conditioned gflownets for controllable multi-objective molecular design, 2023.

Samuel Stanton, Wesley Maddox, Nate Gruver, Phillip Maffettone, Emily Delaney, Peyton Greenside, and Andrew Gordon Wilson. Accelerating bayesian optimization for biological sequence design with denoising autoencoders, 2022.

David A. van Veldhuizen. Evolutionary computation and convergence to a pareto front. 1998. URL `https://api.semanticscholar.org/CorpusID:18585705`.

Joseph D Yesselman, Daniel Eiler, Erik D Carlson, Michael R Gotrik, Anne E d'Aquino, Alexandra N Ooms, Wipapat Kladwang, Paul D Carlson, Xuesong Shi, David A Costantino, et al. Computational design of three-dimensional rna structure and function. *Nature Nanotechnology*, 14(9):866–873, 2019.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/9d63484abb477c97640154d40595a3bb-Paper.pdf`.

Joseph N Zadeh, Conrad D Steenberg, Justin S Bois, Brian R Wolfe, Niles A Pierce, Asif R Khan, Robert M Dirks, and Niles A Pierce. Nupack: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, 32(1):170–173, 2011. doi: 10.1002/jcc.21596.

Shuo Zhang, Yang Liu, and Lei Xie. Molecular mechanics-driven graph neural network with multiplex graph for molecular structures, 2020.

Wenhu Zhou, Runjhun Saran, and Juewen Liu. Metal sensing by dna. *Chemical Reviews*, 117(12): 8272–8325, 2017.

Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, and Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, pages 862–876, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-70928-2.

A Global Orders

We first prove that the Global Rank algorithm is consistent with the local relation of Pareto Dominance.

Theorem 1 (Consistency of Global Rank with Pareto Dominance)**.** *Let $\mathcal{X}$ be a set of samples with associated rewards $R_1, R_2, ..., R_d$, and let $\hat{R} : \mathcal{X} \to \mathbb{R}$ be a ranking function produced by the Global Rank algorithm. For any two samples $x_1, x_2 \in \mathcal{X}$, if x_1 Pareto dominates x_2, then $\hat{R}(x_1) > \hat{R}(x_2)$.*

Proof. The first affirmation to notice is that while x_1 is still in D (i.e., it has not been assigned a rank yet), x_2 will be there as well, due to the fact that x_2 cannot be in the Pareto front as there is a point that dominates it (x_1), hence $\hat{R}(x_2) \leq \hat{R}(x_1)$. Also, it is important to notice that all points that dominate x_1, dominate x_2 as the Pareto dominance is a transitive relation. Therefore, when all the points that dominate x_1 have been assigned a rank, x_1 will be in the new Pareto front, so $\hat{R}(x_1) > \hat{R}(x_2)$. □

A.1 Comparison of Global Orders

To illustrate how the ranks are distributed across samples, we present different examples in Figure 5, where the data samples are the discretization of the square $[0, 1] \times [0, 1]$ in squares of size $1/32$ and the rewards are $r : (x, y) \to (x, y)$, $r : (x, y) \to (x, \frac{y}{1+x^2})$, $r : (x, y) \to (x, (e^{-(x-1/2)^2} + y)/2)$, and finally $r : (x, y) \to ((\pi x)cos(\pi x), (\pi y)sin(\pi y))$. We also show the distribution of the rewards across the different points of the square in Figure 4.

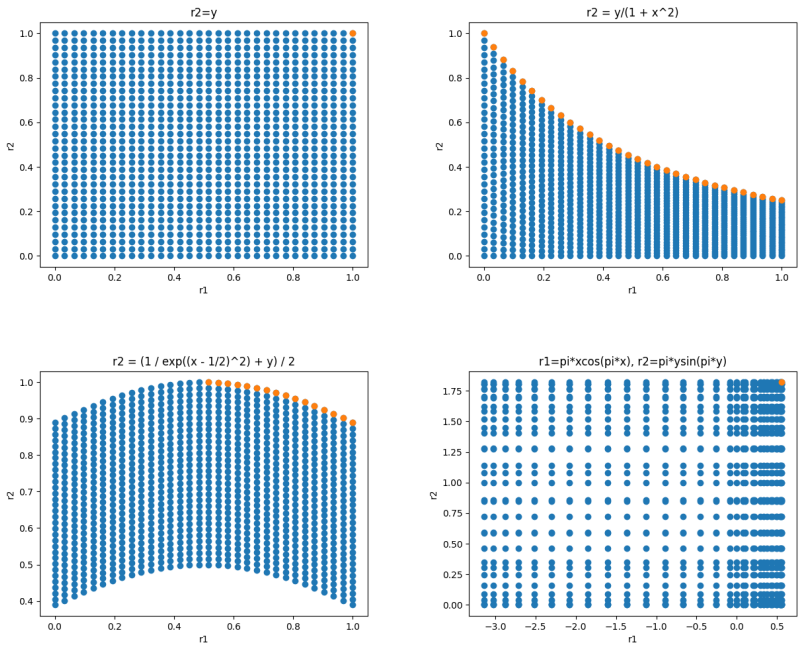

Figure 4: Image of the different points in the $[0, 1] \times [0, 1]$. In orange, we find the Pareto front. In the first cases, we omit $r_1(x, y) = x$ for clarity.

The first example to consider is the more straightforward case, where the rewards are the projections $r_1(x, y) = x$ and $r_2(x, y) = y$. In this setup, we observe that the image of the points is, trivially, uniformly distributed. Although the orders look similar, we can start to see some differences, especially when either x or y values are smaller. Similar behavior can be seen when $r : (x, y) \to (x, (e^{-(x-1/2)^2} + y)/2)$.

The case where we start to see some more notable differences is when using $r : (x, y) \to (x, \frac{y}{1+x^2})$, as when x increases, the images are remarkably more skewed towards the x axis. As the Nearest Neighbor Order takes into account the Euclidean distance to the Pareto front whilst the Global Rank one does not, we observe differences above all when points are assigned a lower rank.

The most significant difference lies in the fourth example, where more complicated rewards ($r(x, y) = ((\pi x)cos(\pi x), (\pi y)sin(\pi y))$) demonstrate that the assigned values in $\hat{R}$ can be completely different from each other.

