Goal-conditioned GFlowNets for Controllable Multi-Objective Molecular Design

Julien Roy [‡ 1 2] **Pierre-Luc Bacon** [1 3 4] **Christopher Pal** [1 2 3 5 6] **Emmanuel Bengio** [7]

Abstract

In recent years, *in-silico* molecular design has received much attention from the machine learning community. When designing a new compound for pharmaceutical applications, there are usually multiple properties of such molecules that need to be optimised: binding energy to the target, synthesizability, toxicity, EC50, and so on. While previous approaches have employed a scalarization scheme to turn the multi-objective problem into a *preference-conditioned* single objective, it has been established that this kind of reduction may produce solutions that tend to slide towards the extreme points of the objective space when presented with a problem that exhibits a concave Pareto front. In this work we experiment with an alternative formulation of *goal-conditioned* molecular generation to obtain a more controllable conditional model that can uniformly explore solutions along the entire Pareto front.

1. Introduction

Modern Multi-Objective optimisation (MOO) is comprised of a large number of paradigms (Keeney et al., 1993; Miettinen, 2012) intended to solve the problem of trading off between different objectives; a setting particularly relevant to molecular design (Jin et al., 2020; Jain et al., 2022b). One particular paradigm that integrates well with recent discrete deep-learning based MOO is *scalarization* (Ehrgott, 2005; Pardalos et al., 2017), which transforms the problem of discovering the Pareto front of a problem into a *family* of problems, each defined by a set of coefficients over the objectives. One notable issue with such approaches is that the solution they give tends to depend on the *shape* of the Pareto front in objective space (Emmerich & Deutz, 2018).

To tackle this problem, we propose to train models which explicitly target specific regions in *objective space*. Taking inspiration from goal-conditional reinforcement learning (Schaul et al., 2015), we condition GFlowNet (Bengio et al., 2021a;b) models on a description of such *goal regions*. Through the choice of distribution over these goals, we enable users of these models to have more fine-grained control over trade-offs. We also find that assuming proper coverage of the goal distribution, goal-conditioned models discover a more complete and higher entropy approximation of the Pareto front for various shapes.

2. Background & Related Work

The **Multi-Objective optimisation** problem can be broadly described as the desire to maximize a set of K objectives over $\mathcal{X}$, $\mathbf{R}(x) \in \mathbb{R}^K$. In typical MOO problems, there is no single optimal solution x such that $R_k(x) > R_k(x') \,\forall\, k, x'$. Instead, the solution set is generally composed of *Pareto optimal* points, which are points x that are not *dominated* by any other point, i.e. $\nexists\, x'$ s.t. $R_k(x) \geq R_k(x') \,\forall\, k$. In other words, a point is Pareto optimal if it cannot be locally improved. The projection in objective space of the set of Pareto optimal points forms the so-called *Pareto front*.

As graph-based models improve (Rampášek et al., 2022) and more molecular data become available (Wu et al., 2018), molecular design has become an active field of research within the deep learning community (Brown et al., 2019; Huang et al., 2021), and core to this research is the fact that molecular design is a fundamentally multi-objective search problem (Papadopoulos & Linke, 2006; Brown et al., 2006). The advent of such tools has led to various important work at the intersection of these two fields (Zhou et al., 2019; Ståhl et al., 2019; Jin et al., 2020; Jain et al., 2022b).

The **Generative Flow Network** (GFlowNet, GFN) framework is a recently introduced method to train energy-based generative models (i.e. models that learn $p_\theta(x) \propto R(x)$; Bengio et al., 2021a). They have now been successfully applied to a variety of settings such as biological sequences (Jain et al., 2022a), causal discovery (Deleu et al., 2022; Atanackovic et al., 2023), discrete latent variable modeling (Hu et al., 2023), and computational graph scheduling (Zhang et al., 2023). The framework itself has also received theoretical attention (Bengio et al., 2021b), for

[‡]Work conducted while interning at Recursion Pharmaceuticals. [1]Mila. [2]École Polytechnique de Montréal. [3]Université de Montréal. [4]Facebook CIFAR AI Chair. [5]ServiceNow. [6]Canada CIFAR AI Chair. [7]Recursion Pharmaceuticals. Correspondence to: Julien Roy <julien.roy@mila.quebec>.

