DIRECT PARETO OPTIMIZATION: LEARNING TO RANK FOR MULTI-OBJECTIVE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Many real-world sequential design problems, such as molecule optimization and neural architecture search, require one to balance multiple conflicting objectives simultaneously. Prior approaches typically address this by decomposing the problem into a series of single-objective problems or optimizing auxiliary indicators such as hypervolume. While effective, these methods either do not directly optimize the quality of the solution set or are limited by the effectiveness of the indicators. In this work, we propose *Direct Pareto Optimization (DPaO)*, a framework that directly optimizes with respect to Pareto Dominance by casting the multi-objective sequential optimization task as a learning-to-rank problem. Specifically, we learn a scalar valued latent reward function from the *partial ordering* induced by Pareto dominance, thereby transforming the *whole* multi-objective problem into a single-objective one. We evaluate the proposed method in both synthetic and real-world tasks, where DPaO consistently ranks among top methods, regardless of the shape of the solution set or number of objectives. Furthermore, we empirically validate that the learned reward function aligns with the Pareto dominance relation.

1 INTRODUCTION

Optimizing multiple, conflicting objectives remains a key challenge in sequential design tasks. For example, in drug discovery, one seeks to find molecules that are effective, safe for human use, and also easily synthesizable (Dara et al., 2021). In Neural Architecture Search (NAS) one needs to balance the performance and computational complexity of the optimized model (Zhao et al., 2024). Previous solutions can be broadly divided into a two classes: 1) *Indicator* based methods utilize some auxiliary performance metrics such as hypervolume or IGD+, requiring access to difficult to estimate environment specific information, which can have a large effect on the ranking of solution sets (Zitzler et al., 2008; Ishibuchi et al., 2018a;b; Pang et al., 2024). 2) *Decomposition* methods reduce the multi-objective problem into series of single-objective problems, which can be solved using existing single-objective approaches. However these methods require one to solve multiple sub-problems without directly optimizing the solution set quality (Roijers et al., 2013; Hayes et al., 2022). Notably, neither type of method can leverage existing single-objective optimization methods, while directly optimizing the Pareto optimality relation. Therefore, in this work, we study the following question: *Can we derive a latent reward function, that directly optimizes the Pareto optimality of the solution set?*

We propose Direct Pareto Optimization (DPaO), which reframes the sequential multi-objective optimization as a learning-to-rank task. DPaO models the *partial order* induced by Pareto dominance via mixture of Plackett-Luce models (Luce, 1959; Plackett, 1975), whose parameters define latent reward function over the solutions. Since no single solution typically optimizes all of the objectives the simultaneously, this latent reward is multi-modal, and thus we seek to find a *maximum entropy* solution, capturing all high performance regions. We leverage closed form solution of maximum entropy Reinforcement Learning (RL) problem (Ziebart et al., 2008; Haarnoja et al., 2017) to directly optimize the policy to generate solutions according to the partial order induced by the Pareto-dominance relation. Figure 1 showcases overview of the proposed method.

DPaO combines the best of indicator based and decomposition based approaches, while avoiding their drawbacks: By deriving a *latent reward* directly from the Pareto dominance relation, we avoid

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

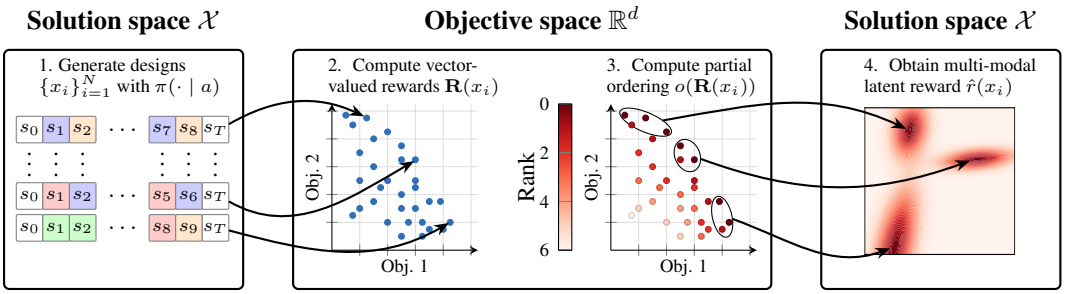


Figure 1: Overview of DPaO. *Firstly*, DPaO generates a batch of trajectories with the current policy. *Secondly*, it computes the partial ordering of the trajectories based on their vector rewards. *Lastly*, it optimizes the log-likelihood of the mixture of PL models to obtain a multi-modal latent reward function $\hat{r}$ over the solution space $\mathcal{X}$.

the need for auxiliary indicators, or environment specific information, while optimizing quality of the solution set. Unlike the decomposition based methods, DPaO is invariant to changing reward scales, and can handle irregularly shaped solution sets effectively, a common challenge, especially when using linear scalarization (Roijers et al., 2013; Hayes et al., 2022; Lu et al., 2023). Furthermore, DPaO removes many decisions, such as the type of scalarization function and weight sampling distribution, that practitioners must make *a priori*, when using decomposition based methods. In contrast, DPaO has only one hyperparameter, the entropy coefficient β, which we fix in all our experiments, demonstrating the method's robustness.

2 Related work

Indicator based methods maintain or generate a set of solutions, which is improved based on a scalar-valued indicator function. Evolutionary algorithms (EAs) such as IBEA (Zitzler & Künzli, 2004) and SMS-EMOA (Beume et al., 2007) apply selection operations that retain candidates based on the improvement in a given metric. Multi-Objective Bayesian Optimization (MOBO) methods update the standard Bayesian Optimization loop with multi-objective acquisition function, such as Expected Hypervolume Improvement (EHVI) (Emmerich, 2005; Emmerich et al., 2011; Daulton et al., 2020) or its noisy variant (Daulton et al., 2021). The acquisition function is then maximized, commonly via gradient based optimization, to choose which candidates are evaluated next. Ultimately, these methods are dependent on the quality of the used indicator, often opting to use the Pareto compliant hypervolume. Unfortunately, computing gradients of hypervolume is difficult (Daulton et al., 2020), and it is known to be sensitive to an environment-specific reference point, which is challenging to approximate in practice (Zitzler et al., 2008; Vamplew et al., 2011; Ishibuchi et al., 2018b;a; Pang et al., 2024).

Decomposition based methods split the problem into multiple single-objective optimization problems by scalarizing the vector-valued reward function (Qingfu Zhang & Hui Li, 2007). The primary advantage of these methods is the ability to use existing single-objective methods (Hayes et al., 2022). For example, Lin et al. (2022) apply REINFORCE (Williams, 1992) to multi-objective combinatorial optimization, Jain et al. (2023) extend GFlowNets (GFNs) (Bengio et al., 2021) to the multi-objective domain via a linear scalarization function, and Roy et al. (2023) propose a goal-conditioned GFN, which assigns a reward only if the generated solution satisfies a given goal. However, decomposition methods require one to solve multiple sub-problems, while not explicitly optimizing the solution set quality (Roijers et al., 2013; Hayes et al., 2022). Moreover, they are sensitive to varying reward scales and commonly rely on linear scalarization, which struggles to recover solutions in non-convex Pareto fronts (Roijers et al., 2013; Hayes et al., 2022; Lu et al., 2023).

Similar to indicator based methods, our method, DPaO, optimizes a scalar objective over the solution set. However, unlike them, the objective is directly derived from the dominance relation, requiring neither environment-specific information nor auxiliary indicators. Compared to decom-

position based methods, DPaO is robust against changing reward scales and can handle varying solution set shapes. In addition, DPaO inherits the benefits of policy-based methods: it learns a generative policy that amortizes the search cost at training time, enabling fast generation of candidates at evaluation. Moreover, it operates directly on discrete state and action spaces, side-stepping the need for learned continuous embeddings, common with MOBO methods (Stanton et al., 2022). Two prior methods are closest to ours. POCCO (Fan et al., 2025) shares the closed-form maximum entropy solution with DPaO, yet uses it to solve series of single-objective problems via scalarization. Order-preserving GFlowNet (OP-GFN) (Chen & Mauch, 2023), trains a policy based on the Pareto optimization criteria, similarly to DPaO. However, OP-GFN discards information among dominated candidates, while DPaO exploits the full ranking induced by the dominance relation.

3 BACKGROUND

Multi-Objective Sequential Optimization (MOO): We study sequential design tasks, where a policy must balance multiple conflicting objectives, while constructing a solution step-by-step. Formally, we model this process as a special case of Multi-Objective Markov Decision Process (MOMDP) (Chatterjee et al., 2006), called *multi-objective compositional discrete process* (MOCDP) (Jiralerspong et al., 2026). Specifically, MOCDP assumes deterministic transitions, and can be expressed as a tuple $\langle \mathcal{G}, \boldsymbol{R} \rangle$, where $\mathcal{G}$ is a directed acyclic graph with a single source state s_o and set of terminal states $\mathcal{X}$. Each node in the graph corresponds to a partial object, with s_0 being an empty set. An action represents an edge $s \to s'$ that adds a part to the constructed object, or terminates if $s' \in \mathcal{X}$. Since we focus specifically on sequential design, the $\mathcal{G}$ becomes a tree (i.e. there is exactly one path from s_0 to each object $x \in \mathcal{X}$). The vector valued reward $\boldsymbol{R} : \mathcal{X} \to \mathbb{R}^d$ returns the $d \geq 2$ *conflicting* reward components for the complete object. Thus, it is not possible to find a single solution that optimizes all the objectives at the same time. Instead we try to find a set of optimal solutions:

Definition 1 (Pareto dominance)**.** *We say that a solution $x \in \mathcal{X}$ Pareto dominates another solution $x' \in \mathcal{X}$, denoted by $\boldsymbol{R}(x) \succ \boldsymbol{R}(x')$, if $\forall i : \boldsymbol{R}_i(x) \geq \boldsymbol{R}_i(x')$ with at least one strict inequality.*

Definition 2 (Pareto Optimality)**.** *A solution $x^* \in \mathcal{X}$ is Pareto optimal if there is no other solution $x \in \mathcal{X}$ that Pareto dominates it. The set of all such solutions is the Pareto optimal set (PS) and their corresponding set of reward vectors $\{\boldsymbol{R}(x^*) \mid x^* \in PS\}$ forms the Pareto front (PF).*

In practice, we seek to approximate the Pareto front, since obtaining the exact solution is often intractable.

Preference-Based Reinforcement Learning (PbRL) considers a reinforcement learning problem without access to a reward function, and instead the policy must learn from preferences over trajectories (Christiano et al., 2017). Typically PbRL methods model the preferences using the Plackett-Luce (PL) model (Luce, 1959; Plackett, 1975), or in the case of pairwise data, the Bradley-Terry (BT) model (Bradley & Terry, 1952). Then they learn a latent reward function, consistent with the chosen preference model. While initial approaches used a separate reward model (Christiano et al., 2017), we utilize more recent direction that derives a relation between a latent reward and the optimal policy (Rafailov et al., 2023; Pan et al., 2025). This allows one to directly optimize the log-likelihood of the desired target, in our case, the mixture of PL models over partial ranking, without explicitly constructing a reward model.

4 METHODOLOGY

In this section, we introduce our approach that builds a bridge between multi-objective optimization and learning-to-rank approaches. Firstly, we introduce Pareto ranking and showcase how it creates a partitioned preference over the solutions. We then model this ordering with mixture of PL models and derive end-to-end policy optimization objective. Finally, we showcase how this objective can be computed efficiently.

We begin by defining *Pareto ranking* (Deb et al., 2002; Zhao et al., 2022; Shrestha et al., 2026)

$$o(x) = \sum_{x' \in \mathcal{X}} \mathbb{1}[\boldsymbol{R}(x') \succ \boldsymbol{R}(x), x \neq x'] \tag{1}$$

where $\mathcal{X}$ is set of all solutions and $\mathbb{1}$ is an indicator function. Given a batch of solutions $B = \{x_i, \boldsymbol{R}(x_i)\}_{i=1}^N$, this ranking creates a *partitioned preference*, such that 1) $S_1 \succ S_2 \succ \cdots \succ S_M$, where $S_i \succ S_j$ indicates that *any* solution $x \in S_i$ is preferred to any solution in S_j, 2) and that preferences over items in the same partitions is unknown (Lebanon & Mao, 2008; Lu & Boutilier, 2014; Ma et al., 2021). Notably, by definition, this ranking is *Pareto compliant* (exact definition in Appendix A), meaning it never contradicts the dominance relation.