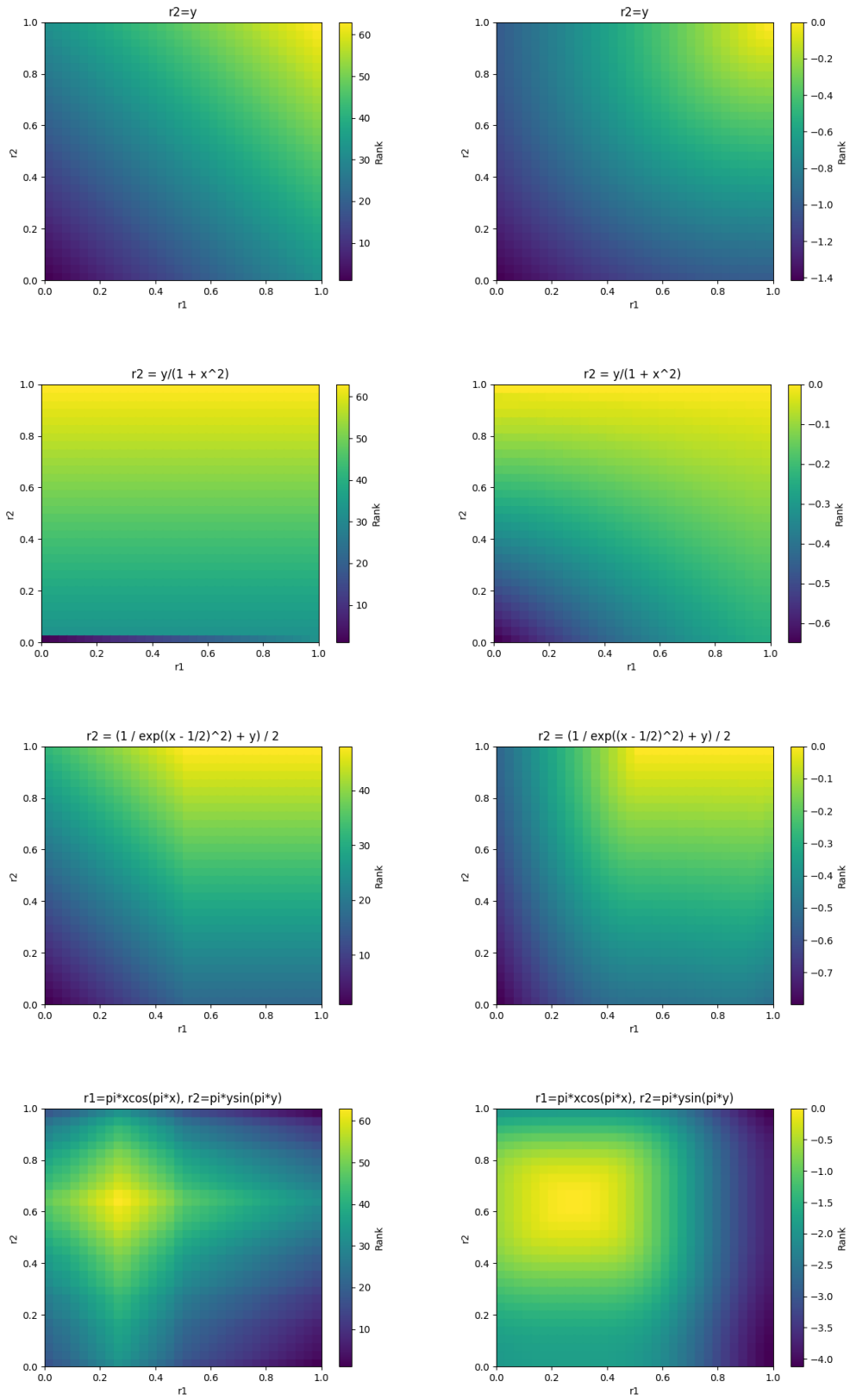

Figure 5: Comparison of the algorithms Global Rank (left) and Nearest Neighbor Order (right) with different rewards

B EVALUATION METRICS

In evaluating the performance of our proposed algorithms, we utilize a range of metrics designed to capture various aspects of solution quality, diversity, and convergence to the true Pareto front. To define the metrics we are going to evaluate our algorithms with, we first denote the true Pareto front points P, the generated candidates by the network S, and finally P', the Pareto front of S. For some problems, the true Pareto front P is not available or is unknown. However, since the functions are primarily normalized or bounded, we will use a discretization of the extreme faces of the hypercube as the reference set. The goal is to provide a comprehensive evaluation that accounts for both the closeness of generated solutions to the true Pareto front and the diversity of those solutions.

With the previous introduction, we introduce the following metrics:

- **Inverted Generational Distance**:

$$\text{IGD+}(S, P) := \frac{1}{|P|} \sum_{p \in P} (\min_{s \in S} ||p - s||^2).$$

  This metric, introduced in Ishibuchi et al. (2015), derives from previous metrics like the **Generational Distance** (GD) van Veldhuizen (1998) and the **Inverted Generational Distance** (IGD) Coello Coello and Reyes Sierra (2004). GD calculates the average distance from each candidate $s \in S$ to the Pareto front P. IGD, in contrast, averages this quantity over the points of P instead of the points of S. By focusing solely on the non-negative part of the distance, this method accounts for the directionality of improvements, avoiding penalties on solutions that exceed the Pareto front.

- **Averaged Hausdorff Distance (Plus)** is given by

$$d_H^+(S, P) = \max(\text{GD+}(S, P), \text{IGD+}(S, P)),$$

  where the **Generational Distance Plus** (GD+) is also introduced. Similar to IGD+, GD+ is defined as:

$$\text{GD+} = \frac{1}{|S|} \sum_{s \in S} (\min_{p \in P} ||p - s||^2).$$

  The standard Averaged Hausdorff Distance is

$$d_H(S, P) = \max(\text{GD}(S, P), \text{IGD}(S, P)).$$

- **Hypervolume Indicator** (HV) Zitzler et al. (2007) measures the quality of a set of solutions by computing the volume covered by the union of rectangles formed between the solutions provided by the network and a reference point. $\text{HV}(S, r)$ can be expressed as the Lebesgue measure of the union of rectangles formed by each point $s \in S$ and a reference point r, such that each rectangle is defined by the product of the intervals $[s_i, r_i]$ for $i \in \{1, ..., d\}$.

- **Pareto-Clusters Entropy** (PC-ent) Roy et al. (2023) metric focuses on the diversity of the generated (Pareto front) samples. Inspired by the formulation of entropy $H(X) = -\sum_{x \in X} x \log(x)$, this metric first creates clusters P'_j with points in P' based on distances to the reference Pareto front P. The PC-ent metric is defined as:

$$\text{PC-ent}(P, P') = -\sum_j \frac{|P'_j|}{|P|} \log \frac{|P'_j|}{|P|}.$$

- **R_2 Indicator** Hansen and Jaszkiewicz (1998) utilizes a set of uniformly distributed reference vectors alongside a utopian point z^*. We introduce the Uniform Reference Vectors Λ, which collects uniformly distributed vectors across the objective space, capturing all its directions. The formula for the R_2 Indicator is:

$$R_2(S, \Lambda, z^*) = \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \min_{s \in S} \max_{i \in \{1, ..., d\}} \{\lambda_i \cdot |z_i^* - s_i|\}.$$

- **Pareto Coverage** is used when the true Pareto front P is available, and it is defined as the proportion of the Pareto front that has been seen in S (or equivalently in P').

- **Samples in Front** is defined as the proportion of generated samples that are in the true Pareto front when the true Pareto front is available.
- **Top-k Divergence** quantifies the diversity among the best samples by measuring how different these samples are from each other, providing insights into the exploratory success of the sampling process in generating diverse high-quality candidates.

C EXPERIMENTS

Across all this section we refer to our experiments in the tables with a "*". Our methods are Global-Rank GFlowNets (GR-GFNs), Trimmed Global-Rank GFlowNets when we assign a maximum rank k (GR-GFNs (k)), Cheap Global-Rank GFlowNets (Cheap GR-GFNs), Nearest neighbor GFlowNets (NN-GFNs) and their version where we linearly interpolate the Pareto Front (NN-int-GFNs).

C.1 HYPERGRID

We investigate the synthetic HyperGrid environment introduced in Bengio et al. (2021). In this setup, states S form a d-dimensional grid with side length H, where each state is defined as $\{(s_1, \ldots, s_d) \mid s_i \in \{1, \ldots, H\}, i \in \{1, \ldots, d\}\}$. The environment allows actions that increment one coordinate without leaving the grid or stopping at a state. We evaluate four different reward functions: `branin`, `currin`, `shubert`, and `beale`, chosen for their diverse sparsity patterns in the Pareto front. Experiments are conducted with $d \in \{2, 3\}$ and $H = 32$.

