A Metrics

In this section we discuss the various metrics that we used to report the results in Section 5.

1. **Generational Distance Plus (GD +)** [18]: This metric measures the euclidean distance between the solutions of the Pareto approximation and the true Pareto front by taking the dominance relation into account. To calculate **GD+** we require the knowledge of the true Pareto front and hence we report this metric for Hypergrid experiments (Section 5.1.1)
2. **Hypervolume (HV) Indicator** [14]: This is a standard metric reported in MOO works that measures the volume in the objective space with respect to a reference point spanned by a set of non-dominated solutions in Pareto front approximation.
3. $\boldsymbol{R_2}$ **Indicator** [16]: R_2 provides a monotonic metric comparing two Pareto front approximations using a set of uniform reference vectors and a utopian point z^* representing the ideal solution of the MOO. Generally, R_2 metric calculations are performed with z^* equal to the origin and all objectives transformed to a minimization setting thereby ensuring monotonicity of the metric. Here we are instead in a (reward) maximization setting and so the utopian point considered is the vector of maximal values for each reward (typically 1 for normalized rewards).
4. **Top-K Reward**: This metric was originally used in [4], which we extend for our multi-objective setting. For MOGFN-PC, we sample N candidates per test preference and then pick the top-k candidates ($k < N$) with highest scalarized rewards and calculate the mean. We repeat this for all test preferences enumerated from the simplex and report the average top-k reward score.
5. **Top-K Diversity**: This metric was also originally used in [4], which we extend for our multi-objective setting. We use this metric to quantify the notion of diversity of the generated candidates. Given a distance metric $d(x, y)$ between candidates x and y we calculate the diversity of candidates as those who have $d(x, y)$ greater than a threshold $epsilon$. For MOGFN-PC, we sample N candidates per test preference and then pick the top-k candidates based on the diversity scores and take the mean. We repeat this for all test preferences sampled from simplex and report the average top-k diversity score. We use the edit distance for sequences, and 1 minus the Tanimoto similarity for molecules.

B Task Details

B.1 Hypergrid

Here we elaborate on the HyperGrid experimental setup which we discussed in Section 5.1.1. We also present additional results where we vary the grid size and show some quantitative metrics. For all our experiments we consider the following rewards: $\boldsymbol{R(x)} = $ `[brannin(x), currin(x), sphere(x), shubert(x), beale(x)]`. In Figure 5, we show the heatmap for each reward function. Note that we normalize all the reward functions between 0 and 1.
Additional Results: Here we present extended qualitative visualizations across more preferences in

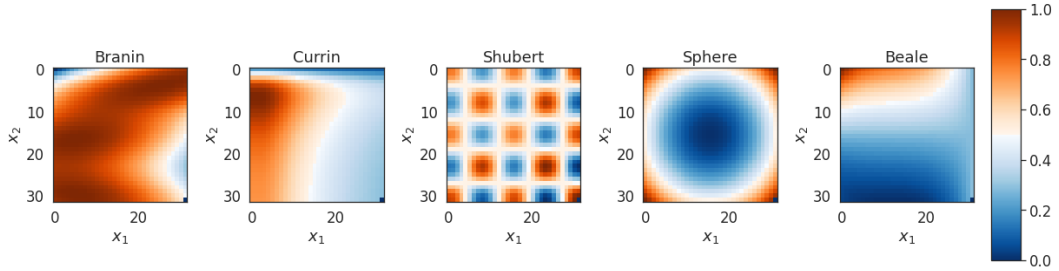

Figure 5: **Reward Functions** Different reward function considered for HyperGrid experiments presented in Section 5.1.1. Here the grid dimension is $H = 32$

.

Figure 6.

Table 6: Hyperparameters use for String Generation Task

Hyperparameter	Values
Learning Rate (P_F)	{0.01, 0.05, 0.001, 0.005, 0.0001}
Learning Rate (Z)	{0.01, 0.05, 0.001}
Reward Exponent: β	{16, 32, 48}
Uniform Policy Mix: δ	{0.01, 0.05, 0.1}

Model Details and Hyperparameters For MOGFN-PC policies we use an MLP with two hidden layers each consisting of 64 units. We use `LeakyReLU` as our activation function as in [4]. All models are trained with `learning rate=0.01` with the Adam optimizer [23] and `batch size=128`. We sample preferences ω from Dirichlet(α) where $\alpha = 1.5$.

B.2 String

Model Details and Hyperparameters We consider a Transformer, with 3 hidden layers of dimension 64 with 8 attention heads, as the architecture all methods. We present the hyperparameters we used in Table 6. Each method is trained for 10,000 iterations with a minibatch size of 128.