Workshop on Challenges in Deployable Generative AI at International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

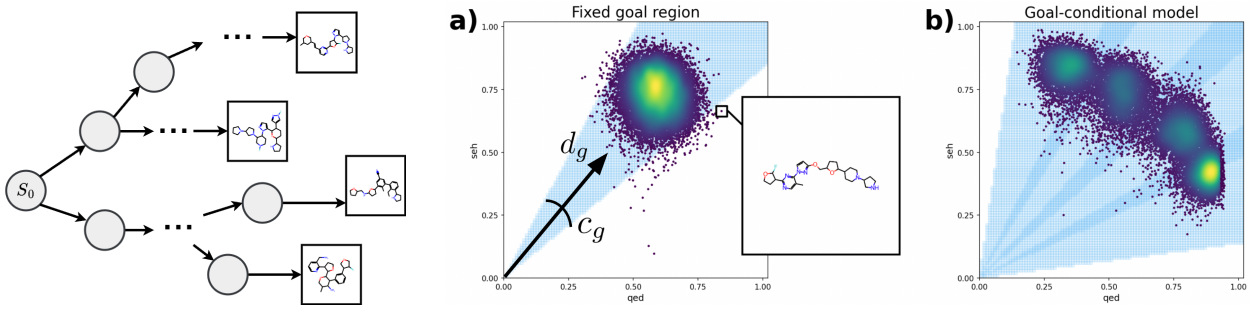

Figure 1. The diagram on the left depicts the state space of a GFlowNet molecule generator which learns a forward policy that sequentially builds diverse molecules. **a)** The sampling distribution learned by such a model on a two-objective problem (seh, qed). Each dot represents a molecule's image in the objective space. The focus region (see Section 3.1) is depicted as a light blue cone, and the colors represent the density of the distribution. The model learns to produce molecules that mostly belong within the focus region. **b)** By training a goal-conditioned GFlowNet and sampling from several focus regions (here showing 4 distinct regions), we can cover a wider section of the objective space and increase the diversity of proposed candidates.

example, highlighting its connections to variational methods (Zhang et al., 2022; Malkin et al., 2022b), and several objectives to train GFNs have been proposed (Malkin et al., 2022a; Madan et al., 2022; Pan et al., 2023) including extensions to continuous domains (Lahlou et al., 2023).

In the context of molecular design, GFlowNets have several important properties that make them an interesting method for this task. Notably, they are naturally well-suited for discrete compositional object generation, and their multi-modal modeling capabilities allow them to induce greater state space diversity in the solutions they find than previous methods. A recent GFN-based approach to multi-objective molecular design, which we call *preference-conditioning* (Jain et al., 2022b), amounts to scalarizing the objective function by using a set of weights (or preferences) w:

$$R_w(x) = \sum_k w_k r_k \quad , \quad \sum_k w_k = 1 \quad , \quad w_k \geq 0 \quad (1)$$

and then passing this preference vector w as input to the model. By sampling various w's from a distribution such as Dirichlet's during training, one can obtain a model that can be conditioned to emphasize some preferred dimensions of the reward function. Jain et al. (2022b) also find that such a method finds diverse candidates in both state and objective spaces.

3. Methods

3.1. Goal-conditioned GFlowNets

Building on the method of Jain et al. (2022b), our approach also formulates the problem as a conditional generative task but now imposes a hard constraint on the model: the goal is to generate samples for which the image in objective space falls into the specified goal region. While many different goal-design strategies could be employed, we take inspiration from Lin et al. (2019) and state that a sample x meets the specified goal g if the cosine similarity between its reward vector r and the goal direction d_g is above the threshold c_g: $g := \{r \in \mathbb{R}^K : \frac{r \cdot d_g}{||r|| \cdot ||d_g||} \geq c_g\}$. We call such a goal a *focus region*, which represents a particular choice of trade-off in the objective space (see Figure 1). The method can be considered a form of goal-conditional reinforcement learning (Schaul et al., 2015), where the reward function R_g depends on the current goal g. In our case we have:

$$R_g(x) = \begin{cases} \sum_k r_k, & \text{if } r \in g \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