Utilizing the partitioned preference obtained from the Pareto-ranking, *We frame the MOO problem as a learning-to-rank task*. We model the probability of observing a given partial ordering over the set using a mixture of PL models (Luce, 1959; Plackett, 1975), as a single PL model is not capable of modeling partial rankings (Ma et al., 2021):

$$p(S_1 \succ S_2 \succ \cdots \succ S_M) = \sum_{\substack{(x_1,\ldots,x_N)\in \\ \Omega(S_1\succ S_2\succ\cdots\succ S_M)}} \prod_{i=1}^{N} \frac{\exp(\hat{r}(x_i))}{\sum_{k=i}^{N}\exp(\hat{r}(x_k))} \tag{2}$$

where $\Omega(\cdot)$ is a function that maps the partial ranking to a set of all possible permutations of N items, consistent with the partial ranking and $\hat{r} : \mathcal{X} \to \mathbb{R}$ is a latent reward function that assigns the importance for each solution.

The key challenge is that the latent reward $\hat{r}$ is unknown, and thus we seek to learn it. Since the Pareto front typically contains many non-dominated solutions, corresponding to different trade-offs between the objectives, the latent reward function is inherently multi-modal. Therefore standard reward maximization formulation would collapse to a single mode. To this end, we wish to find a *maximum entropy* solution, capturing all modes of the latent reward. Formally, our goal is to find policy such that

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\pi_\theta}\left[\hat{r}(x) + \beta\mathcal{H}(\pi_\theta)\right] \tag{3}$$

where $\mathcal{H}(\pi_\theta) = -\sum_{x\in\mathcal{X}} \pi_\theta(x)\log\pi_\theta(x)$ is the entropy of the policy, and β controls the strength of the entropy regularization. To avoid constructing an explicit reward model, we follow previous research (Ziebart et al., 2008; Haarnoja et al., 2017; Pan et al., 2025), and write the analytical closed form solution, and rearrange the terms to obtain the relation between the latent function and the optimal policy (Rafailov et al., 2023; Pan et al., 2025):

$$\begin{aligned} \pi^*(x) &= \frac{1}{Z}\exp(\frac{1}{\beta}\hat{r}(x)) \\ \hat{r}(x) &= \beta\log\pi^*(x) + \beta\log Z \end{aligned} \tag{4}$$

where the partition function $Z = \sum_{x\in\mathcal{X}}\exp(\frac{1}{\beta}\hat{r}(x))$. Similarly to Rafailov et al. (2023) and Pan et al. (2025), we approximate the optimal policy with the current policy π_θ, a choice we validate empirically in Section 5.4. To obtain the probability of a given partial ordering with respect to the policy, we replace the latent reward function in Equation 2 with the RHS of Equation 4:

$$p_\theta(S_1 \succ S_2 \succ \cdots \succ S_M) = \sum_{\substack{(x_1,\ldots,x_N)\in \\ \Omega(S_1\succ S_2\succ\cdots\succ S_M)}} \prod_{i=1}^{N} \frac{\exp(\beta\log\pi_\theta(x_i))}{\sum_{k=i}^{N}\exp(\beta\log\pi_\theta(x_k))} \tag{5}$$

where the partition function Z cancels out. We can now directly optimize the policy to generate solutions according to the partial order by minimizing the negative log-likelihood of Equation 5 with respect to the policy parameters θ. Algorithm 1 showcases the Pseudocode of DPaO.

Algorithm 1: Pseudocode of DPaO

input: Policy π_θ, reward function $\boldsymbol{R}$
while *budget left* **do**
  $B = \{x_i, \boldsymbol{R}(x_i)\}_{i=1}^N, x_i \sim \pi_\theta$ // Generate batch of solutions & obtain rewards
  $\{S_1, S_2, \ldots, S_M\} \leftarrow o(\{\boldsymbol{R}(x_i)\}_{i=1}^N)$ // Compute partitioned preference using Eq. 1
  $\theta \leftarrow \theta - \nabla_\theta \log p_\theta(S_1 \succ S_2 \succ \cdots \succ S_M)$ // Optimize log-likelihood of Eq. 5
end

Unfortunately, a naive computation of the log-likelihood of Equation 5 exhibits a *factorial* computational complexity of $O(N + S_{\max}!)$, where $S_{\max} = \max_{i \in \{1,\dots M-1\}} |S_i|$ is the size of the largest set in the top $M - 1$ partitions, making the computational complexity infeasible even for moderately sized batches (Ma et al., 2021). To circumvent this, we follow Ma et al. (2021) and utilize the random utility formulation of the PL model with Gumbel distribution (Yellott, 1977), which allows one to decompose the gradient computation into summation of one-dimensional integrals, resulting in $O(N + S_{\max}^3)$ computational complexity. We present the details of the efficient PL model in Appendix B, and validate the practical computational complexity empirically in Appendix D.

In summary, DPaO optimizes directly against Pareto dominance relation by 1) constructing a Pareto compliant ranking, without any environment-specific information, 2) modeling the probability over the ranking with a mixture of PL models, and 3) recasting the multi-objective optimization as a single-objective maximum-entropy RL problem.

5 Experiments

In this section, we firstly answer to our main question from Section 1 by evaluating the feasibility of learning a latent reward against decomposition based methods and order-preserving GFN (sections 5.2 and 5.3). Secondly, we study if our approximation of the optimal policy with learned policy holds in practice (Section 5.4). Lastly, we investigate if modeling the partial rank distribution is beneficial over using auxiliary metrics for breaking the ties between equally ranked solutions (Section 5.5). We receive affirmative answers to all our questions, as detailed below.

5.1 Experimental setup

Environments: We consider five tasks, spanning different domains. REGEX (Stanton et al., 2022) is a synthetic task where goal is to maximize the number of predetermined target substrings (either unigrams or bigrams), in the generated sequence. In NASBench201 (NB201) (Dong & Yang, 2019) and DARTS (Liu et al., 2019; Lu et al., 2024) one needs to optimize neural networks accuracy, while minimizing FLOPs and parameter count over $\sim 15\text{k}$ and $\sim 10^{18}$ options respectively. In DNA sequence design (Jain et al., 2023) one generates nucleotide sequences while considering free-energy (computed with NUPACK (Zadeh et al., 2011)), the number of base pairs, the number of hairpins and the inverse of sequence length. Finally, in small molecular design one generates SELFIES strings (Krenn et al., 2020) (up to 128 tokens), while optimizing octanol-water coefficient (logP) and the Quantitative Estimate of Drug-likeness (QED) (Stanton et al., 2022). The detailed descriptions of the benchmarks are shown in Appendix F.

Evaluation: We use common MOO metrics, including **Hypervolume** (HV) (⇑) (Zitzler & Thiele, 1998), **IGD+** (⇓) (Ishibuchi et al., 2015), **R2-metric** (⇓) (Hansen & Jaszkiewicz, 1998), **ϵ-metric** (⇓) (Zitzler et al., 2008) and **PC entropy** (⇑) (Roy et al., 2023). We specify the details of these metrics in Appendix G. When evaluating the methods, we generate 256 samples, except in the NB201, where we generate only 32 samples, as the search-space is relatively small. All experiments are repeated with 10 different seeds. More comprehensive description of the implementation details is given in H.

Baselines: We compare against 6 policy-based baselines. MO-REINFORCE (Lin et al., 2022) is a multi-objective adaption of the traditional REINFORCE (Williams, 1992) algorithm, while we adapt UREX (Nachum et al., 2017), a variant of policy gradients designed to improve exploration, to multi-objective domain via decomposition. POCCO (Fan et al., 2025) is a PbRL based policy-gradient method that, similarly to us, uses the closed form solution for the maximum entropy optimization, yet they use scalarization to obtain total ordering over the solution space. For these methods we consider both linear and Chebyshev (Van Moffaert et al., 2013) scalarization functions. CAPQL (Lu et al., 2023) is an actor-critic maximum entropy MORL method that uses scalarization to generalize Soft-Actor-Critic (Haarnoja et al., 2018) algorithm to multi-objective domain. MOGFN (Jain et al., 2023) builds on top of trajectory balance GFlowNet (Malkin et al., 2022), and extends it to multi-objective domain via decomposition. OP-GFN (Chen & Mauch, 2023) minimizes the divergence from the Pareto optimality criterion, making it related to our approach. A more detailed description of the baselines is in Appendix E.

Notably, we do not include MOBO baselines, since they are targeted at continuous search-spaces (Zhao et al., 2022; Daulton et al., 2022), and rely on pre-trained models latent representation to map discrete tasks to continuous spaces (Stanton et al., 2022), making the comparison unfair. Moreover, applying these methods to tasks where there is no pre-existing datasets, such as DNA sequence generation, is challenging (Jain et al., 2023).

5.2 Aggregated performance profile

We begin by investigating the overall performance of DPaO with respect to the baselines. To this end, we compute bootstrapped *rank-profile* (Agarwal et al., 2021), which estimates the fraction of runs belonging to a given rank. The best performing methods should have high fraction of runs with small ranks, with very small fraction of runs with high ranks. Figure 2 shows the computed profile.

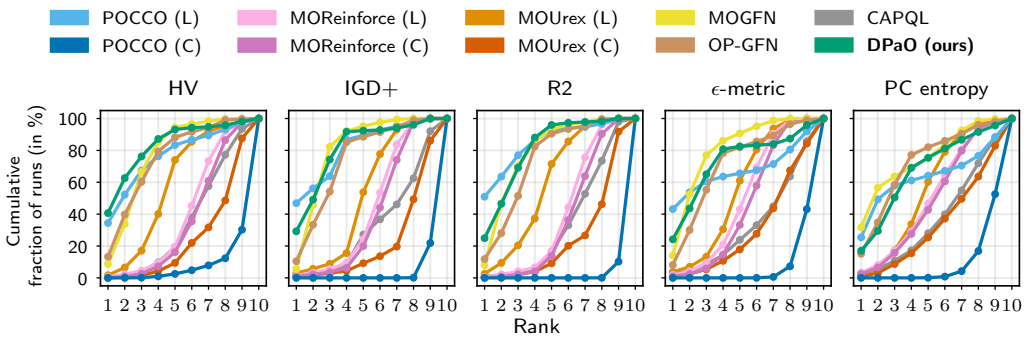


Figure 2: **DPaO consistently ranks among top methods across all tasks**. Figure shows the cumulative fraction of runs as a function of ranks across all tasks for 5 different metrics. Better performing methods are located at the top left of the plot. DPaO outperforms all baselines in HV, while achieving similar ranking to POCCO in IGD+ and R2, ϵ-metric. Notably, it outperforms GFN methods in HV, IGD+ and R2.

From Figure 2, we can make three conclusions: 1) DPaO consistently ranks among the top methods, having most of it's runs placed in the top ranks. Specifically, *it outperforms all other methods in hypervolume, the only Pareto compliant metric*, while achieving similar performance to POCCO in IGD+, R2-metric and ϵ-metric. 2) DPaO performs relatively the worst in PC-entropy, which measures the uniformity of the solution set. This is expected, since we focus on finding high performance solutions that dominate more diverse candidates. 3) DPaO and POCCO, the two best performing methods, utilize the closed form solution to the maximum entropy problem, implying that this policy based approach can be effective, even when compared to GFN based approaches, specifically designed to solve the maximum entropy task.

To further study the robustness of the improvements, we compute the average *probability of improvement* by performing Mann-Whitney U rank test (Mann & Whitney, 1947) across all the tasks, and averaging the per-task probabilities, displayed in Figure 3. Notably in HV and IGD+, DPaO obtains statistically significant improvement over GFN-based baselines, while in rest of the metrics, the probability of improvement is more modest, which is inline with the results from Figure 2.

5.3 Closer look at task level performance

To better understand the properties of DPaO, especially against POCCO, which has similar rank profile to DPaO, we study the per task performance of the methods. To this end, we plot examples of the generated solution sets in Figure 4 (more solution sets shown in Appendix I) and display the final performance in selected environments in Table 1 (full evaluation results presented in Appendix J).

From these, we can infer two main conclusions. Firstly, decomposition based methods, including POCCO, tend to perform well in 2-objective tasks, with performance quickly degrading as the number of objectives increases, while *DPaO retains its performance with increasing number of*

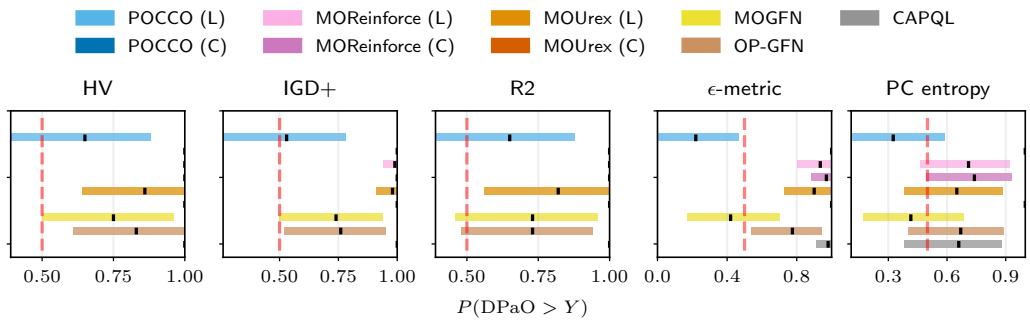


Figure 3: **DPaO achieves statistically significant improvement over GFN based baselines in HV and IGD+.** Figure displays the average probability of improvement over all tasks, computed using Mann-Whitney U rank test. The 95% confidence intervals (CIs) are computed using stratified bootstrapping with $K = 50\,000$ samples. If the CIs are on the right side of the red line ($P = 50\%$), the improvement is statistically significant (Agarwal et al., 2021).