The objective functions are defined as follows:

$$\texttt{branin}(x_1, x_2) = a\left(x_2 - bx_1^2 + cx_1 - r\right)^2 + s(1-t)\cos(x_1) + s,$$

where $a = 1$, $b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $r = 6$, $s = 10$, $t = \frac{1}{8\pi}$ and x_1, x_2 are scaled as $x_1 \leftarrow 15x_1 - 5$ and $x_2 \leftarrow 15x_2$

$$\texttt{currin}(x_1, x_2) = \left(1 - e^{-0.5x_2}\right) \cdot \frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{13.77(100x_1^3 + 500x_1^2 + 4x_1 + 20)}$$

$$\texttt{shubert}(x_1, x_2) = \frac{\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\sum_{i=1}^{5} i\cos((i+1)x_2 + i)}{397} + \frac{186.8}{397}$$

$$\texttt{beale}(x_1, x_2) = \frac{(1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_3^2)^2}{38.8}$$

For $d = 2$, we utilize Global Rank GFlowNets (GR-GFNs) and Nearest Neighbor GFlowNets (NN-GFNs), including both basic and interpolated versions. For $d = 3$, we also employ Global Rank (Trimmed) GFlowNets with a maximum rank of 25. Comparisons are made against Preference-Conditional GFlowNets (PC-GFNs) and Order-Preserving GFlowNets (OP-GFNs).

The learned GFlowNet sampler is used to generate 1280 candidates for evaluation.

C.1.1 NETWORK STRUCTURE AND TRAINING DETAILS

As in the rest of experiments, we use the guidance of Chen and Mauch (2024). The backward transition probability P_B is set to be uniform across all states. The forward transition probability P_F is parameterized by a 3-layer MLP, each with 64 hidden units and LeakyReLU as the activation function. The Adam optimizer is employed for training, using a learning rate of 0.01 and a batch size of 128. The model is trained over 1000 steps. Following guidance from Jain et al. (2023), for PC-GFNs the weight vector w is drawn from a Dirichlet(1.5) distribution, and β is sampled from a $\Gamma(16, 1)$ distribution during training. For evaluation, w is sampled from the same Dirichlet(1.5) distribution, but β is fixed at 16.

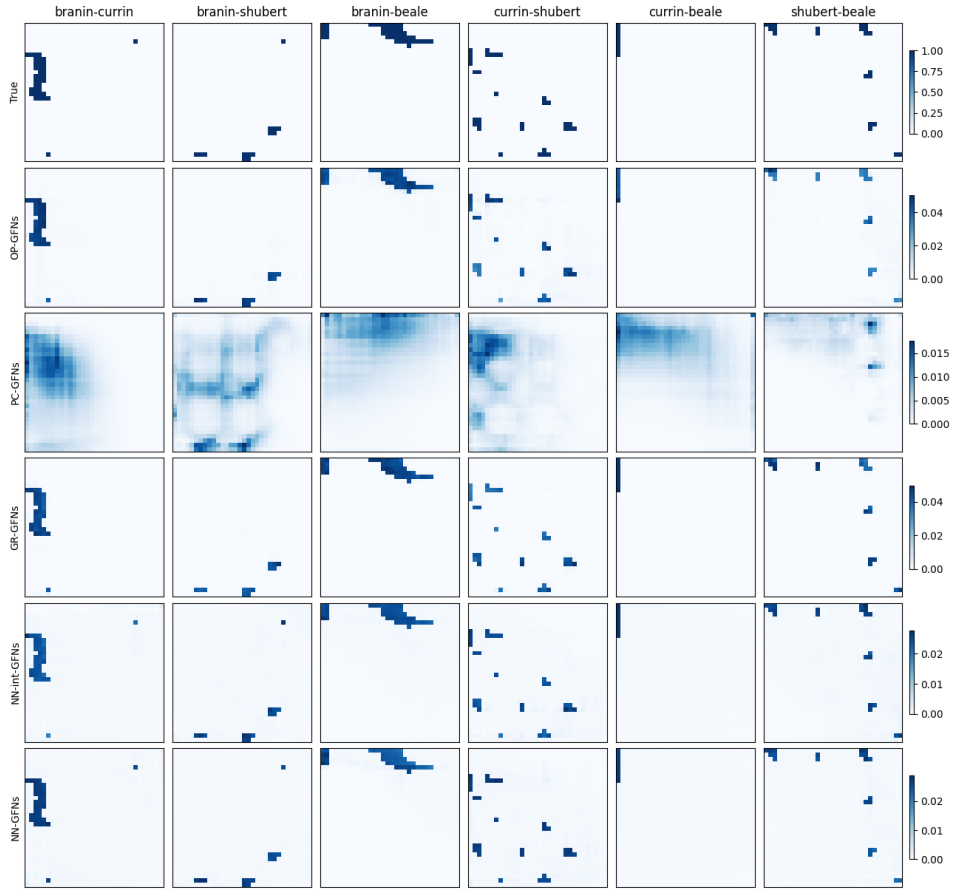

Figure 6: In the top row, we plot the indicator function of the true Pareto front. In the other rows, we plot the learned reward distribution of the different methods.

C.1.2 RESULTS

We first present the results for $d = 2$. Due to the low dimension, we can comfortably plot the results. First, we display the generated Pareto fronts in Figure 6, and then, we show how the different generated sample rewards are distributed in Figure 7. The results for the different metrics are summarized in Tables 2, 3.

From Figure 7 we can observe that GR-GFNs accomplish having most of the generated samples in the Pareto front, in contrast with NN-Gto NN-GFNs that, althoughg a lot of samples in the Pareto front, also havealso have manybserve that, especially for the case of the `shubert` functions, GR-GFNs generate much better results than the previous GFNs. We observe that in terms of d_H and Samples in front, GR-GFNs greatly surpass the other methods, whilst in IGD+ and Pareto Coverage, NN-GFNs are the best ones.

Our methods, compared to OP-GFNs, are very similar in terms of Pareto coverage, except for the cases with `branin-currin` and `branin-shubert`, where the isolated point in the top-right corner is only achieved with the NN-GFNs (both basic and interpolated).

We now present the results for $d = 3$ in Figure 8. Even with the expansion of a dimension, we still see that our methods outperform OP-GFNs and PC-GFNs in most of the metrics, especially in Samples in front where we appreciate the greatest difference with respect to the rest in the case of GR-GFNs. In fact, only in the Pareto coverage of the `branin-shubert-beale` functions, OP-GFNs are slightly better than the rest.