To alleviate the effects of the now increased sparsity of the reward function R_g, we use a replay buffer which proved to stabilise the learning dynamics of our models (see Appendix C.1). Notably, by explicitly formulating a goal, we can measure the *goal-reaching accuracy* of our model, which refers to the proportion of samples that successfully landed in their prescribed region. This measurement enables us to employ hindsight experience replay (Andrychowicz et al., 2017), which lets the model learn from the sampled trajectories that didn't meet their goal. Finally, to further increase the goal-reaching accuracy we sharpen the reward function's profile to help the model generate samples closer to the center of the focus region (see Appendix C.2).

3.2. Learned Goal Distribution

Preference conditioning uses soft constraints to steer the model in some regions of the objective space. While hard constraints provide a more explicit way of incorporating the user's intentions in the model (Amodei et al., 2016; Roy et al., 2021), they come with the unique challenge that not every goal may be feasible. In such cases, the model will only observe samples with a reward of 0 and thus return molecules of little interest drawn uniformly across the state

space. These "bad samples" are not harmful in themselves and can easily be filtered out. Still, their prominence will affect the sampling efficiency of goal-conditioned approaches compared to their soft-constrained counterpart. Moreover, the number of infeasible regions will likely multiply as the number of objectives grows, further aggravating this disparity. To cope with this challenge, we propose to use a simple *tabular goal-sampler* (Tab-GS) which maintains a belief about whether any particular goal direction d_g is feasible. Once learned, we can start drawing new goals from it with a much lower likelihood on the goals that are believed to be infeasible, thus restoring most of the lost sample efficiency. We give more details on this approach in Appendix C.3 and use it in our experiments in Section 4.3.

3.3. Evaluation Metrics

While there exists many multi-objective scoring functions to choose from, any single metric only partially captures the desirable properties of the learned generative distribution (Audet et al., 2021). In this work, we focus on sampling high-performing molecules across the entire Pareto front in a controllable manner at test time. With that in mind, we propose combining three metrics to evaluate our solution. The first one, the Inverted Generational Distance (IGD) (Coello & Cortés, 2005), uses a set of reference points P (the *true* Pareto front) and takes the average of the distance to the closest generated sample for each of these points: $\text{IGD}(S, P) := \frac{1}{|P|}\sum_{p\in P}\min_{s\in S}||s-p||_2^2$ where $S = \{s_i\}_{i=1}^N$ is the image in objective space of a set of N generated molecules s_i. When the true Pareto front is unknown, we use a discretization of the extreme faces of the objective space hypercube as reference points. IGD thus captures the width and depth at which our Pareto front approximation reaches out in the objective space. The second metric, which we call the Pareto-Clusters Entropy (PC-ent), measures how uniformly distributed the samples are along the true Pareto front. To accomplish this, we use the same reference points P as for IGD, and cluster together in the subset S_j all of the samples s_i located closer to the reference point p_j than any other reference point. PC-ent computes the entropy of the histogram of each counts $|S_j|$, reaching its maximum value of $-\log\frac{1}{|P|}$ when all the samples are uniformly distributed relative to the true Pareto front: $\text{PC-ent}(S, P) := -\sum_j \frac{|S_j|}{|P|}\log\frac{|S_j|}{|P|}$. Finally, to report on the *controllability* of the compared methods, we measure the Pearson correlation coefficient (PCC) between the conditional vector c (goal or preference) and the resulting reward vector s, averaged across objectives k: $\text{Avg-PCC}(S, C) := \frac{1}{K}\sum_{k=1}^{K}\text{PCC}(s_{\cdot,k}, c_{\cdot,k})$.