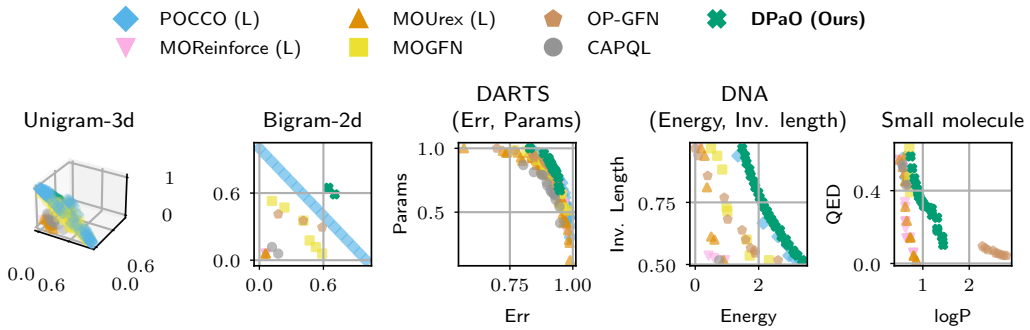


Figure 4: **DPaO generates competitive solutions regardless of the shape of the solution set.** Examples of the generated solution sets in various environments with representative seed (seed = 9). DPaO performs comparably to decomposition based methods in near-convex solution sets, while outperforming them on irregularly shaped environments. To avoid visual clutter, only the best performing variants of MOReinforce, MOURex and POCCO are shown.

objectives. This is a limitation of the decomposition based approach, as one has to solve increasingly many subproblems when the number of objectives is increased. Secondly, *DPaO achieves competitive performance regardless of the solution set shape*, while decomposition based methods

Table 1: **DPaO maintains performance with irregularly shaped solution sets and increasing number of objectives** Table displays the mean ± std over 10 seeds in selected environments, with **best performer** and runner-up highlighted. DPaO has competitive performance in near convex and 2d objective tasks (NB201, DARTS), while retaining the performance in many-objective (Bigram-4d) and irregularly shaped tasks (DNA).

	NB201 (Error, FLOPs)		DARTS (Error, Params)		Bigram-4d		DNA (Energy, Inv. Length)	
Method	HV (⇑)	R2 (⇓)	HV (⇑)	R2 (⇓)	HV (⇑)	R2 (⇓)	HV (⇑)	R2 (⇓)
POCCO (L)	**0.92±0.01**	**1.35±0.03**	**0.96±0.00**	**1.19±0.01**	0.00±0.00	6.44±0.01	1.89±0.37	1.53±0.56
MOReinforce (L)	0.89±0.02	1.49±0.07	0.81±0.02	6.88±0.04	0.00±0.00	8.61±0.07	0.50±0.05	3.50±0.48
MOUrex (L)	0.91±0.01	1.44±0.06	0.92±0.01	1.34±0.04	0.00±0.00	8.60±0.10	1.31±0.38	**1.50±0.41**
CAPQL	0.89±0.01	1.50±0.06	0.81±0.02	1.89±0.12	0.00±0.00	8.90±0.44	0.43±0.05	4.07±0.33
MOGFN	0.91±0.01	1.39±0.04	0.92±0.03	1.34±0.11	0.00±0.00	6.28±0.23	1.37±0.22	2.43±1.41
OP-GFN	0.90±0.03	1.48±0.14	0.94±0.01	1.28±0.05	0.00±0.00	6.67±0.67	1.53±0.49	1.57± 0.42
DPaO (Ours)	0.91±0.01	1.39±0.03	0.94±0.01	1.25±0.03	**0.04±0.01**	**4.31±0.20**	**2.27± 0.30**	1.84± 0.83

have significantly lower HV in irregularly shaped DNA and molecule optimization (see Table 19) tasks. Notably, we achieve these results with fixed β value, demonstrating the robustness of directly optimizing for the ordering of the solutions instead of decomposing the problem into series of single-objective optimization tasks. We further study the robustness of DPaO against the choice of β in Appendix C.

5.4 Does policy reward correspondence hold in practice?

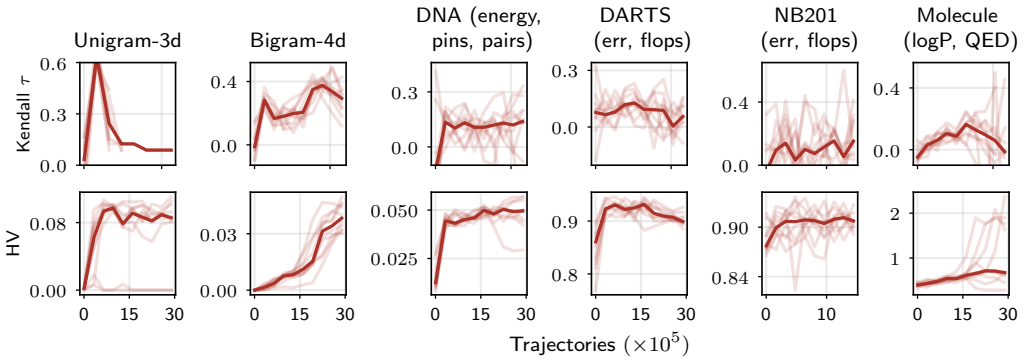


Figure 5: **DPaO attains positive rank correlation with Pareto ordering**. Figure display Kendall τ (top) and hypervolume (HV) (bottom) as a function of generated trajectories in various environments. The dark line represents median, while lighter lines represent individual seeds. DPaO can hold positive rank-correlation with the true Pareto ordering in all environments.

As noted in Section 4, there exists an analytical closed form solution that connects the *optimal* policy and the latent reward function. However, as we do not have access to the optimal policy, we approximate it with the currently learned policy. To validate this approximation, we compute the Kendall-τ (KD) rank-correlation (Kendall et al., 1938) between the log-probabilities of the generated solutions, and the ranking obtained by using Eq. 1. Figure 5 displays KD (top) and HV (bottom) of the learned policy as a function of generated trajectories in various environments. We highlight that *DPaO retains positive rank correlation in all tasks*, indicating that approximating the optimal policy with the currently learned policy is a practically viable solution. We also note that the rank correlation tends to stabilize when the algorithm converges, displaying that despite the challenging learning dynamics, caused by the constantly changing reward, DPaO remains stable.

5.5 Is Pareto-dominance enough?

Compared to total order, learning distribution over the partial order requires additional care. To this end, we study if 1) if using auxiliary metrics for breaking the ties produces similar performance, and 2) if modeling the whole distribution over the rankings is necessary, or if the pairwise BT model achieves similar performance. For auxiliary metrics, we consider crowding distance (Deb et al., 2002), which assigns lower rank to solutions further away from other solutions, thus encouraging diversity, and hypervolume contribution (Bringmann & Friedrich, 2012; Guerreiro & Fonseca, 2018), which assigns lower rank to points based on how much the hypervolume is decreased by removing them from the solution set.

Figure 6 showcases various performance metrics of the PbRL variants with tie-breaking as a function of generated trajectories. We can draw two main conclusions from Figure 6. Firstly, *DPaO outperforms the tie breaking variants in all environments*. This indicates that the secondary metrics may not be needed for learning an useful latent reward function. Notably, the tie-breaking variants have similar rank correlation with the true ordering than DPaO, yet their performance falls short of what DPaO can achieve. Indeed, auxiliary metrics prefer some solutions over others, even if they are of same worth, thus biasing the learned reward function, resulting in overall worse performance. Secondly, we note that *PL based variants surpass the pairwise BT model variants* demonstrating

that modeling the distribution over the whole ranking is beneficial when learning a latent reward function. These results further support our choice of modeling MOO as a learning-to-rank task.

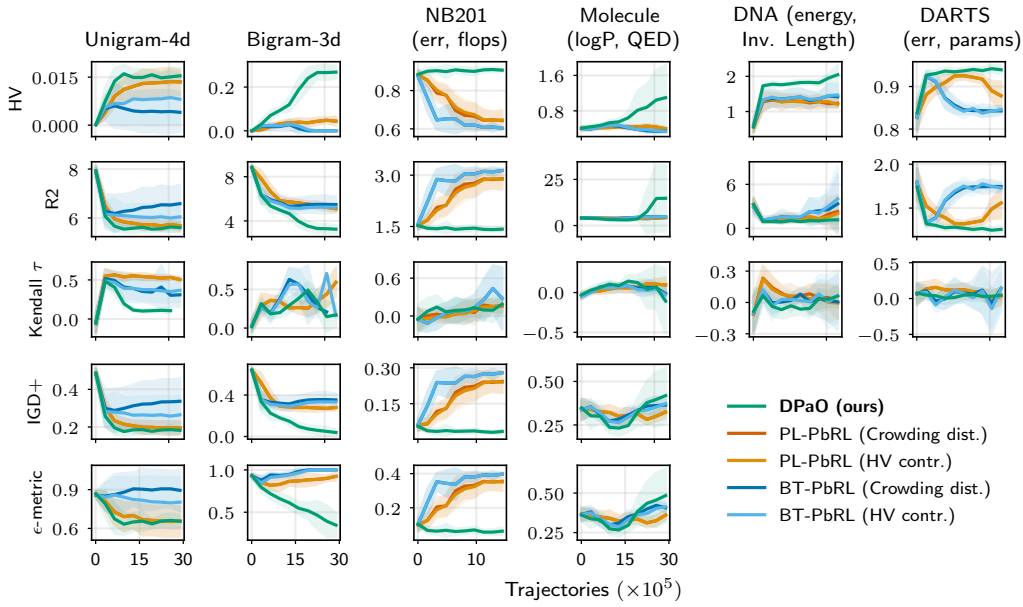


Figure 6: **DPaO outperforms PbRL variants with tie-breaking mechanism**. Figure shows mean $\pm$ std of HV ($\Uparrow$), R2 ($\Downarrow$), Kendall-τ ($\Uparrow$), IGD+ ($\Downarrow$) and ϵ-metric ($\Downarrow$) as a function of generated trajectories in various environments. DPaO outperforms or matches variants with tie-breaking mechanism in all environments in both convergence speed and final performance.

6 Conclusion & Limitations

This paper introduced Direct Pareto Optimization (DPaO), that transforms a multi-objective optimization problem into a single-objective problem over a learned latent reward. DPaO directly models the probability over partial ranking of solutions, side-stepping need for environment specific information, auxiliary perfomance indicators or scalarization. This is achieved by utilizing recent advances in Preference based RL, yielding one end-to-end policy optimization objective with only one hyperparameter, to which the method is robust to. Despite its consistent performance, DPaO has various limitations.

Firstly, as shown in Section 5.2, in some environments, the generated solutions are not exceptionally diverse. This is a trade-off: by optimizing for the Pareto optimality, defined by the reward, we assume that it captures the user intent with enough accuracy, thus making diversity less desirable. Nevertheless, GFlowNets are known to excel in generation of diverse candidates (Jain et al., 2023; Jiralerspong et al., 2026). Therefore, we see the application of ideas from DPaO to GFlowNets via policy gradients (Tiapkin et al., 2024; Niu et al., 2024) as a promising avenue for future work.

Moreover, in it's current form, DPaO is restricted to discrete state- and action-spaces, a limitation caused by our choice to use formulation similar to Rafailov et al. (2023). To extend ideas from DPaO to non-discrete domains, one could train a separate reward model or utilize regret-based PbRL framework (Knox et al., 2024a;b), known to work even with partial returns.

Lastly, compared to the decomposition based methods, we lose *controllability* of the learned policy, which may be desirable when using the final policy for e.g. robot control. While this is not an issue in design applications, where the final designs are validated before going through expensive real-world evaluations, it is an intriguing problem. To this end, we see combination of our latent reward learning with goal-conditioned approach (Roy et al., 2023) as path to integrate controllability to the learned latent function, without losing its desirable properties.