B.3 QM9

More Details As mentioned in Section 5.2.1, we consider four reward functions for our experiments. Here we elaborate on them. The first reward function is the HUMO-LUMO gap, for which we rely on the predictions of a pretrained model trained on the QM9 dataset [33]. The second reward is the standard Synthetic Accessibility score which we calculate using the RDKit library, to get a reward we take $(10 - \texttt{SA})/9$. The third reward function is molecular weight target. Here we first calculate the molecular weight of a molecule using RDKit, and then construct a reward function of the form $e^{-(\texttt{molWt}-105)^2/150}$ which is maximized at 105. Our final reward function is a logP target, $e^{-(\texttt{logP}-2.5)^2/2}$, which is again calculated with RDKit and is maximized at 2.5.

Model Details and Hyperparameters We use a graph neural network based on a graph transformer architecture [43].

Hyperparameter	Values
Learning Rate (P_F)	0.0005
Learning Rate (Z)	0.0005
Reward Exponent: β	32
Batch Size:	64
Number of Embeddings	64
Uniform Policy Mix: δ	0.001
Number of layers	4

B.4 Fragments

More Details As mentioned in Section 5.2.2, we consider four reward functions for our experiments. The first reward function is a proxy trained on molecules docked with AutodockVina [37] for the sEH

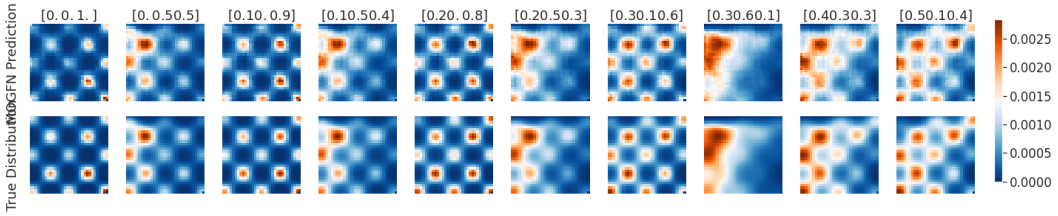

Figure 6: Extended Qualitative Visualizations for Hypergrid epxeriments

target; we use the weights provided by [4]. We also use synthetic accessibility, as for QM9, and a weight target *region* (instead of the specific target weight used for QM9), `((300 - molwts) / 700 + 1).clip(0, 1)` which favors molecules with a weight of under 300. Our final reward function is QED which is again calculated with RDKit.

Additional Results

Model Details and Hyperparameters We again use a graph neural network based on a graph transformer architecture [43]. We additionally sample from a lagged model whose parameters are updated as $\theta' = \tau\theta' + (1-\tau)\theta$.

Hyperparameter	Values
Learning Rate (P_F)	0.0005
Learning Rate (Z)	0.0005
Reward Exponent: β	96
Batch Size:	256
Sampling model τ	0.95
Number of Embeddings	128
Number of layers	6

B.5 DNA Sequence Design

More details In order to compute the free energy and number of base with the software NUPACK, we used 310 K as the temperature. The inverse of the length L objective was calculated as $\frac{30}{L}$, as 30 was the minimum length for sampled sequences.

Model Details and Hyperparameters As with the synthetic string generation task, we consider a 4-layer Transformer, with 256 units per layer and 16 attention heads, as the architecture for all methods. We detail the most relevant hyperparameters Table 7.

B.6 Active Learning

We follow the exact experimental setup used in [36]. The active learning batch size is set to 16, and we run 64 rounds of optimization. Table 8 presents the hyperparameters used for MOGFN-AL.

Table 7: Hyperparameters tuned for DNA-Aptamers Task. The values used in the final models are highlighted in bold.

Hyperparameter	Values
Learning Rate (P_F)	{0.01, 0.001, 0.0001, **0.00001**, 0.000001}
Learning Rate (Z)	0.001
Reward Exponent: β	{5, 20, 40, **60**, **80**, 100}
Batch Size:	16
Training iterations:	10,000
Dirichlet α	{0.1, **1.0**, 5.0}

Table 8: Hyperparameters tuned for the Active Learning Task

Hyperparameter	**Values**
Learning Rate (P_F)	{0.01, 0.001, 0.0001}
Learning Rate (Z)	{0.01, 0.001}
Reward Exponent: β	{16, 24}
Uniform Policy Mix: δ	{0.01, 0.05}
Number of mutations	{10, 15, 20}