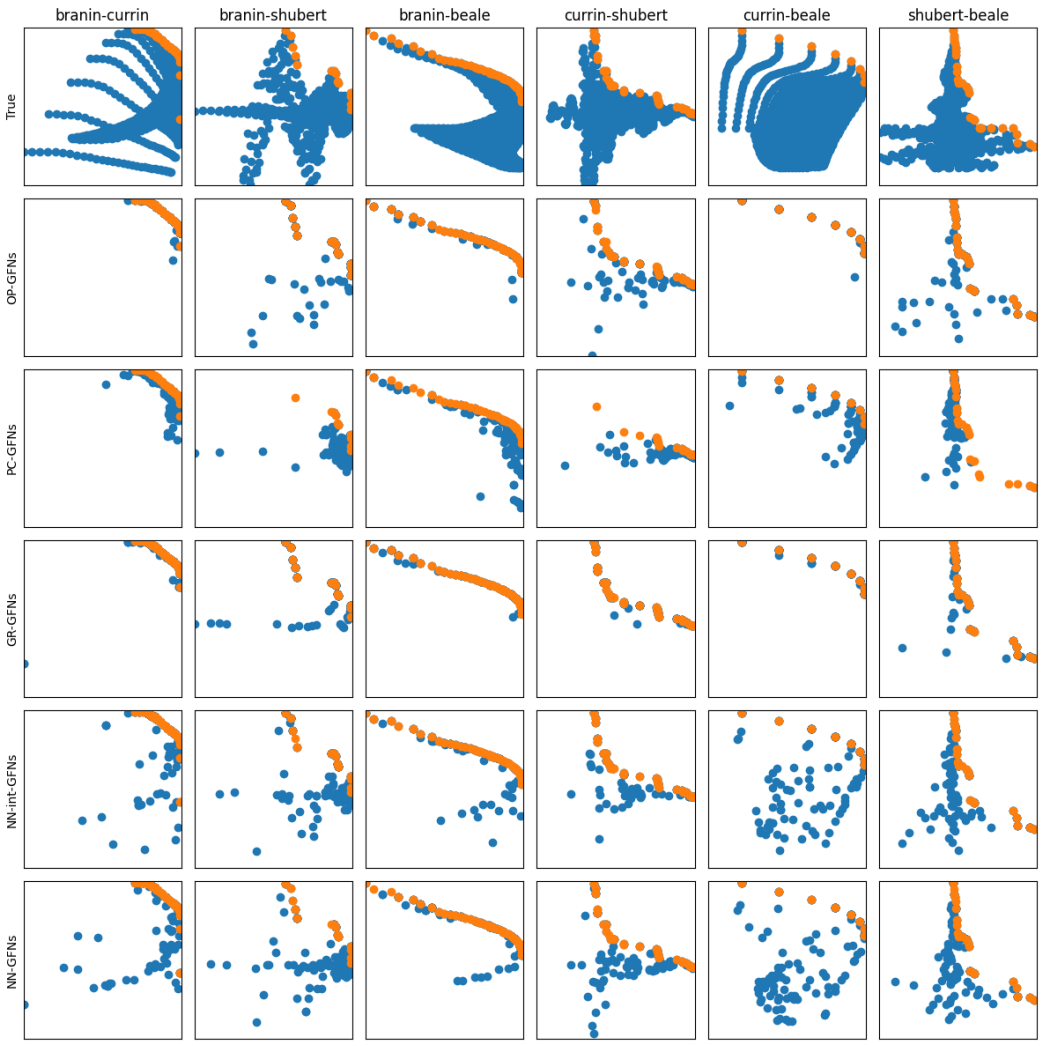

Figure 7: 128 generated candidates (blue) and their respective Pareto front (orange). The first row, being the ground truth, is the image of all possible values of the discretized grid

Table 2: HyperGrid Results for $d = 2$

Method	$\mathrm{d}_H(P', P)$ (↓)	IGD+ (↓)	PC-ent (↑)	% front (↑)	Coverage (↑)
branin-currin					
OP-GFNs	0.014	2.66×10^{-6}	3.555	0.889	0.972
PC-GFNs	0.041	2.66×10^{-6}	3.555	0.251	0.972
GR-GFNs*	**0.008**	2.66×10^{-6}	3.555	**0.935**	0.972
NN-int-GFNs*	0.06	**1.18e-08**	**3.584**	0.675	**1**
NN-GFNs*	0.063	**1.18e-08**	**3.584**	0.669	**1**
branin-shubert					
OP-GFNs	0.054	6.82×10^{-6}	2.565	0.736	0.929
PC-GFNs	0.078	2.60×10^{-2}	2.303	0.052	0.714
GR-GFNs*	**0.028**	6.82×10^{-6}	2.565	**0.823**	0.929
NN-int-GFNs*	0.098	**9.00e-09**	**2.639**	0.402	**1**
NN-GFNs*	0.095	**9.00e-09**	**2.639**	0.412	**1**
branin-beale					
OP-GFNs	0.005	**1.38e-08**	**3.784**	0.899	**1**
PC-GFNs	0.065	**1.38e-08**	**3.784**	0.225	**1**
GR-GFNs*	**0.002**	**1.38e-08**	**3.784**	**0.95**	**1**
NN-int-GFNs*	0.031	**1.38e-08**	**3.784**	0.784	**1**
NN-GFNs*	0.018	**1.38e-08**	**3.784**	0.832	**1**
currin-shubert					
OP-GFNs	0.036	**1.29e-08**	**3.466**	0.727	**0.941**
PC-GFNs	0.038	3.60×10^{-2}	3.000	0.127	0.529
GR-GFNs*	**0.010**	**1.29e-08**	**3.466**	**0.889**	**0.941**
NN-int-GFNs*	0.049	**1.29e-08**	**3.466**	0.63	**0.941**
NN-GFNs*	0.055	**1.29e-08**	**3.466**	0.601	**0.941**
currin-beale					
OP-GFNs	0.002	**1.57e-08**	**2.079**	**0.977**	**1**
PC-GFNs	0.078	**1.57e-08**	**2.079**	0.203	**1**
GR-GFNs*	**0.001**	**1.57e-08**	**2.079**	0.974	**1**
NN-int-GFNs*	0.023	**1.57e-08**	**2.079**	0.434	**1**
NN-GFNs*	0.259	**1.57e-08**	**2.079**	0.379	**1**
shubert-beale					
OP-GFNs	0.038	**7.10e-09**	**3.091**	0.757	**0.733**
PC-GFNs	0.033	1.70×10^{-2}	3.015	0.362	0.667
GR-GFNs*	**0.008**	**7.10e-09**	**3.091**	**0.889**	**0.733**
NN-int-GFNs*	0.067	**7.10e-09**	**3.091**	0.523	**0.733**
NN-GFNs*	0.068	**7.10e-09**	**3.091**	0.521	**0.733**

Table 3: HyperGrid results for $d = 3$

Method	$\mathrm{d}_H(P', P)$ (↓)	IGD+ (↓)	PC-ent (↑)	% front (↑)	Coverage (↑)
branin-currin-shubert					
OP-GFNs	0.025	9.07×10^{-7}	4.630	0.786	**0.991**
PC-GFNs	0.055	0.0023	4.510	0.331	0.860
GR-GFNs*	0.014	9.07×10^{-7}	4.630	0.907	0.963
GR-GFNs (25)*	**0.013**	9.07×10^{-7}	4.630	**0.929**	**0.991**
NN-int-GFNs*	0.056	**1.61e-08**	**4.640**	0.64	**0.991**
NN-GFNs*	0.025	9.071×10^{-7}	4.630	0.827	0.963
branin-currin-beale					
OP-GFNs	0.017	2.00×10^{-4}	5.411	0.746	0.978
PC-GFNs	0.017	8.00×10^{-4}	5.305	0.723	0.882
GR-GFNs*	**0.003**	**1.64e-05**	**5.425**	**0.969**	**0.991**
GR-GFNs (25)*	**0.003**	6.46×10^{-5}	**5.425**	0.963	**0.991**
NN-int-GFNs*	0.011	1.00×10^{-4}	**5.425**	0.860	**0.991**
NN-GFNs*	0.011	1.00×10^{-4}	5.421	0.934	0.987
branin-shubert-beale					
OP-GFNs	0.024	2.00×10^{-4}	4.984	0.685	**0.987**
PC-GFNs	0.041	2.80×10^{-2}	4.650	0.355	0.686
GR-GFNs*	0.010	**1.45e-08**	**4.990**	0.896	0.967
GR-GFNs (25)*	**0.004**	**1.45e-08**	**4.990**	**0.928**	0.967
NN-int-GFNs*	0.028	**1.45e-08**	**4.990**	0.704	0.967
NN-GFNs*	0.024	1.00×10^{-4}	4.984	0.817	0.961
currin-shubert-beale					
OP-GFNs	0.023	**1.41e-08**	**4.844**	0.690	**1**
PC-GFNs	0.037	4.00×10^{-2}	4.025	0.340	0.411
GR-GFNs*	**0.009**	**1.41e-08**	**4.844**	0.902	0.992
GR-GFNs (25)*	**0.009**	**1.41e-08**	**4.844**	**0.908**	0.992
NN-int-GFNs*	0.035	**1.41e-08**	**4.844**	0.673	**1**
NN-GFNs*	0.020	**1.41e-08**	**4.844**	0.830	0.992