AI USE STATEMENT

We have not used generative AI tools for dataset generation, or generating theoretical or conceptual frameworks. We have not used generative AI for proving mathematical claims, writing proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments. Moreover, generative AI has not been used to implement methods, to clean and reformat data, or perform qualitative or thematic data-analysis, or to interpret any results. We have used Generative AI systems to i) improve the writing of this paper, and to ii) improve scientific figures shown in the paper. No AI output has been used directly in this paper, rather the outputs of AI has used as a motivation. Therefore, we have reviewed all AI outputs before they have influenced any decisions related to the contents of this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The presented method, Direct Pareto Optimization, is a general-purpose multi-objective optimization method for sequential design. While not harmful on it's own, it could be misused to design better performing solutions for adversarial applications. Therefore, we emphasize the need for safeguards to mitigate possible risks associated when applying these methods to new applications.

REPRODUCIBILITY STATEMENT

To ensure reproducibility of our results, we provide supplementary materials with our submissions. More specifically, we include *initial* version of our source code, which contains both our proposed method DPaO and the implementations of baselines used to produce the results in this paper. We also include the exact configurations, with some examples used to produce the results presented in the main paper. Furthermore, in Appendix A, we include further definitions and explanations related to Pareto compliance, which may be helpful when trying to understand the proposition of our method.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In A Beygelzimer, Y Dauphin, P Liang, and J Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation. In *Advances in Neural Information Processing Systems*, 2021.

Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Karl Bringmann and Tobias Friedrich. Approximating the least hypervolume contributor: Np-hard in general, but fast in practice. *Theoretical Computer Science*, 425:104–116, 2012. Theoretical Foundations of Evolutionary Computation.

Krishnendu Chatterjee, Rupak Majumdar, and Thomas A Henzinger. Markov decision processes with multiple objectives. In *STACS 2006*, 2006.

Yihang Chen and Lukas Mauch. Order-Preserving GFlowNets. In *The Twelfth International Conference on Learning Representations*, 2023.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

Suresh Dara, Swetha Dhamercherla, Surender Singh Jadav, Ch Madhu Babu, and Mohamed Jawed Ahsan. Machine learning in drug discovery: A review. *Artificial intelligence review*, 55(3):1947–1999, 2021.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. In *Advances in Neural Information Processing Systems*, 2020.

Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *Advances in Neural Information Processing Systems*, 2021.

Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Multi-objective Bayesian optimization over high-dimensional search spaces. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, 2022.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *International Conference on Learning Representations*, 2019.

Michael T. M. Emmerich. *Single- And Multi-Objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels*. PhD thesis, Universität Dortmunt, 2005.

Michael T. M. Emmerich, André H. Deutz, and Jan Willem Klinkenberg. Hypervolume-based expected improvement: Monotonicity properties and exact computation. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, 2011.

Mingfeng Fan, Jianan Zhou, Yifeng Zhang, Yaoxin Wu, Jinbiao Chen, and Guillaume Adrien Sartoretti. Preference-Driven Multi-Objective Combinatorial Optimization with Conditional Computation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Arvid Frydenlund, Gagandeep Singh, and Frank Rudzicz. Language Modelling via Learning to Rank. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10), 2022.

Andreia P. Guerreiro and Carlos M. Fonseca. Computing and updating hypervolume contributions in up to four dimensions. *IEEE Transactions on Evolutionary Computation*, 22(3):449–463, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Michael Pilegaard Hansen and Andrzej Jaszkiewicz. *Evaluating the quality of approximations to the non-dominated set*. IMM, Department of Mathematical Modelling, Technical University of Denmark, 1998.

Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. A Practical Guide to Multi-Objective Reinforcement Learning and Planning. *Autonomous Agents and Multi-Agent Systems*, 36(1), 2022.

Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, and Yusuke Nojima. Modified Distance Calculation in Generational Distance and Inverted Generational Distance. In *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, 2015.

Hisao Ishibuchi, Ryo Imada, Naoki Masuyama, and Yusuke Nojima. Use of two reference points in hypervolume-based evolutionary multiobjective optimization algorithms. In *Parallel Problem Solving from Nature – PPSN XV"*, pp. 384–396. Springer International Publishing, 2018a.

Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. How to specify a reference point in hypervolume calculation for fair performance comparison. *Evolutionary Computation*, 26(3): 411–440, 2018b.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-Objective GFlowNets. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Marco Jiralerspong, Esther Derman, Danilo Vucetic, Esmeralda S. Whitammer, Bilun Sun, Tianyu Zhang, Pierre-Luc Bacon, and Gauthier Gidel. Discrete compositional generation via general soft operators and robust reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Maurice G Kendall et al. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

W. Bradley Knox, Stephane Hatgis-Kessell, Sigurdur Orn Adalgeirsson, Serena Booth, Anca Dragan, Peter Stone, and Scott Niekum. Learning Optimal Advantage from Preferences and Mistaking It for Reward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(9), 2024a.

W. Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, and Alessandro G Allievi. Models of human preference for learning reward functions. *Transactions on Machine Learning Research*, 2024b.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4), 2020.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, 2020.

Guy Lebanon and Yi Mao. Non-parametric modeling of partially ranked data. *Journal of Machine Learning Research*, 9(79):2401–2429, 2008.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.

Haoye Lu, Daniel Herman, and Yaoliang Yu. Multi-Objective Reinforcement Learning: Convexity, Stationarity and Pareto Optimality. In *The Eleventh International Conference on Learning Representations*, 2023.

Tyler Lu and Craig Boutilier. Effective sampling and learning for mallows models with pairwise-preference data. *Journal of Machine Learning Research*, 15(117):3963–4009, 2014.

Zhichao Lu, Ran Cheng, Yaochu Jin, Kay Chen Tan, and Kalyanmoy Deb. Neural Architecture Search as Multiobjective Optimization Benchmarks: Problem Formulation and Performance Assessment. *IEEE Transactions on Evolutionary Computation*, 28(2), 2024.

Duncan Robert Luce. *Individual choice behavior: A theoretical analysis*. John Wiley, 1959.

Jiaqi Ma, Xinyang Yi, Weijing Tang, Zhe Zhao, Lichan Hong, Ed Chi, and Qiaozhu Mei. Learning-to-Rank with Partitioned Preference: Fast Estimation for the Plackett-Luce Model. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021.

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

Henry B Mann and Donald R Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pp. 50–60, 1947.

Ofir Nachum, Mohammad Norouzi, and Dale Schuurmans. Improving policy gradient by exploring under-appreciated rewards. In *International Conference on Learning Representations*, 2017.

Puhua Niu, Shili Wu, Mingzhou Fan, and Xiaoning Qian. GFlowNet Training by Policy Gradients. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Mingjun Pan, Guanquan Lin, You-Wei Luo, Bin Zhu, Zhien Dai, Lijun Sun, and Chun Yuan. Preference Optimization for Combinatorial Optimization Problems. In *Forty-second International Conference on Machine Learning*, 2025.

Lie Meng Pang, Hisao Ishibuchi, Linjun He, Ke Shang, and Longcan Chen. Hypervolume-based cooperative coevolution with two reference points for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 28(4):1054–1068, 2024.

R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(2), 1975.

Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *Advances in Neural Information Processing Systems*, 36, 2023.

Diederik M. Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Julien Roy, Pierre-Luc Bacon, and Christopher Pal andEmmanuel Bengio. Goal conditioned gflownets for controllable multi-objective molecular design. Published at ICML2023 Deployable Generative AI workshop, 2023.

Jatan Shrestha, Santeri Heiskanen, Kari Hepola, Severi Rissanen, Pekka Jääskeläinen, and Joni Pajarinen. Pareto-conditioned diffusion models for offline multi-objective optimization. In *International Conference on Learning Representations*, 2026.

Adam Stafa, Santeri Heiskanen, Petr Novotny, and Joni Pajarinen. Momba: Network modernization improves multi-objective reinforcement learning. *Reinforcement Learning Journal*, 7, 2026.

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. In *Proceedings of the 39th International Conference on Machine Learning*, 2022.

Daniil Tiapkin, Nikita Morozov, Alexey Naumov, and Dmitry P. Vetrov. Generative Flow Networks as Entropy-Regularized RL. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, 2024.

Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84 (1), 2011.

Kristof Van Moffaert, Madalina M. Drugan, and Ann Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2013.

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

John I. Yellott. The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15(2): 109–144, 1977.

Joseph N. Zadeh, Conrad D. Steenberg, Justin S. Bois, Brian R. Wolfe, Marshall B. 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.

Yiyang Zhao, Linnan Wang, Kevin Yang, Tianjun Zhang, Tian Guo, and Yuandong Tian. Multi-objective optimization by learning space partition. In *International Conference on Learning Representations*, 2022.

Yiyang Zhao, Linnan Wang, and Tian Guo. Multi-Objective Neural Architecture Search by Learning Search Space Partitions. *Journal of Machine Learning Research*, 25(177), 2024.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, volume 8, pp. 1433–1438, 2008.

E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature - PPSN VIII*, pp. 832–842. Springer Berlin Heidelberg, 2004.

Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In *Parallel Problem Solving from Nature — PPSN V*, 1998.

Eckart Zitzler, Joshua Knowles, and Lothar Thiele. *Quality Assessment of Pareto Set Approximations*, pp. 373–404. Springer Berlin Heidelberg, 2008.

A PARETO COMPLIANCE & PARTITIONED PREFERENCE

In this section, we present the exact definition of Pareto compliance, a key property for multi-objective performance indicators, which our method obtains by definition. Moreover, we provide further explanation on how the definition of Pareto compliance is related to the definition of *partitioned preference* (Lebanon & Mao, 2008; Ma et al., 2021).

We begin by extending the Pareto dominance relation (Definition 3) for two sets, following Zitzler et al. (2003). Firstly, let A be set of reward vectors. We call A *non-dominated* when no reward vector in A is dominated by any other reward vector in A. We also use $|A|$ to denote the cardinality of A. Let $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \dots \boldsymbol{a}_{|A|}\}$ and $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_{|B|}\}$ be two non-dominated sets of objective vectors. Then the *Pareto dominance* for sets is defined as

$$A \succ B \Leftrightarrow \forall \boldsymbol{b}_i \in B, \exists \boldsymbol{a}_i \in A : \boldsymbol{a}_i \succ \boldsymbol{b}_j \tag{6}$$

Moreover, we can define *Weak Pareto dominance* for sets as

$$A \succeq B \Leftrightarrow \forall \boldsymbol{b}_i \in B, \exists \boldsymbol{a}_i \in A : \boldsymbol{a}_i \succeq \boldsymbol{b}_j \tag{7}$$

Notably $A \succ B$ requires that A and B are disjoint, while $A \succeq B$ allows $A = B$. That is to say, A and B can be the same set. For handling partially overlapping sets, Zitzler et al. (2003) proposed "better" relation for sets as

$$A \triangleright B \Leftrightarrow A \succeq B \text{ and } A \neq B \tag{8}$$

Using, these definitions, we can define Pareto compliance and weak Pareto compliance as follows:

Definition 3 (Pareto Compliance)**.** *Given two non-dominated sets A and B, we say that an indicator is Pareto compliant if whenever $A \triangleright B$, then $I(A) < I(B)$; $A \triangleright B \Rightarrow I(A) < I(B)$*

Definition 4 (Weak Pareto Compliance)**.** *Given two non-dominated sets A and B, we say that an indicator is Pareto compliant if whenever $A \triangleright B$, then $I(A) \leq I(B)$; $A \triangleright B \Rightarrow I(A) \leq I(B)$*

In the above definitions, I is an arbitrary scalar valued indicator function. Intuitively, a (weakly) Pareto compliant indicator will never produce a result that is in contradiction of Pareto dominance.

Firstly, we note that by definition, Pareto ranking from Eq. 1 is Pareto compliant. This is notable, since hypervolume, the other Pareto compliant performance indicator, requires access to environment specific information. Secondly, we note that the sets created by Pareto ranking will be 1) disjoint, and 2) non-dominated. The latter point implies that one cannot assign order to solutions belonging to the same set. Notably, this matches the exact definition of *Partitioned preference* (Lebanon & Mao, 2008; Ma et al., 2021), used in learning-to-rank literature. This relation between the partition created by Pareto ranking and partitioned preference allows us to directly model the multi-objective optimization task as a learning-to-rank task.