4. Results

4.1. Evaluation Tasks

We primarily experiment on a two-objective task, the well-known drug-likeness heuristic QED (Bickerton et al., 2012), which is already between 0 and 1, and the sEH binding energy prediction of a pre-trained publicly available model (Bengio et al., 2021a); we divide the output of this model by 8 to ensure it will likely fall between 0 and 1 (some training data goes past values of 8). For 3 and 4 objective tasks, we use a standard heuristic of synthetic accessibility (Ertl & Schuffenhauer, 2009) and a penalty for compounds exceeding a molecular weight of 300. See Appendix A for all task and training details.

4.2. Comparisons in Difficult Objective Landscapes

To simulate the effect of complexifying the objective landscape while keeping every other parameter of the evaluation fixed, we incorporate *unreachable regions*, depicted in dark in Figures 2 & 3, by simply setting to *null* the reward function of any molecule whose image in the objective space would fall into these dark regions. We can see that the preference-conditioned approach can effectively solve problems exhibiting a convex pareto-front (Figure 2 & 3, columns 1-2). However, it is far less effective on problems exhibiting more complex objective landscapes. When faced with a concave Pareto front, the algorithm favours solutions towards the extreme ends (Figure 2 & 3, columns 3-7). In contrast, by explicitly forcing the algorithm to sample from each trade-off direction in the objective space, our goal-conditioned method learns a sampling distribution that spans the entire space diagonally, no matter how complex we make the objective landscape. Table 1 reports the performance of both methods on these objective landscapes in terms of IGD, Avg-PCC and PC-ent (mean $\pm$ sem, over 3 seeds). We see in Table 1 that according to IGD, preference-conditioning and goal-conditioning perform similarly in terms of pushing the empirical Pareto front forward. While the two learned distributions are in many cases very different (Figure 2, columns 3-7), the preference-conditioning method still manages to produce a few samples in the middle areas of the Pareto front, which satisfies IGD as it only looks for the single closest sample to each reference point. However, the two algorithms differ drastically in terms of controllability of the distribution (color-coded in Figure 2) and uniformity of the distribution along the Pareto front (color-coded in Figure 3), which are highlighted by the Avg-PCC and PC-ent criteria in Table 1.

4.3. Comparisons for Increasing Number of Objectives

Using the same metrics, we also evaluate the performance of both methods when the number of objectives increases.

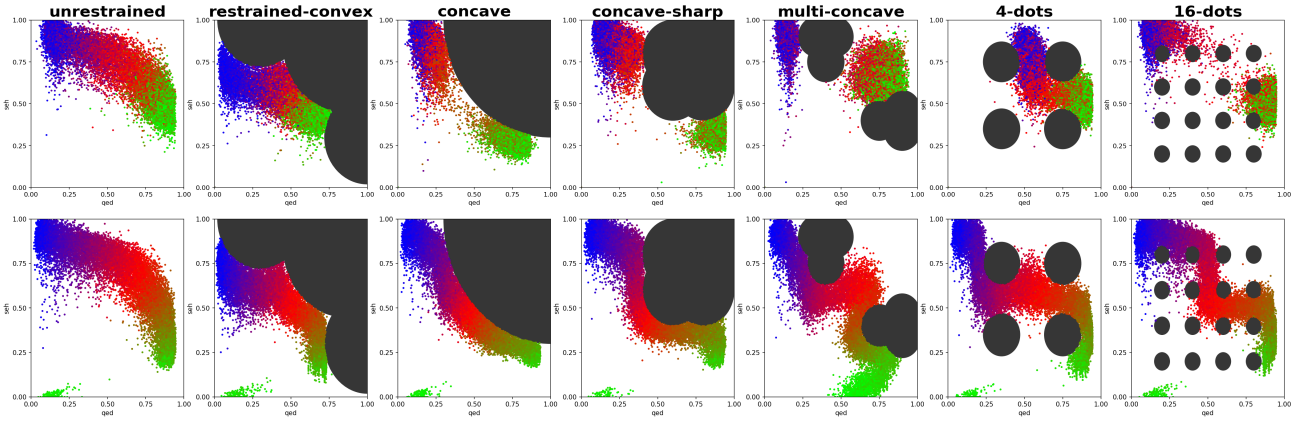

Figure 2. Comparisons between a preference-conditioned GFN (top row) and a goal-conditioned GFN (bottom row) on a set of increasingly complex modifications of a two-objective (seh, qed) fragment-based molecule generation task (Jain et al., 2022b). The BRG colors represent the angle between the vector $[1, 0]$ and either the preference-vector w (top) or the goal direction d_g (bottom), respectively. For example, in the case of preference-conditioning, a green dot means that such samples were produced with a strong preference for the qed-objective, while in the goal-conditioning case, a green dot means that the model *intended* to produce a sample alongside the qed-axis. We see that goal-conditioning allows to span the entire objective space even in very challenging landscapes (columns 3-7) and in a more controllable way.