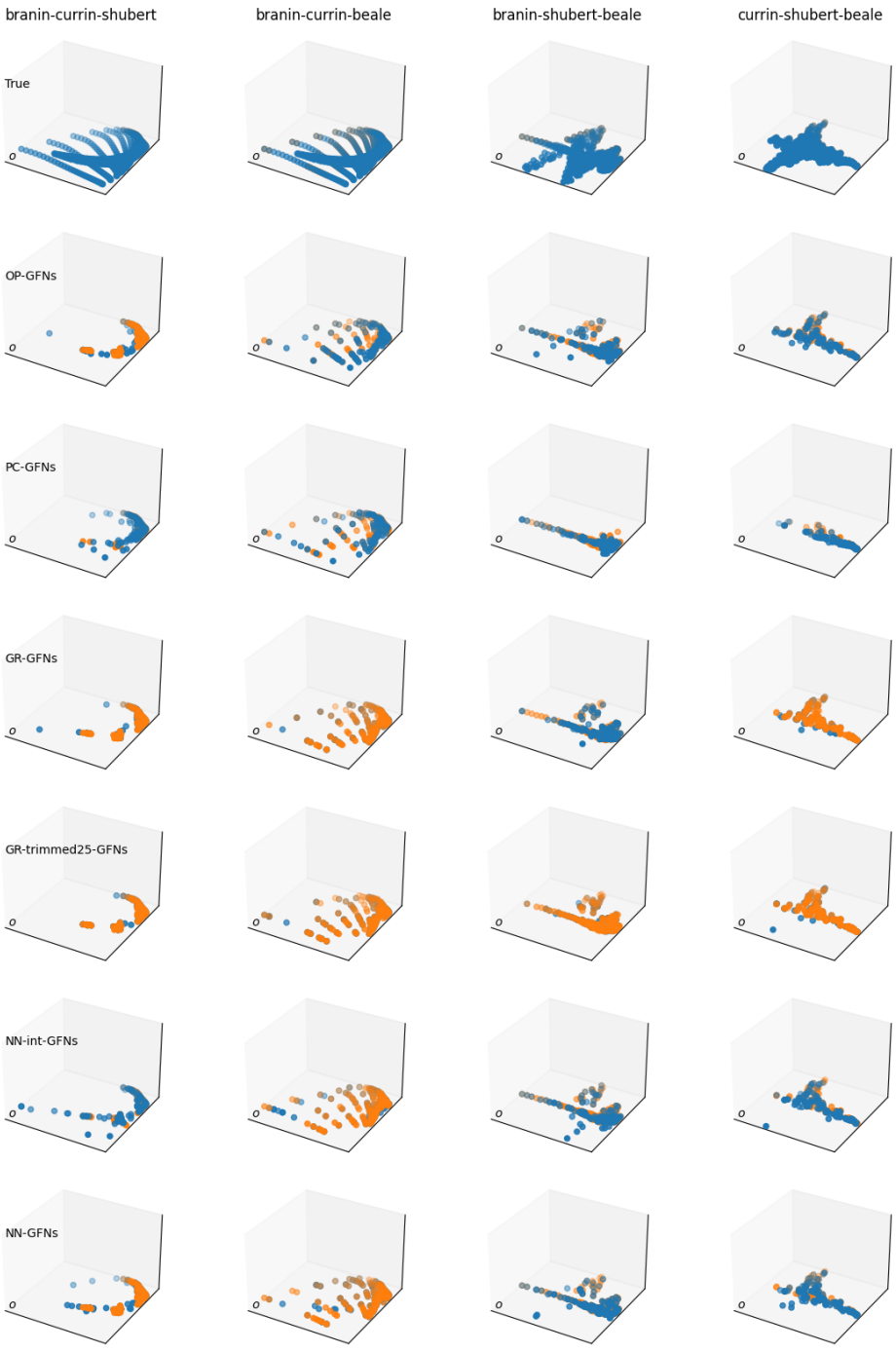

Figure 8: Results for $d = 3$

C.2 N-GRAMS

This next synthetic benchmark is called N-Grams, and it was proposed by Stanton et al. (2022). The goal here is to generate strings of a maximum length L that are rewarded better if they have more occurrences of some given substrings, in this case individual letters (unigrams) or pairs (bigrams). We consider the objectives shown in Table 4. For the unigrams case, we set $L = 18$ and for the bigrams, $L = 36$. In this benchmark, we saw that NN-GFNs were already underperforming compared to the

Table 4: Objectives considered for the n-grams task

# Objectives	Unigrams	Bigrams
2	A, C	AC, CV
3	A, C, V	AC, CV, VA
4	A, C, V, W	AC, CV, VA, AW

other three main algorithms, and therefore, we leave it out in this study. Hence we compare OP-GFNs, PC-GFNs and the Cheap-GR-GFNs, with replay buffer of capacity 10000 and warm-up 1000.

C.2.1 NETWORK STRUCTURE AND TRAINING DETAILS

As in the previous benchmark, the backward transition probability P_B is configured to be uniform, whereas the forward transition probability P_F is modeled using a Transformer-based encoder. This encoder is implemented featuring three hidden layers each with a dimension of 64 and utilizing eight attention heads for the embedding of the current state s. It is characterized by being unidirectional with no dropout. For the PC-GFNs, as indicated in Jain et al. (2023); Chen and Mauch (2024), the preference is encoded using Dir(1), 50 bins, and the exponent β for the reward being 96. We use Adam optimizer. In this case, with learning rates 10^{-4} and 10^{-3} for P_F and Z respectively.

C.2.2 RESULTS

After training the network, we generate 1280 candidates and the results are summarized in Table 5, where the k in the Top k-diversity is 10. We observe that in this particular case the two algorithms that perform better are PC-GFNs and Cheap-GR-GFNs. Except for the case of 2 unigrams (PC-GFNs and Cheap-GR-GFNs perform similarly) and the 2 and 3 bigrams (PC-GFNs perform better), we observe our method to outperform the others. We remark that our methods, even having similar results for two and three objectives, stand out with 4 objectives.

Table 5: N-Grams Results

REGEX	HV (↑)	R_2 (↓)	PC-ent (↑)	$d_H(P', P)$ (↓)	Diversity (↑)
2 Unigrams					
OP-GFNs	**0.47**	1.46	2.25	0.34	7.17
PC-GFNs	**0.47**	**1.45**	**2.26**	**0.33**	3.40
Cheap-GR-GFNs*	**0.47**	**1.45**	**2.26**	**0.33**	**8.22**
2 Bigrams					
OP-GFNs	0.53	1.66	1.90	0.34	5.14
PC-GFNs	0.52	**1.42**	**2.15**	**0.30**	**14.99**
Cheap-GR-GFNs*	**0.57**	1.50	2.08	0.31	5.76
3 Unigrams					
OP-GFNs	0.13	11.07	3.69	0.59	9.74
PC-GFNs	0.04	10.30	2.99	0.60	5.22
Cheap-GR-GFNs*	**0.14**	**9.53**	**3.88**	**0.57**	**9.87**
3 Bigrams					
OP-GFNs	0.30	10.32	1.10	0.56	1.08
PC-GFNs	0.32	**8.38**	**2.25**	**0.42**	**13.98**
Cheap-GR-GFNs*	**0.33**	9.21	2.20	0.49	10.15
4 Unigrams					
OP-GFNs	**0.03**	53.79	4.78	0.79	11.30
PC-GFNs	0.01	49.77	3.36	0.79	4.49
Cheap-GR-GFNs*	**0.03**	**45.54**	**5.07**	**0.76**	**11.44**
4 Bigrams					
OP-GFNs	0.06	50.18	3.89	0.68	**16.88**
PC-GFNs	0.05	48.09	3.90	0.67	15.13
Cheap-GR-GFNs*	**0.09**	**39.94**	**4.52**	**0.60**	12.79

C.3 DNA Sequence Generation

C.3.1 Objective Functions

With a fixed length of 30 elements, we can compute several rewards:

- `energy`: Free energy of the secondary structure computed with NUPACK Zadeh et al. (2011)
- `pairs`: Number of base pairs
- `pins`: DNA hairpin index

Due to a time limitation we could only evaluate `energy-pins` and `energy-pins-pairs`, and it is left for the future to evaluate the other combinations of objective functions.