B EFFICIENT COMPUTATION OF LOG-LIKELIHOOD OF MIXTURE OF PLACKETT-LUCE MODELS

Below, we introduce the approach used to compute the log-likelihood of mixture of PL models in $O(N + S_{\max}^3)$ time, where N is the size the batch and $S_{\max}$ is the size of the largest set in the top $M-1$ partitions. We first restate the probability of observing a certain ranking over the partitioned preference from Section 4:

$$p(S_1 \succ S_2 \succ \cdots \succ S_M; \boldsymbol{w}) = \sum_{\substack{(x_1,\ldots,x_N)\in \\ \Omega(S_1 \succ S_2 \succ \cdots \succ S_M)}} \prod_{i=1}^{N} \frac{\exp(w_{i_l})}{\sum_{k=i}^{N} \exp(w_{i_r}))} \tag{9}$$

where $\boldsymbol{w}$ denote the utility scores the solutions, such as the log-probabilities of the solutions in our case. To compute the log-likelihood of Eq. 9 efficiently, we introduce Lemma 1 and Proposition 1 by Ma et al. (2021):

Lemma 1 (Lemma 1 by Ma et al. (2021))**.** *Let $\Omega(\cdot)$ be a function that maps a set of items to the set of all possible permutations of these items. Then Eq. 9 can be rewritten as:*

$$\prod_{m=1}^{M-1} \left(\sum_{(i_1,\ldots,i_{n_m})\in\Omega(S_m)} \prod_{l=1}^{|m|} \frac{e^{w_{i_l}}}{\sum_{j\in R_m} e^{w_j} - \sum_{r=1}^{l-1} e^{w_{i_r}}} \right)$$

Which can be further expressed as

$$\prod_{m=1}^{M-1} p(S_m \succ R_{m+1})$$

where R_m is the set of items that do not belong to the top $m-1$ partitions. That is to say $R_m = \cup_{r=m}^{M} S_r$.

Proposition 1 (Proposition 1 by Ma et al. (2021))**.** *Given PL model parameterized by $\boldsymbol{w}$, for any A, B such that $A \cap B = \emptyset$, the probability of $A \succ B$ is given by*

$$\begin{aligned} p(A \succ B, \boldsymbol{w}) =& p(\min_{a\in A} g_{w_a} > \min_{b\in B} g_{w_b}) \\ =& \int_{u=0}^{1} \prod_{a\in A} (1 - u^{\exp(w_a - w_B)}) du \end{aligned} \tag{10}$$

where g_w denotes a random variable following Gumbel(w) and $w_B = \log \sum_{b\in B} \exp(w_b)$.

Notably, the second equality allows one to efficiently compute the likelihood of a preference over two disjoint partitions. It then directly follows from Lemma 1 and Proposition 1 that the probability of observing a certain ranking over the partitioned preference can be written as (Ma et al., 2021)

$$\begin{aligned} & p(S_1 \succ \cdots \succ S_m; \boldsymbol{w}) \\ =& \prod_{m=1}^{M-1} p(S_M \succ R_{m+1}; \boldsymbol{w}) \\ =& \prod_{m=1}^{M-1} \int_{u=0}^{1} \prod_{i\in S_m} (1 - u^{\exp(w_i - w_{R_{m+1}})}) du \end{aligned} \tag{11}$$

where $w_{R_m} = \log \sum_{j\in R_{m+1}} \exp(w_j)$, consisting of the utilities of solutions not belonging to the top $M-1$ ranked sets. Then the log-likelihood can be written as

$$\begin{aligned} \log p(S_1 \succ \ldots S_m; \boldsymbol{w}) &= \sum_{m=1}^{M-1} \log \left(\int_{u=0}^{1} \prod_{i\in S_m} \left(1 - u^{\exp(w_i - w_{R_{m+1}})}\right) du \right) \\ &\simeq \sum_{m=1}^{M-1} \log \left(\sum_{t=1}^{T} \prod_{i\in S_m} \left(1 - u_t^{\exp(w_i - w_{R_{m+1}})}\right) \right) \end{aligned} \tag{12}$$

where in the second line the exact integral is replaced with numerical approximation. Finally Equation 12 can be written as

$$\begin{aligned}
&\sum_{m=1}^{M-1} \log \left(\sum_{t=1}^{T} \prod_{i \in S_m} \left(1 - u^{\exp(w_i - w_{R_{m+1}})} \right) \right) \\
&= \sum_{m=1}^{M-1} \log \left(\sum_{t=1}^{T} \exp \left(\log \left(\prod_{i \in S_m} \left(1 - u_t^{\exp(w_i - w_{R_{m+1}})} \right) \right) \right) \right) \\
&= \sum_{m=1}^{M-1} \log \left(\sum_{t=1}^{T} \exp \left(\sum_{i \in S_m} \log \left(1 - u_t^{\exp(w_i - w_{R_{m+1}})} \right) \right) \right)
\end{aligned} \tag{13}$$

where the RHS of the last equality contains the final expression used to compute the log-likelihood in practice. Both the inner $\log(1-x)$ and the outer log-sum-exp operations have numerically stable implementations (Ma et al., 2021). Thus the log-likelihood computation can be simplified into a single loop over the rankings, which can be implemented easily in any modern machine learning frameworks such as Jax.

C ABLATION OVER ENTROPY COEFFICIENT β

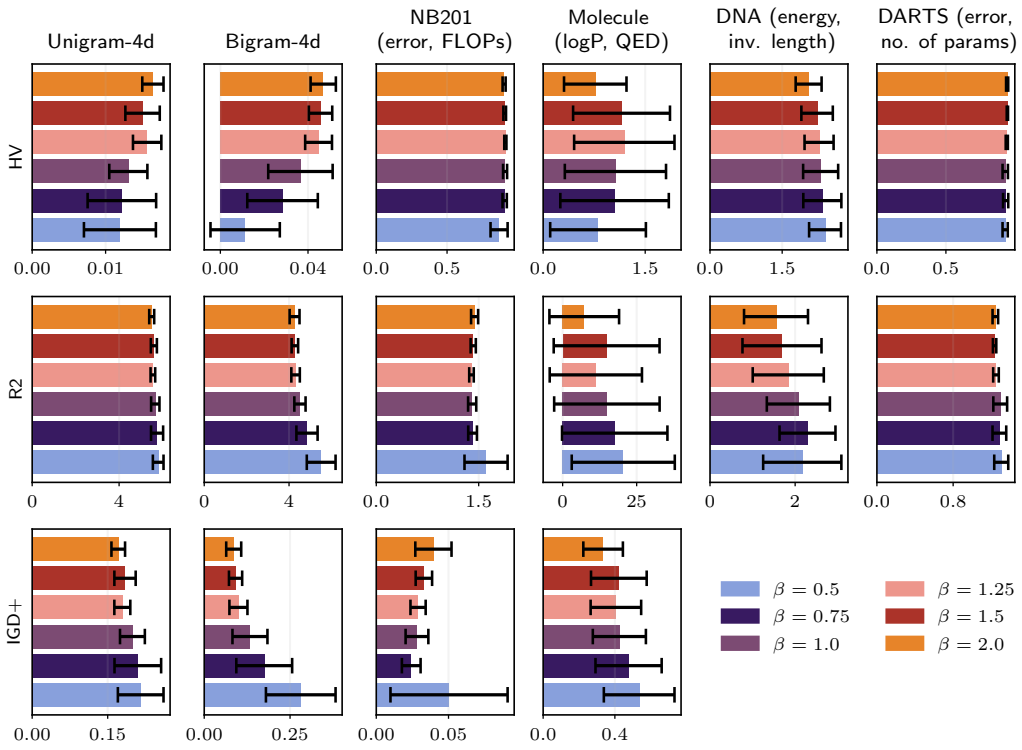


Figure 7: **DPaO retains similar performance with varying β.** Figure showcases the effect of changing β in various environments. The plots showcase mean $\pm$ std over 10 seeds. DPaO is robust to changing β, and more importantly, consistent across environments.

While DPaO simplifies the hyperparameter selection common in decomposition based MOO methods to a single hyperparameter β, it raises the question, how sensitive DPaO is to the entropy regularization effect? Intuitively, having low entropy could lead to the model getting stuck on generating solutions with low rewards, making it difficult for the model to improve. To this end, we perform a sweep over β in various environments and display the final performance in Figure 7. As one can see from the results, DPaO is quite robust to the changes in β, achieving good performance when $\beta \in [0.75, 1.5]$ in all environments. The consistency across environments allows a practitioner tune the parameter in a cheap-to-compute synthetic environment, such as REGEX, and then use the same value in other environments.

D COMPUTATIONAL COMPLEXITY

As noted in Section 4, computing the update of DPaO efficiently requires some extra care, raising concerns about computational complexity in practical terms. Compared to decomposition based methods, DPaO has two steps that increase it computational complexity: firstly, we must compute the Pareto ranking to obtain the partitioned preference. In practice, this is done by performing an non-dominated sort, which has complexity of $O(N \cdot D^2)$, where N is the number of solutions and D is the number of objectives (Deb et al., 2002). Secondly, we must compute the log-likelihood (and gradients) over the mixture of PL-models. As mentioned in Section 4, we utilize the random utility formulation by Ma et al. (2021) (details in Appendix B), which brings down the computational complexity of the log-likelihood to $O(N+S_{\max}^3$, where $S_{\max}$ is the size of the largest set in the $M-1$ partitions. Therefore, the total time-complexity of DPaO is $O(N \cdot D^2+N+S_{\max}^3) \approx O(N \cdot D^2+N)$, since usually $S_{\max} \ll N$. This indicates that in theory, the cost of performing non-dominated sort will dominate the computational complexity when the number of objectives increases. To test if this is an issue in practice, we perform a study, where the number of objectives is increased from 2 to 6 in the unigram environments. We compute the average update time during each run over 10 seeds, and plot the results in Figure 8.

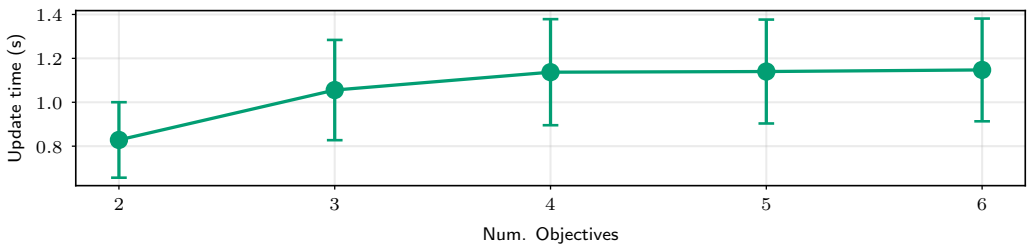


Figure 8: **DPaO retains acceptable runtime with increasing number of objectives** . Figure shows the average $\pm$ std time of a policy update as function of number of objectives. DPaO's update time is negligible even when considering 6 objectives.

As one can see from Figure 8, the practical computational complexity is reasonable, taking at most $\approx 1.2s$ when the number of objectives is increased. Interestingly, we note that the actual runtime seems to converge to a constant value when $D > 4$. This is likely due to the fact that we use highly optimized C++ implementation of the non-dominated sort, while the log-likelihood computation is done in Python. We also note that the implementation of the log-likelihood computation could be further optimized, using techniques similar to Frydenlund et al. (2022). Regardless, we would like to point out that in practice, the computational complexity of the algorithms is dominated by the time used to evaluate e.g. the DNA sequences generated by the algorithm, making the computational complexity DPaO negligible.

E BASELINES

Table 2: Overview of the considered baselines. We consider Policy-gradient methods, Actor-Critic methods and GFlowNet methods, based on both decomposition and ordering approaches.

Method	**Principle**	**Type**	**Scalarization functions**
MOReinforce	Decomposition	Policy-Gradients	Linear & Chebyshev
MOURex	Decomposition	Policy-Gradients	Linear & Chebyshev
POCCO	Decomposition	Policy-Gradients	Linear & Chebyshev
CAPQL	Decomposition	Actor-Critic	Linear
MOGFN	Decomposition	GFN	Linear
OP-GFN	Ordering	GFN	N/A

Table 2 summarizes the main properties of the used baselines, which we will explain in more details below. As one can notice, most of the baselines are base on the decomposition approach, while OP-GFN utilize approach more similar to us.

Firstly, let us describe the policy-gradient (PG) based methods. **MOReinforce** (Lin et al., 2022) is an adaptation of original REINFORCE (Williams, 1992) to multi-objective setting. MOReinforce trains a conditioned policy $\pi(\cdot|\boldsymbol{\omega})$, using the original PG objective on the scalarized returns, and making it a decomposition based method. Following Kwon et al. (2020), we use the mean of the generated batch as the baseline to reduce the variance of the PG estimate.

UREX (Nachum et al., 2017) is an variant of traditional PG that seeks to improve the exploration of PG methods without relying on entropy-regularization. Specifically, UREX optimizes the *mean-seeking* direction of the KL-divergence between the learned and optimal policy, compared to the entropy-regularized PG, which optimizes the mode-seeking direction. This requires sampling from the optimal policy, which is not possible. Therefore, UREX utilizes self-normalized importance sampling to approximate it's objective. Intuitively, UREX places more emphasis on reward regions *under-appreciated* by the current policy. We adapt this to multi-objective setting by training a conditioned policy $\pi(\cdot|\boldsymbol{\omega})$ and optimizing the scalarized objective, similarly to MOReinforce.