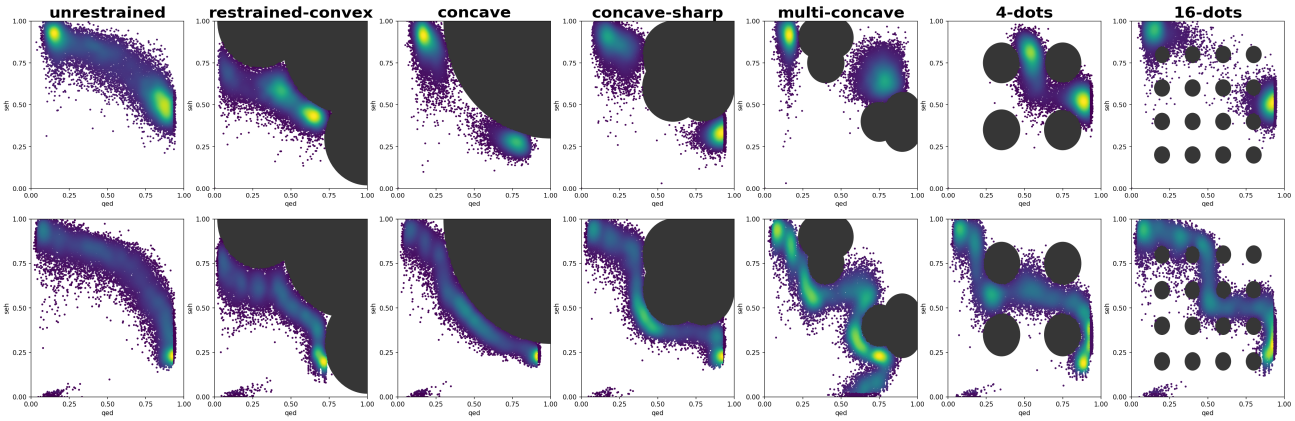

Figure 3. Comparisons of the same sampling distributions depicted in Figure 2. Now the colors indicate how densely populated a particular area of the objective space is (brighter is more populated). We can see that by explicitly targeting different trade-off regions in objective space, our goal-conditioning approach (bottom row) produces far more evenly distributed samples along the Pareto front than with preference-conditioning (top row).

Table 1. Comparisons according to IGD, Avg-PCC and PC-ent between preference-conditioned and goal-conditioned GFNs on a set of increasingly difficult objective landscapes, metrics reported on 3 seeds (mean ± sem).

	algorithm	**unrestrained**	**restrained-convex**	**concave**	**concave-sharp**	**multi-concave**	**4-dots**	**16-dots**
IGD (↓)	pref-cond	$\mathbf{0.087 \pm 0.001}$	0.316 ± 0.002	0.272 ± 0.001	$\mathbf{0.180 \pm 0.002}$	$\mathbf{0.152 \pm 0.006}$	0.130 ± 0.011	0.109 ± 0.009
	goal-cond	0.095 ± 0.002	$\mathbf{0.310 \pm 0.001}$	$\mathbf{0.266 \pm 0.001}$	0.197 ± 0.002	0.173 ± 0.004	0.134 ± 0.002	0.115 ± 0.004
Avg-PCC (↑)	pref-cond	0.905 ± 0.001	0.673 ± 0.009	0.830 ± 0.002	0.855 ± 0.004	0.700 ± 0.009	0.768 ± 0.038	0.770 ± 0.011
	goal-cond	$\mathbf{0.967 \pm 0.002}$	$\mathbf{0.953 \pm 0.001}$	$\mathbf{0.926 \pm 0.002}$	$\mathbf{0.915 \pm 0.001}$	$\mathbf{0.946 \pm 0.004}$	$\mathbf{0.928 \pm 0.002}$	$\mathbf{0.948 \pm 0.001}$
PC-ent (↑)	pref-cond	2.170 ± 0.004	1.913 ± 0.019	1.563 ± 0.009	1.629 ± 0.002	1.867 ± 0.015	1.521 ± 0.022	1.610 ± 0.019
	goal-cond	$\mathbf{2.472 \pm 0.006}$	$\mathbf{2.242 \pm 0.013}$	$\mathbf{1.997 \pm 0.002}$	$\mathbf{1.918 \pm 0.001}$	$\mathbf{2.380 \pm 0.020}$	$\mathbf{2.270 \pm 0.025}$	$\mathbf{2.262 \pm 0.014}$