C.3.2 Network Structure and Training Details

We are going to compare ourselves with the same algorithms as before and also the same parameters as the N-Grams task, due to the similar nature of the problem. Following the advice of Jain et al. (2023); Chen and Mauch (2024), we now set $\beta = 80$ for PC-GFNs. Network architectures and training setup are identical to the previous experiment.

C.4 Fragment-Based Molecule Generation

C.4.1 Objective Functions

We use the same rewards adopted in the previous works Jain et al. (2023); Chen and Mauch (2024); Roy et al. (2023):

- `SEH`: A pretrained model acts as a proxy to predict the binding energy of a molecule to the soluble epoxide hydrolase, closely related to the Alzheimer's treatment Griñán-Ferré et al. (2020).
- `QED`: A measure for a drug's likeness Bickerton et al. (2012).
- `SA`: Synthetic Accessibility Ertl and Schuffenhauer (2009). SA is extracted from RDKit RDKit: Open-source cheminformatics, and the final reward is $\mathrm{R}_{SA} = (10 - \texttt{SA})/9$.
- `MW`: Molecular Weight, in this case a region that favors weights under 300: $\mathrm{R}_{MW} = ((300 - \texttt{MW})/700 + 1).\mathrm{clip}(0, 1)$

C.4.2 Network Structure and Training Details

As usual, P_B is not parameterized but set as uniform. The novelty lies in P_F, which is now modeled by a Graph Neural Network (GNN) based on the graph transformer architecture Yun et al. (2019). It has two layers with node embedding size of 64. As indicated in Chen and Mauch (2024), the preference vector w is represented using thermometer encoding with 16 bins, while the temperature β is similarly encoded but with 32 bins. The training details are very similar to previous sections, except for the fact that GR-GFNs are trained in 10000 steps instead of 20000. Due to limited computational resources, we set 30 the maximum rank in GR-GFNs.

C.4.3 Ablation study

In training with a replay buffer, there is a question of how to balance samples on the Pareto front and general samples. To address this, we conduct an experiment utilizing the `SEH-QED` objective functions, examining sample ratios of 0.1, 0.2, and 0.4. The outcomes of this investigation are shown in Figure 9. Our analysis indicates that setting the Pareto ratio to 0.1 results in generated samples that are highly aligned with the desired optimization objectives, closely resembling the ideal Pareto front. Conversely, setting the ratio to 0.4 yields less favorable outcomes. Specifically, this higher ratio appears to constrict the network's exploratory capabilities, leading to premature convergence on previously recognized Pareto front points during the initial stages of training (which, of course, are worse than the ideal Pareto front). This observation emphasizes the importance of carefully selecting the sample ratio to balance effective exploration with optimal convergence.

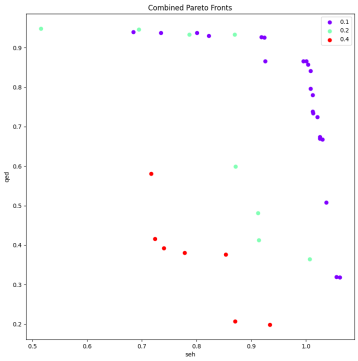

Figure 9: Ablation study of the Pareto ratio sizes

C.4.4 Plots and Tables

Table 6: Results for Fragmentation-Based Molecule Generation benchmark

SEH-QED	PC-ent (↑)	IGD+ (↓)	$\mathrm{d}_H(P', P)$ (↓)	R_2 (↓)	Non-dominated (↑)
OP-GFNs	**2.12**	0.22	**0.1819**	7.7	0
PC-GFNs	1.88	0.43	0.36	13.277	0
GC-GFNs	1.59	0.44	0.38	13.35	0
GR-GFNs (30)*	1.37	0.21	0.33	**3.82**	**9**
Cheap GR-GFNs*	1.76	**0.19**	0.23	4.39	1
SEH-SA					
OP-GFNs	**1.73**	**0.25**	0.12	**3.27**	**9**
PC-GFNs	1.54	0.29	0.24	10.89	0
GC-GFNs	1.58	0.32	0.28	10.84	0
GR-GFNs (30)*	1.43	0.28	0.15	4.55	2
Cheap GR-GFNs*	1.55	0.34	**0.01**	3.69	6
SEH-MW					
OP-GFNs	1.95	**0.21**	0.18	6.83	0
PC-GFNs	1.85	0.32	0.36	11.54	0
GC-GFNs	**2.02**	0.29	0.35	11.27	0
GR-GFNs (30)*	0.64	0.37	0.35	1.66	2
Cheap GR-GFNs*	1.55	0.28	**0.13**	**1.35**	**9**
QED-SA					
OP-GFNs	0.85	0.39	0.44	**1.93**	**15**
PC-GFNs	**1.73**	**0.06**	0.35	11.64	0
GC-GFNs	1.47	0.07	0.33	11.8	0
GR-GFNs (30)*	1.33	0.36	0.24	3.32	5
Cheap GR-GFNs*	1.04	0.37	**0**	7.8	3
QED-MW					
OP-GFNs	0	0.5	0.83	**1.27**	**2**
PC-GFNs	0.69	0.44	0.38	12.1	0
GC-GFNs	**1.32**	**0.43**	**0.36**	12.9	0
GR-GFNs (30)*	0	0.51	0.81	1.72	1
Cheap GR-GFNs*	0	0.51	0.43	2.39	0
SA-MW					
OP-GFNs	0	0.53	**0**	**0**	**1**
PC-GFNs	0.87	0.38	0.32	10.86	0
GC-GFNs	**1.33**	**0.32**	0.27	10.19	0
GR-GFNs (30)*	0	0.53	**0**	**0**	**1**
Cheap GR-GFNs*	0	0.53	0.02	**0**	**1**