POCCO (Fan et al., 2025) is a recent method that, similarly to us, utilizes the closed form solution to the entropy-regularized optimization task, and optimizes the pairwise Bradley-Terry model. To obtain a full-ordering over the solution-space, Fan et al. (2025) apply first scalarization to the obtained vector rewards, and train a conditioned policy $\pi(\cdot|\boldsymbol{\omega})$. As our goal is to study the algorithmic improvements,we only adapt the algorithmic improvements in POCCO, without the architectural improvements, such as using hypernetworks.

We use POMO-style rollouts (Kwon et al., 2020) for all the above mentioned policy-gradient based methods, since we found them to consistently improve the performance. More specifically, during training, the first action of each sequence is taken from a random policy. Moreover, for these PG based methods, we consider both the linear $f(\boldsymbol{R}(x), \boldsymbol{\omega}) = \boldsymbol{\omega}^T \boldsymbol{R}(x)$, and Chebyshev scalarization (Van Moffaert et al., 2013)$f(\boldsymbol{R}(x), \boldsymbol{\omega}) = \boldsymbol{\omega}|r_i - z_i^*|$, where $\boldsymbol{z}^*$ is utopian point, dependent on the environment.

CAPQL (Lu et al., 2023) is an entropy regularized Actor-Critic MORL method, which resembles Soft-Actor-Critic (Haarnoja et al., 2018). To adapt CAPQL to sequential design tasks, we perform the following changes: 1) following recent research, we directly train a conditioned Q-network to directly predict the scalarized Q-values instead of predicting the Q-values separately for each reward component (Stafa et al., 2026). 2) Instead of sampling a new preference for each timestep, as in the original paper, we fix the preference for each trajectory. This is natural, since the reward is received only when completing the sequence, and thus only the final preference directly affects the returns.

MOGFN (Jain et al., 2023) is an adaption of GFNs (Bengio et al., 2021) to multi-objective setting. MOGFN trains a conditioned policy such that $\pi(\cdot|\boldsymbol{\omega}) \propto f(\boldsymbol{R}(x), \boldsymbol{\omega})$, utilizing the trajectory balance objective (Malkin et al., 2022). In practice, this is achieved by sampling preferences from a Dirichlet distribution, and scalarizing the vector-valued rewards using linear scalarization. In contrast to most single-objective GFNs, MOGFN does not use replay-buffer. We also confirmed that using replay-buffer does not improve the performance.

OP-GFN (Chen & Mauch, 2023) is a recent GFN development that, similarly to us, tries to infer a reward function consistent with the ordering produced by Pareto optimality. However, OP-GFN considers only the one-hot distribution created by Pareto-optimality, and minimizes the KL-divergence between the learned distribution and this target distribution. We follow implementation details specified by Chen & Mauch (2023), such as using replay buffer, tempering the sampling policy with uniform-policy, akin to ϵ-greedy exploration and having separate sampling and training policies, where the sampling policy is updated via soft updates.

F ENVIRONMENTS

In section, we summarize the environments used in main experiments in Table 3, and provide more detailed description below.

Table 3: Summary of the benchmarks used in the paper. Regex, and NAS benchmarks have near-convex solution sets, while DNA and molecule generation tasks provide irregularly shaped solution sets.

Task	$\lvert\mathcal{X}\rvert$	Objectives	#. Objectives
Regex	20^{32}	# of predetermined strings	2-4
NB201	15 625	Error, FLOPs	2
DARTS	$\approx 10^{18}$	Error, FLOPs, # of Parameters	2-3
DNA	4^{60}	Free energy, # of base-pairs, # of hairpins, inv. length	2-3
Molecule	N/A	logP, QED	2

Regex (Stanton et al., 2022) is a task, where one is tasked to generate sequences of length 32 to 36 tokens from vocabulary $|\mathcal{V}| = 20$, while optimizing number of occurrences of predetermined strings. In **Unigram** tasks, the goal objectives are conflicting, whereas in **Bigrams**, they are correlated, due to shared letters in the target strings. We use 2D, 3D and 4D variants of both Uni- and Bigram tasks. We use normalize the rewards to be between $[0, 1]$, and set the reference point to zero vector $\boldsymbol{r} = \mathbf{0}_d$ and utopian point to $\boldsymbol{z}^* = 1.1 \times \mathbf{1}_d$. Moreover, we generate points from a d-dimensional simplex to act as an reference set.

NB201 (Dong & Yang, 2019) tasks one to construct an neural network block from 5 predefined operations, with predefined topology, consisting of 6 choices. This macroblock then repeated multiple times before the architecture is completed. The goal of this task is to maximize test accuracy based on validation accuracy, while minimizing the number of floating point operations (FLOPs). We focus on the downscaled imagenet variant of the benchmark, as it offers the most broad performance distribution. We define the task of finding proper architecture as a sequential optimization task, were we add new operations to the block in predetermined order. Since we can enumerate all the states, we normalize all rewards between $[0, 1]$ and use reference point $\boldsymbol{r} = \mathbf{0}_d$ and utopian point $\boldsymbol{z}^* = 1.1 \times \mathbf{1}_d$. We use the true Pareto front as reference set.

DARTS (Liu et al., 2019; Lu et al., 2024) is a neural architecture search task, where one seeks to find both the normal and reduction cell used in the architecture skeleton. Each cell is created from 7 operations and 4 intermediate nodes, where each intermediate node has 2 inputs. Each of these cells consists of $\approx 10^9$ possible blocks (without considering graph isomorphisms), and thus the total number of architectures in this search-space is $\approx 10^{18}$. We wish to minimize the validation error while also minimizing both the number of parameters and FLOPs. We use surrogate models from (Lu et al., 2024) in our experiments. Similarly to NB201, we defined the task as adding new operations to the architecture block sequentially. However, in this case we apply masking during the generation to ensure that all the generated solutions are valid. We normalize rewards between $[0, 1]$ using approximated values by Lu et al. (2024), and thus use reference point $\boldsymbol{r} = \mathbf{0}_d$ and an utopian point $\boldsymbol{z}^* = 1.1 \times \mathbf{1}_d$. We do not have access to a reference set in this environment, and thus we evaluate only HV and R2-metric.

DNA aptamer generation (Jain et al., 2023) requires one to generate single-stranded nucleotide sequences by adding one nucleobase (A, C, T or G) to the current sequence at each timestep. We limit the number length of sequences to 60 bases. The goal is to maximize the secondary structure free energy (computed with NUPACK (Zadeh et al., 2011)), the number of base pairs, number of hairpins and the inverse of the sequence length, which incentives the generation of shorter sequences. We use values provided by Jain et al. (2023) to normalize rewards between $[0, 1]$, and use reference point $\boldsymbol{r} = \mathbf{0}_d$ and utopian point $\boldsymbol{z}^* = 1.1 \times \mathbf{1}_d$. We do not have access to a reference set in this environment, and thus we evaluate only HV and R2-metric.

Small Molecule generation task (Stanton et al., 2022) requires one to generate SELFIES (Krenn et al., 2020) encoded molecules. The vocabulary for the strings is extracted from Zinc dataset (Krenn et al., 2020), and the maximum length of the generated strings is set to 128 tokens. The goal

is to optimize for the octanol-water coefficient (logP) and the Quantitative Estimate of Druglikeness (QED), as solutions considering only logP are rarely useful in practice (Stanton et al., 2022). We normalize the rewards between $[0, 1]$ using values from the precomputed dataset and set reference point $\boldsymbol{r} = \mathbf{0}_d$ and utopian point $\boldsymbol{z}^* = 1.1 \times \mathbf{1}_d$. We use the best non-dominated set in the dataset as our reference set.

G PERFORMANCE INDICATORS

In this section, we describe the performance indicators used in this paper for evaluating the solution sets. In all definitions below, we use $S = \{\boldsymbol{s}^i\}_{i=i}^{|S|}$ denote the rewards of generated solutions , $P = \{\boldsymbol{p}^j\}_{j=1}^{|P|}$ to refer a reference set and S_{nd} to denote the solution set where the non-dominated points are removed. Moreover, we use $S(i)$ to denote the i^{th} element of the set S and $\boldsymbol{s}_j$ to denote the j^{th} component of an element. d is the number of objectives.

Hypervolume (HV) ($\Uparrow$) (Zitzler & Thiele, 1998): measures the space or volume enclosed by the solutions in the set S

$$\text{HV}(S) = \int_{\mathbb{R}^n} \mathbb{1}_{H(S)}(z)dz$$

where $H(S) = \{z \in Z | \exists i : 0 \leq i \leq |S|, \boldsymbol{r}_0 \preceq z \preceq S(i)\}$. Here $\boldsymbol{r}_0$ is the reference point and $\mathbb{1}_{H(S)}$ is an indicator function. Notably, hypervolume is a *Pareto compliant indicator* (Zitzler et al., 2003). Unfortunately, it's well known that hypervolume is sensitive to the chosen reference point, making it impossible to compare results computed with different reference points (Zitzler et al., 2008; Vamplew et al., 2011; Ishibuchi et al., 2018a;b; Pang et al., 2024).

Inverted Generational Distance+ (IGD+) ($\Downarrow$) (Ishibuchi et al., 2015): Computes the average modified distance to the closest reference point in a reference set for each generated sample. Formally, IGD+ is defined as

$$\text{IGD+}(S, P) = \frac{1}{|P|} \sum_{\boldsymbol{p} \in P} \min_{\boldsymbol{s} \in S} \| \max(\boldsymbol{p} - \boldsymbol{s}, 0)\|_2$$

IGD+ is *weakly Pareto compliant*, and when used in conjunction with uniformly distributed reference set, IGD+ measures both spread and convergence of the solution set (Ishibuchi et al., 2015).

R2-indicator ($\Downarrow$) (Hansen & Jaszkiewicz, 1998) measures the quality of a solution set with respect to an ideal / utopian reference point $\boldsymbol{z}^*$. Formally, it is defined as a expected Chebyshev utility under an uniform distribution of weight vectors Λ:

$$\text{R2}(S, \boldsymbol{z}^*) = \frac{1}{|\Lambda|} \sum_{\lambda \in \Lambda} \min_{\boldsymbol{s} \in S} \max_{i \in 1, \dots d} \{\lambda_i | \boldsymbol{z}_i^* - \boldsymbol{s}_i|\}$$

R2 indicator is known to capture both the convergence and uniformity of the solution set simultaneously (Hansen & Jaszkiewicz, 1998).

ϵ-indicator ($\Downarrow$) (Zitzler et al., 2008) measures how close a solution set S approximates a reference set P. We specifically consider the *additive* ϵ-metric, which is defined as

$$I_\epsilon = \inf_{\epsilon \in \mathbb{R}} \{\forall \boldsymbol{r}' \in P, \exists \boldsymbol{r} \in S : \boldsymbol{r}'_i \leq \boldsymbol{r}_i + \epsilon, \forall i \in \{1, \dots, d\}\}$$

Intuitively, a solution set S is ϵ approximate solution of P if for each element in P, there exists at least one solution in S, such that it's at most ϵ smaller than the reference solution in each objective.

PC entropy ($\Uparrow$) (Roy et al., 2023): PC-entropy measures the uniformity of the generated solution set S with respect to a given reference set P. Firstly, we divide the solution set S_{nd} into a disjoint subset of points $\{\boldsymbol{s} \in S_{\text{nd}} : \forall i \neq j, \|\boldsymbol{s} - \boldsymbol{s}_j\|_2 \leq \|\boldsymbol{s} - \boldsymbol{s}_{j'}\|\}$, after which the entropy for the histogram is defined as

$$\text{PCEnt}(\text{S}_{\text{nd}}, \text{P}) = -\sum_j \frac{|S_{\text{nd}}(j)|}{|P|} \log \frac{|S_{\text{nd}}(j)|}{|P|}$$

PC entropy is maximized when the generated points are uniformly distributed along the reference set.

H IMPLEMENTATION DETAILS

Since our goal is to focus on algorithm design, we utilize same architectural choices for all methods. More specifically, we use a transformer based model with 3 layers and 8 heads as the policy. We use 64 dimensional embeddings, and we fix dropout to 0.1. We use Adam as our optimizer with learning rate 10^{-4}. For GFN methods, we set the learning rate for $\log Z$ to 10^{-3}. We train each policy for 500 episodes, with batch size $B = 64$, resulting in total of $32\ 000$ generated trajectories. During evaluation, we samples 256 solutions, except in NASBench201, were we sample only 32 solutions due to the small size of the search-space. The most important parameters are displayed in Table 4.

Table 4: Summary of the training and policy network details used in the main experiments

Training details		**Network details**	
Parameter	**Value**	**Parameter**	**Value**
Optimizer	Adam	No. of Layers	2
Learning rate	10^{-4}	No. of Heads	8
$\log Z$ learning rate	10^{-3}	No. of output head layers	2
Max gradient norm	10	Embedding dim	64
No. of episodes	500	Output head dim	256
Batch size	32 (NB201) & 64 (Others)	Dropout rate	0.1
No. of eval samples	32 (NB201) & 256 (Others)		

In addition, we display method specific parameters in Table 5. To select the entropy regularization coefficients, we performed grid-search over options, and chose one value for each method based on average performance. For MOGFN, we use separate reward exponents based on values reported by Jain et al. (2023).

Table 5: Summary of method specific parameters

Method	**Parameter**	**Value(s)**
CAPQL	Entropy coefficient	1.0
CAPQL	Target network update freq	10
MOGFN	Reward exponent	{70 (NAS), 80 (DNA & Zinc), 96 (REGEX)}
POCCO	Entropy coefficient	1.25
MOReinforce	Entropy coefficient	0.25
DPaO (ours)	Entropy coefficient	1.25

I EXAMPLES OF THE GENERATED SOLUTION SETS

In this section, we represent examples of the generated solution sets in 2D and 3D cases. Figure 9 showcases solutions in REGEX tasks, Figure 10 displays solutions in NASBench201 and DARTS, while Figure 11 shows solutions in small molecule generation and DNA sequence design. Notably, the solution sets in REGEX and NAS tasks are (nearly) convex, while the solution sets of molecule and DNA sequence generations are irregularly shaped, making the optimization task more challenging.

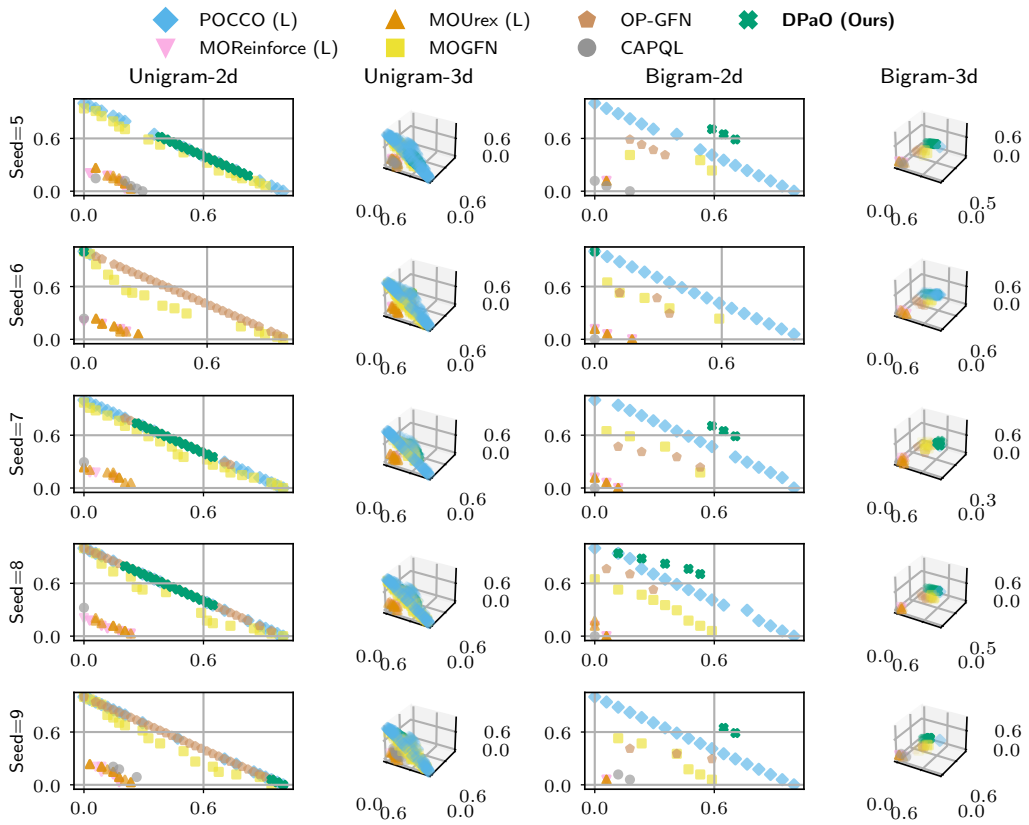


Figure 9: Examples of the generated solution sets in 2D and 3D REGEX environments with seeds 5,6,7,8 and 9. To avoid visual clutter, only the better performing variants of MOReinforce, MOURex and POCCO are shown.

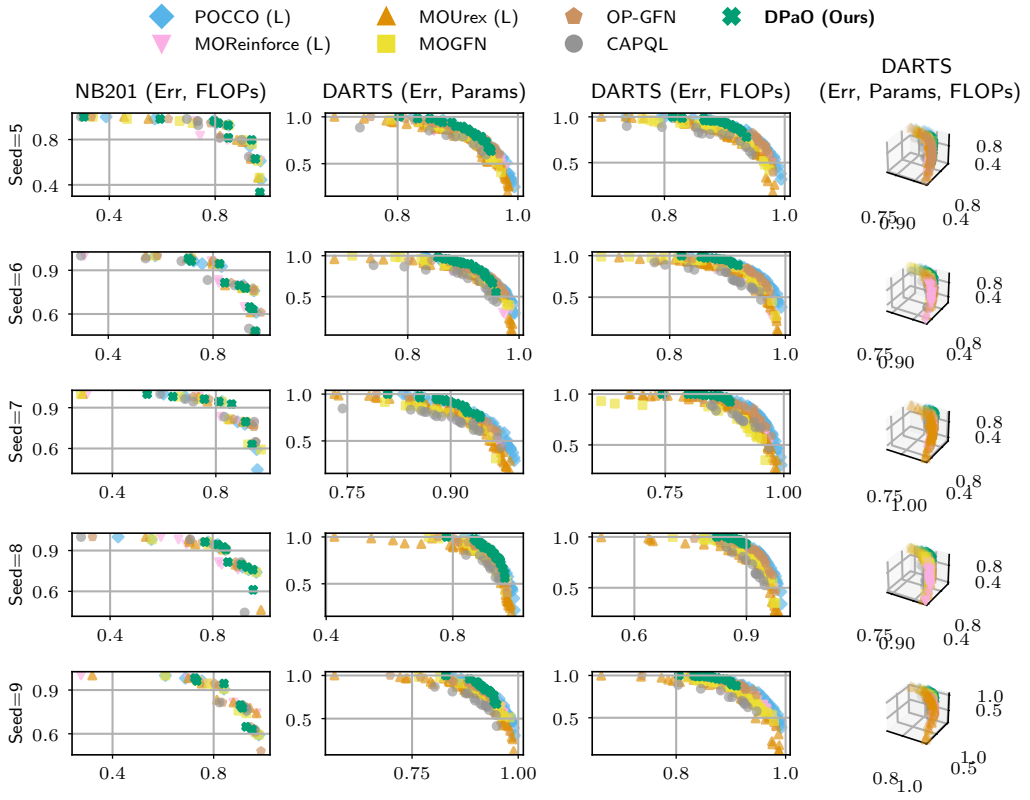


Figure 10: Examples of the generated solution sets in NASBench201 and DARTS environments with seeds 5,6,7,8 and 9. To avoid visual clutter, only the better performing variants of MOReinforce, MOUrex and POCCO are shown.

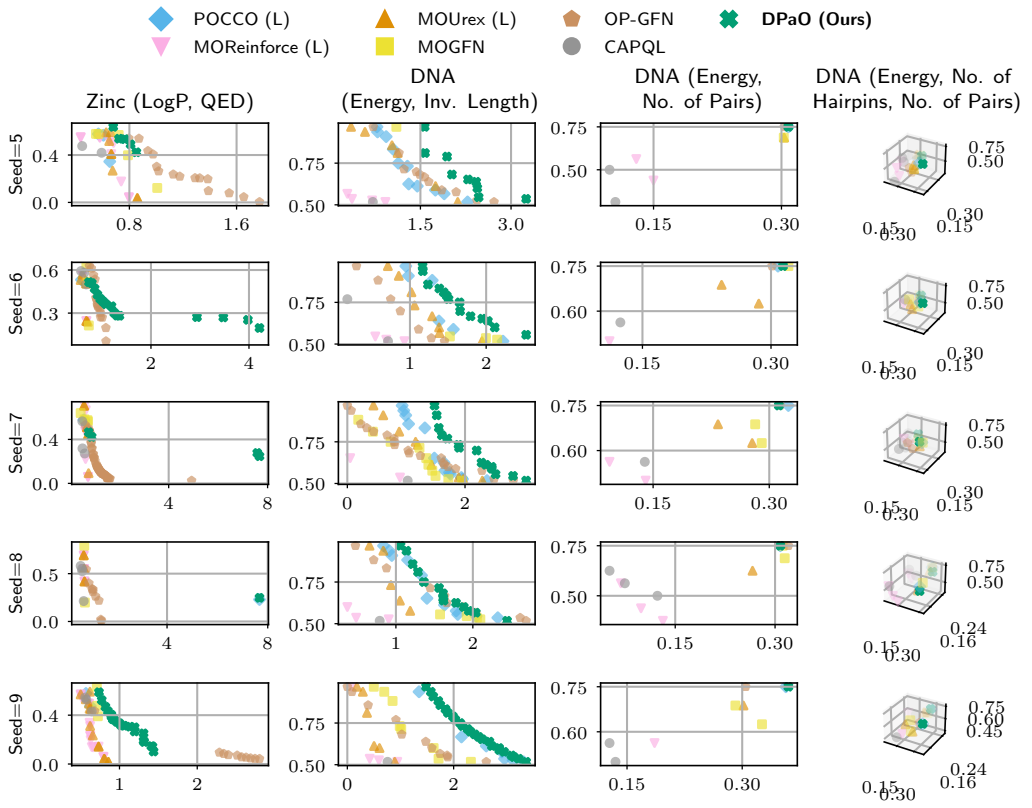


Figure 11: Examples of the generated solution sets in Small molecule task and DNA sequence design environments with seeds 5,6,7,8 and 9. To avoid visual clutter, only the better performing variants of MOReinforce, MOURex and POCCO are shown.

J FULL RESULT TABLES

Below we display the full result tables for all environments considered in the main paper. Tables 6, 7, 8, 9, 10 and 11 display the results in REGEX tasks. Tables 12, 14, 13 and 15 display the results of NAS tasks. Tables 16, 17, 18 and 19 display the results in DNA sequence generation and small molecule tasks. Each table showcase the mean and standard deviation over 10 seeds. For each metric, the best performing method is **bolded**, while the runner up underlined.