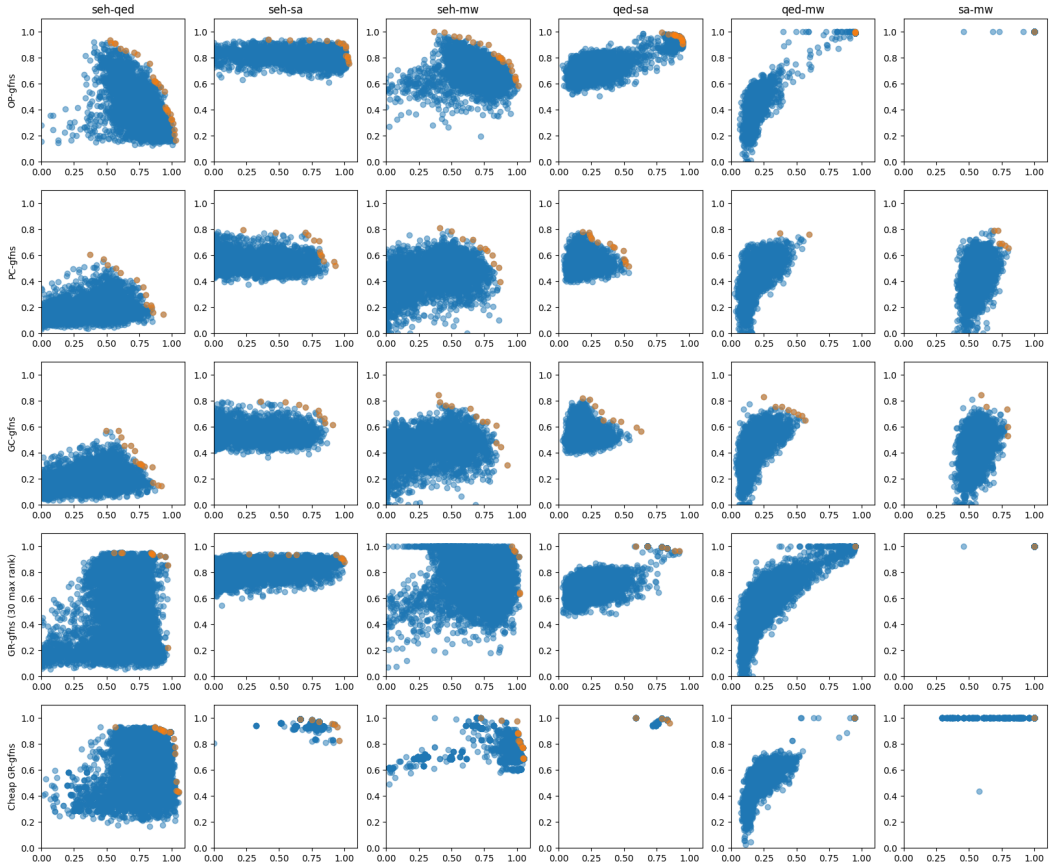

Figure 10: Results for the fragmentation-based molecule generation: generated samples (blue) and the Pareto fronts (orange)

C.5 QM9

C.5.1 Objective Functions

The environment QM9 Ramakrishnan et al. (2014) provides different metrics when evaluating the sequentally generation of molecules of up to 9 atoms and different bonds. Following previous works Jain et al. (2023); Chen and Mauch (2024) we consider the following objective functions:

- `MXMNet`: This is the main reward, a proxy Zhang et al. (2020) trained to predict the HOMO-LUMO gap.
- `logP`: The molecular logP target which can be extracted from RDKit: $\mathrm{R}_{logP} = \exp\left(-(\texttt{logP} - 2.5)^2/2\right)$
- `SA`: Same as 4.2
- `MW`: In this case we modify this function following what has been done in Jain et al. (2023): $\mathrm{R}_{MW} = \exp\left(-(\texttt{MW} - 105)^2/150\right)$

C.5.2 Network Structure and Training Details

We examined them under varying training durations. Specifically, we subjected both PC-GFNs and GR-GFNs to 100000 steps. In contrast, OP-GFNs underwent a shorter training period with 50000 steps, while Cheap-GR-GFNs had the least, with 30000 steps. As in the previous experiment the chosen model is the GNN, with 4 layers and number of embeddings being 128.

C.5.3 Plots and Tables

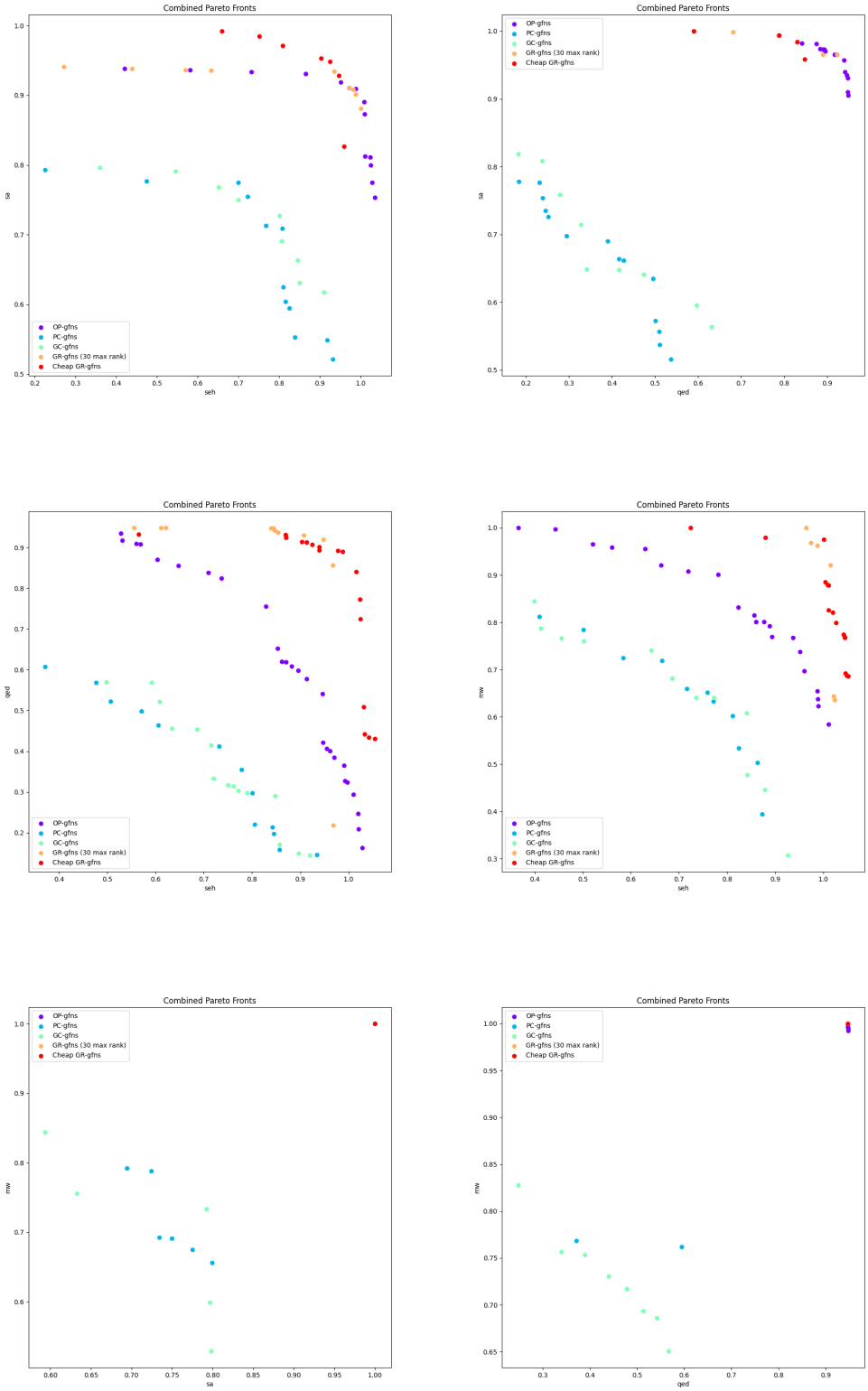

Figure 11: Fragmentation-based molecule generation: Comparison of the different Pareto fronts provided by each method.