Table 6: Unigram-2D: mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	0.38±0.14	0.08±0.14	3.09±1.23	0.22±0.28	1.82±0.68
POCCO (C)	0.00±0.00	0.82±0.01	10.02±0.11	0.99±0.01	0.00±0.00
MOReinforce (L)	0.02±0.00	0.64±0.02	8.41±0.21	0.82±0.02	0.83±0.34
MOReinforce (C)	0.02±0.00	0.64±0.02	8.43±0.20	0.83±0.02	0.79±0.34
MOURex (L)	0.03±0.00	0.60±0.02	8.00±0.17	0.79±0.02	1.04±0.06
MOUrex (C)	0.01±0.01	0.70±0.05	8.94±0.41	0.91±0.04	0.51±0.38
CAPQL	0.01±0.02	0.68±0.08	8.83±0.66	0.92±0.08	0.36±0.40
MOGFN	0.36±0.04	0.08±0.02	**2.88±0.08**	**0.12±0.03**	**2.26±0.08**
OP-GFN	**0.46±0.03**	**0.05±0.04**	3.11±0.50	0.17±0.12	2.02±0.26
DPaO (ours)	0.16±0.20	0.35±0.19	5.64±1.40	0.76±0.30	0.67±0.76

Table 7: Unigram-3D: mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	0.08±0.01	**0.06±0.01**	**3.12±0.04**	**0.17±0.03**	3.34±0.14
POCCO (C)	0.00±0.00	0.71±0.01	9.43±0.08	0.99±0.01	0.68±0.40
MOReinforce (L)	0.00±0.00	0.54±0.01	8.03±0.12	0.84±0.02	1.22±0.29
MOReinforce (C)	0.00±0.00	0.54±0.01	8.06±0.13	0.84±0.02	1.16±0.29
MOURex (L)	0.00±0.00	0.52±0.01	7.81±0.14	0.81±0.02	1.50±0.23
MOUrex (C)	0.00±0.00	0.60±0.07	8.42±0.77	0.92±0.02	1.11±0.48
CAPQL	0.00±0.00	0.58±0.07	8.31±0.61	0.93±0.07	0.97±0.68
MOGFN	**0.09±0.01**	0.09±0.01	3.59±0.28	0.23±0.09	**3.63±0.17**
OP-GFN	0.08±0.01	0.17±0.02	5.05±0.15	0.54±0.04	2.67±0.10
DPaO (ours)	0.07±0.04	0.22±0.14	5.36±0.95	0.63±0.20	2.23±0.86

Table 8: Unigram-4D: mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	0.01±0.00	**0.07±0.00**	**3.30±0.03**	**0.17±0.02**	**4.42±0.08**
POCCO (C)	0.00±0.00	0.65±0.00	9.16±0.04	0.99±0.01	0.82±0.22
MOReinforce (L)	0.00±0.00	0.47±0.01	7.83±0.09	0.84±0.02	2.02±0.15
MOReinforce (C)	0.00±0.00	0.48±0.01	7.83±0.10	0.84±0.01	1.98±0.08
MOURex (L)	0.00±0.00	0.47±0.01	7.77±0.11	0.83±0.02	2.07±0.16
MOUrex (C)	0.00±0.00	0.50±0.01	7.99±0.14	0.87±0.02	1.89±0.17
CAPQL	0.00±0.00	0.50±0.06	8.03±0.46	0.94±0.07	1.51±0.72
MOGFN	0.01±0.00	0.12±0.02	4.20±0.49	0.40±0.17	4.35±0.21
OP-GFN	0.01±0.00	0.26±0.02	6.20±0.17	0.69±0.05	3.07±0.17
DPaO (ours)	**0.02±0.00**	0.18±0.02	5.55±0.11	0.64±0.06	3.24±0.09

Table 9: Bigram-2D: mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	**0.45±0.16**	**0.06±0.16**	**2.96±1.34**	**0.12±0.31**	**2.10±0.74**
POCCO (C)	0.00±0.00	0.83±0.01	10.09±0.08	1.00±0.00	0.00±0.00
MOReinforce (L)	0.00±0.00	0.73±0.02	9.21±0.22	0.92±0.03	0.64±0.37
MOReinforce (C)	0.00±0.00	0.73±0.02	9.21±0.22	0.92±0.03	0.64±0.37
MOURex (L)	0.00±0.00	0.73±0.02	9.21±0.22	0.92±0.03	0.68±0.40
MOUrex (C)	0.00±0.00	0.76±0.02	9.44±0.22	0.95±0.02	0.35±0.37
CAPQL	0.00±0.01	0.78±0.06	9.66±0.58	0.96±0.06	0.25±0.42
MOGFN	0.19±0.04	0.28±0.06	5.08±0.64	0.49±0.14	1.59±0.33
OP-GFN	0.17±0.07	0.35±0.08	6.00±0.61	0.65±0.14	0.99±0.39
DPaO (ours)	0.38±0.16	0.18±0.13	4.31±1.03	0.48±0.25	0.96±0.41

Table 10: Bigram-3D: mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	0.06±0.12	0.27±0.14	4.73±1.13	0.81±0.39	0.40±0.84
POCCO (C)	0.00±0.00	0.73±0.01	9.59±0.05	1.00±0.00	0.00±0.00
MOReinforce (L)	0.00±0.00	0.64±0.01	8.80±0.10	0.94±0.00	0.74±0.34
MOReinforce (C)	0.00±0.00	0.64±0.01	8.80±0.10	0.94±0.00	0.75±0.33
MOURex (L)	0.00±0.00	0.64±0.01	8.80±0.10	0.94±0.00	0.74±0.34
MOUrex (C)	0.00±0.00	0.68±0.01	9.13±0.11	0.95±0.02	0.80±0.45
CAPQL	0.00±0.00	0.68±0.06	9.15±0.52	0.96±0.05	0.41±0.46
MOGFN	0.06±0.03	0.24±0.06	5.24±0.41	0.78±0.16	0.94±0.70
OP-GFN	0.07±0.03	0.23±0.12	5.52±0.90	0.61±0.15	**1.73±0.66**
DPaO (ours)	**0.24±0.05**	**0.05±0.05**	**3.49±0.50**	**0.38±0.18**	1.66±0.62

Table 11: Bigram-4D: mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	0.00±0.00	0.42±0.00	6.44±0.01	1.00±0.00	0.00±0.00
POCCO (C)	0.00±0.00	0.67±0.01	9.32±0.08	1.00±0.00	0.00±0.00
MOReinforce (L)	0.00±0.00	0.58±0.01	8.61±0.07	0.94±0.00	1.63±0.39
MOReinforce (C)	0.00±0.00	0.58±0.01	8.62±0.09	0.94±0.00	1.55±0.50
MOURex (L)	0.00±0.00	0.58±0.01	8.60±0.10	0.94±0.00	1.61±0.39
MOUrex (C)	0.00±0.00	0.59±0.01	8.70±0.13	0.94±0.00	1.45±0.36
CAPQL	0.00±0.00	0.61±0.05	8.90±0.44	0.97±0.05	0.67±0.72
MOGFN	0.00±0.00	0.34±0.03	6.28±0.23	0.91±0.05	1.12±0.43
OP-GFN	0.00±0.00	0.33±0.09	6.67±0.67	0.78±0.09	2.24±0.80
DPaO (ours)	**0.04±0.01**	**0.10±0.03**	**4.31±0.20**	**0.62±0.09**	**2.70±0.44**

Table 12: NASBench201 (Imagenet): mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	**0.92±0.01**	0.03±0.01	**1.35±0.03**	**0.04±0.01**	**1.81±0.17**
POCCO (C)	0.41±0.17	0.41±0.14	2.84±0.37	0.50±0.18	0.83±0.21
MOReinforce (L)	0.89±0.02	0.05±0.01	1.49±0.07	0.09±0.02	1.51±0.21
MOReinforce (C)	0.88±0.02	0.07±0.02	1.55±0.07	0.11±0.02	1.48±0.19
MOURex (L)	0.91±0.01	0.04±0.01	1.44±0.06	0.08±0.02	1.58±0.13
MOUrex (C)	0.27±0.14	0.52±0.13	3.12±0.31	0.67±0.16	0.76±0.18
CAPQL	0.89±0.01	0.06±0.01	1.50±0.06	0.11±0.02	1.52±0.17
MOGFN	0.91±0.01	**0.03±0.00**	1.39±0.04	0.05±0.01	1.72±0.14
OP-GFN	0.90±0.03	0.04±0.02	1.48±0.14	0.08±0.03	1.53±0.18
DPaO (ours)	0.91±0.01	0.03±0.01	1.39±0.03	0.05±0.02	1.65±0.21

Table 13: DARTS (Error, No. of Parameters): mean ± std over 10 seeds

Method	HV (⇑)	R2 (⇓)
POCCO (L)	**0.96±0.00**	**1.19±0.01**
POCCO (C)	0.00±0.00	6.88±0.04
MOReinforce (L)	0.81±0.02	1.90±0.09
MOReinforce (C)	0.79±0.02	1.99±0.14
MOURex (L)	0.92±0.01	1.34±0.04
MOUrex (C)	0.68±0.16	2.63±0.94
CAPQL	0.81±0.02	1.89±0.12
MOGFN	0.92±0.03	1.34±0.11
OP-GFN	0.94±0.01	1.28±0.05
DPaO (ours)	0.94±0.01	1.25±0.03

Table 14: DARTS (Error, FLOPs): mean ± std over 10 seeds

Method	HV (⇑)	R2 (⇓)
POCCO (L)	**0.96±0.00**	**1.16±0.01**
POCCO (C)	0.00±0.00	6.87±0.04
MOReinforce (L)	0.83±0.02	1.77±0.10
MOReinforce (C)	0.81±0.02	1.90±0.13
MOURex (L)	0.91±0.02	1.38±0.07
MOUrex (C)	0.72±0.16	2.42±0.91
CAPQL	0.83±0.02	1.75±0.11
MOGFN	0.93±0.02	1.32±0.11
OP-GFN	0.92±0.02	1.35±0.09
DPaO (ours)	0.91±0.01	1.41±0.06

Table 15: DARTS (Error, No. of Parameters, FLOPs): mean ± std over 10 seeds

Method	HV (⇑)	R2 (⇓)
POCCO (L)	0.88±0.02	1.28±0.07
POCCO (C)	0.00±0.00	7.85±0.04
MOReinforce (L)	0.70±0.03	1.87±0.12
MOReinforce (C)	0.66±0.04	2.05±0.15
MOURex (L)	0.89±0.03	1.25±0.09
MOUrex (C)	0.79±0.11	1.57±0.38
CAPQL	0.70±0.05	1.86±0.18
MOGFN	**0.91±0.02**	**1.20±0.05**
OP-GFN	0.90±0.03	1.20±0.07
DPaO (ours)	0.90±0.01	1.21±0.04

Table 16: DNA (Energy, inverse of length): mean ± std over 10 seeds

Method	HV (⇑)	R2 (⇓)
POCCO (L)	1.89±0.37	1.53±0.56
POCCO (C)	0.00±0.01	7.66±0.36
MOReinforce (L)	0.50±0.05	3.50±0.48
MOReinforce (C)	0.46±0.04	3.68±0.43
MOURex (L)	1.31±0.38	**1.50±0.41**
MOUrex (C)	0.25±0.17	5.22±1.13
CAPQL	0.43±0.05	4.07±0.33
MOGFN	1.37±0.22	2.43±1.41
OP-GFN	1.53±0.49	1.57±0.42
DPaO (ours)	**2.27±0.30**	1.84±0.83

Table 17: DNA (Energy, No. of base Pairs): mean ± std over 10 seeds

Method	HV (⇑)	R2 (⇓)
POCCO (L)	**0.24±0.01**	5.46±0.10
POCCO (C)	0.00±0.00	10.05±0.20
MOReinforce (L)	0.07±0.02	7.12±0.21
MOReinforce (C)	0.07±0.02	7.13±0.22
MOURex (L)	0.19±0.02	5.88±0.18
MOUrex (C)	0.07±0.04	7.33±0.62
CAPQL	0.07±0.01	7.13±0.17
MOGFN	0.21±0.02	5.67±0.18
OP-GFN	0.20±0.06	5.75±0.54
DPaO (ours)	**0.24±0.01**	**5.44±0.10**

Table 18: DNA (Energy, No. of hairpins, No. of base Pairs): mean ± std over 10 seeds

Method	HV (⇑)	R2 (⇓)
POCCO (L)	0.04±0.01	6.21±0.23
POCCO (C)	0.00±0.00	9.60±0.00
MOReinforce (L)	0.02±0.00	7.02±0.18
MOReinforce (C)	0.02±0.00	7.01±0.18
MOURex (L)	0.04±0.00	6.31±0.08
MOUrex (C)	0.01±0.00	7.61±0.38
CAPQL	0.02±0.00	6.95±0.18
MOGFN	**0.05±0.00**	6.22±0.11
OP-GFN	0.04±0.01	6.34±0.34
DPaO (ours)	0.05±0.01	**6.11±0.20**

Table 19: Small molecule (log P, QED): mean ± std over 10 seeds

Method	HV (⇑)	IGD+ (⇓)	R2 (⇓)	ϵ-metric (⇓)	PC entropy (⇑)
POCCO (L)	0.65±0.59	0.42±0.12	11.82±15.25	0.47±0.13	0.34±0.58
POCCO (C)	0.00±0.00	14.61±0.58	91.60±3.51	14.84±0.58	0.00±0.00
MOReinforce (L)	0.40±0.04	0.33±0.06	4.12±0.29	0.36±0.04	**1.18±0.35**
MOReinforce (C)	0.40±0.04	0.34±0.06	4.15±0.29	0.37±0.04	1.16±0.23
MOURex (L)	0.43±0.05	0.31±0.06	3.95±0.34	0.34±0.05	1.16±0.21
MOUrex (C)	0.40±0.04	0.34±0.05	4.19±0.36	0.36±0.05	0.98±0.44
CAPQL	0.37±0.09	0.37±0.09	4.40±0.72	0.38±0.08	1.00±0.36
MOGFN	0.46±0.04	**0.28±0.06**	**3.73±0.28**	**0.32±0.03**	1.16±0.35
OP-GFN	0.53±0.11	0.33±0.07	3.80±0.73	0.39±0.07	1.11±0.62
DPaO (ours)	**1.19±0.74**	0.41±0.14	11.21±15.58	0.47±0.15	0.80±0.64