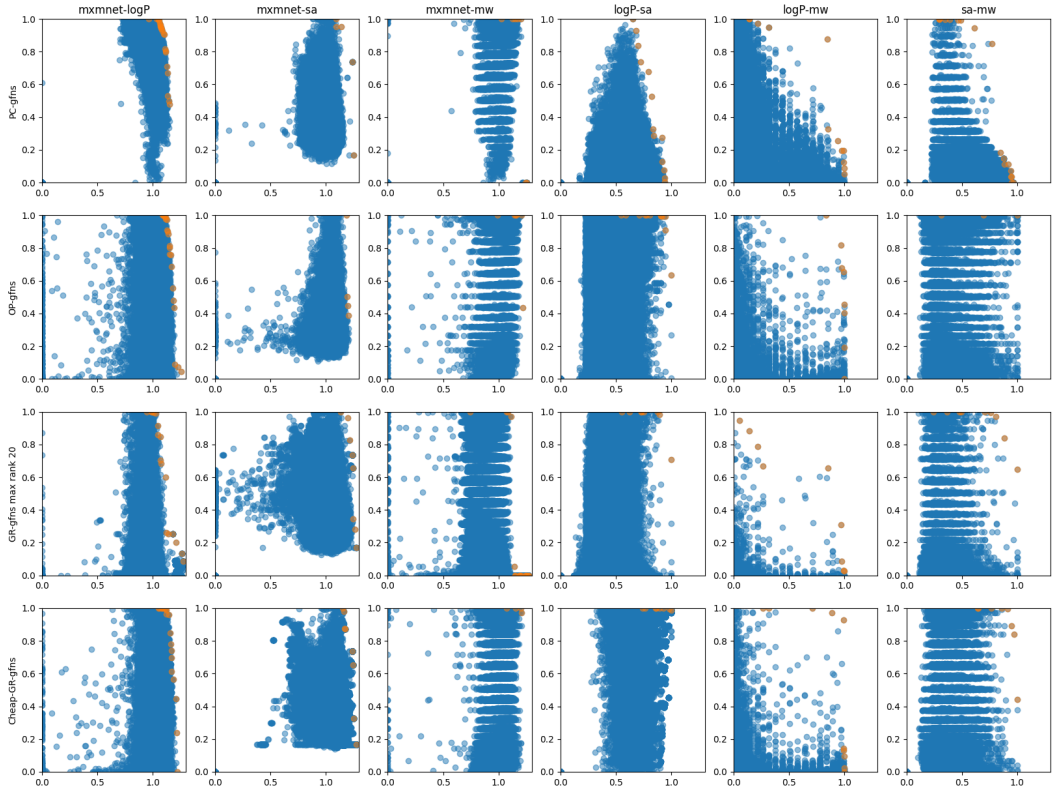

Figure 12: All generated samples (blue) for each method and their respective Pareto front (orange)

Table 7: Results for QM9 benchmark

MXMNet-logP	PC-ent (↑)	IGD (↓)	$\mathrm{d}_H(P', P)$ (↓)	R_2 (↓)	Non-dominated (↑)
PC-GFNs	1.38	0.33	**0.1**	62.47	0
OP-GFNs	1.55	0.31	0.71	44.8	7
GR-GFNs (20)*	1.37	**0.29**	1.01	**8.7**	2
Cheap GR-GFNs*	**1.75**	**0.29**	0.88	26.11	**12**
MXMNet-SA					
PC-GFNs	0.95	0.33	0.33	66.25	0
OP-GFNs	1.39	0.39	0.77	33.32	1
GR-GFNs (20)*	1.56	**0.31**	0.69	37.23	**6**
Cheap GR-GFNs*	**1.67**	0.32	**0.24**	**32.79**	3
MXMNet-MW					
PC-GFNs	**0.96**	0.35	**0.09**	32.2	3
OP-GFNs	0.87	0.32	0.95	**19.64**	3
GR-GFNs (20)*	0.82	**0.27**	0.42	27.37	**5**
Cheap GR-GFNs*	0.56	0.5	1.09	25.01	3
logP-SA					
PC-GFNs	**1.82**	0.23	**0.41**	51.74	0
OP-GFNs	1.63	**0.22**	0.82	8.33	1
GR-GFNs (20)*	1.49	0.25	1.1	8.82	0
Cheap GR-GFNs*	1.04	0.41	0.44	**3.54**	**8**
logP-MW					
PC-GFNs	**1.98**	0.14	0.77	35.18	2
OP-GFNs	1.73	0.22	0.99	1.43	**5**
GR-GFNs (20)*	1.89	0.2	**0.72**	1.66	1
Cheap GR-GFNs*	1.83	**0.13**	0.99	**0.62**	3
SA-MW					
PC-GFNs	1.75	**0.17**	**0.36**	51.64	0
OP-GFNs	1.1	0.32	0.83	**0.01**	1
GR-GFNs (20)*	**1.89**	0.18	1.02	6.6	0
Cheap GR-GFNs*	1.56	0.2	0.98	7.04	**3**

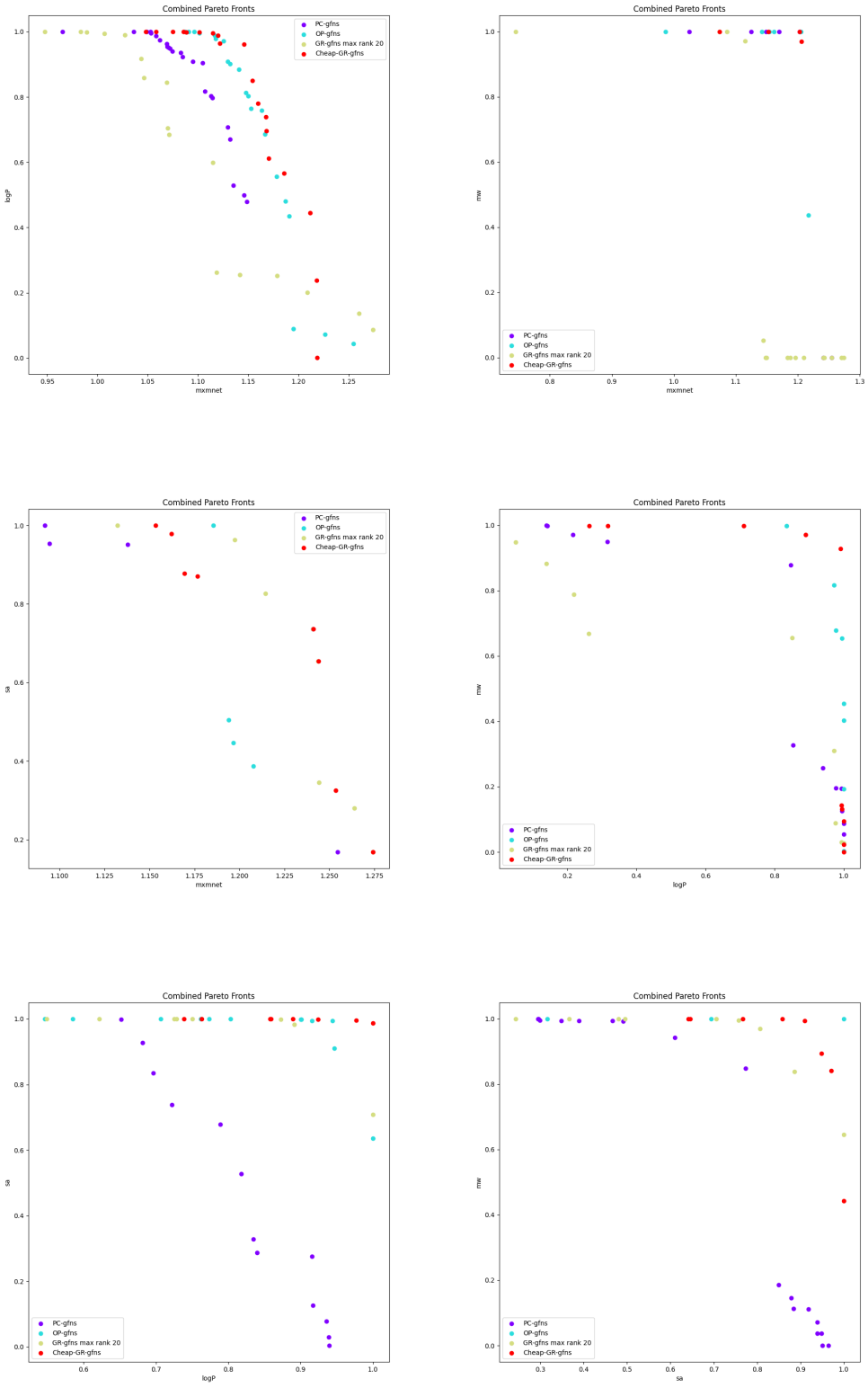

Figure 13: QM9: Comparison of the different Pareto fronts provided by each method.