As described in Section 3.2, to maintain the sample efficiency of our goal-conditioned approach we sample the goal directions d_g from a learned tabular goal-sampler (Tab-GS) rather than uniformly across the objective space (Uniform-GS). We can see in Table 2 (and Appendix C.3) that with this adaptation, our goal-conditioned approach maintains its advantages in terms of controllability and uniformity of the learned distribution as the number of objectives increases, proving to be an effective method for probing large, high-dimensional objective spaces for diverse solutions.

Table 2. Comparisons according to IGD, Avg-PCC and PC-ent between preference- and goal-conditioned GFNs faced with increasing objectives (3 seeds, mean $\pm$ sem).

	algorithm	2 objectives	3 objectives	4 objectives
IGD (↓)	pref-cond	$\mathbf{0.088 \pm 0.001}$	0.218 ± 0.003	0.370 ± 0.000
	goal-cond	0.094 ± 0.004	$\mathbf{0.199 \pm 0.002}$	$\mathbf{0.303 \pm 0.001}$
Avg-PCC (↑)	pref-cond	0.904 ± 0.002	0.775 ± 0.004	0.612 ± 0.002
	goal-cond	$\mathbf{0.961 \pm 0.001}$	$\mathbf{0.909 \pm 0.001}$	$\mathbf{0.893 \pm 0.002}$
PC-ent (↑)	pref-cond	2.166 ± 0.007	3.775 ± 0.016	4.734 ± 0.004
	goal-cond	$\mathbf{2.471 \pm 0.001}$	$\mathbf{4.571 \pm 0.008}$	$\mathbf{6.320 \pm 0.009}$

5. Future Work

In this work, we proposed goal-conditioned GFlowNets for multi-objective molecular design. We showed that they are an effective solution to give practitioners more control over their generative models, allowing them to obtain a large set of more widely and more uniformly distributed molecules across the objective space. An important limitation of the proposed approach was the reduced sample efficiency of the method due to the existence of *a priori unknown* infeasible goals. We proposed a tabular approach to gradually discredit these fruitless goal regions as we explore the objective space. However, this set of parameters, one for every goal direction, grows exponentially with the number of objectives K, eventually leading to statistical and memory limitations. As future steps, we plan to experiment with a GFlowNet-based Goal Sampler (GFN-GS) which would learn to sample feasible goal directions dimension by dimension, thus benefiting from parameter sharing and the improved statistical efficiency of its hierarchical structure.

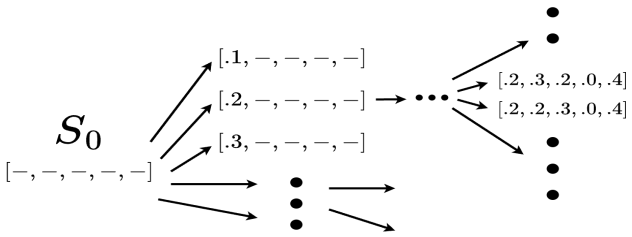

Figure 4. Depiction of a GFlowNet Goal Sampler (GFN-GS) gradually building goal directions d_g as a sequence of K steps.

Acknowledgements

We wish to thank Berton Earnshaw, Paul Barde and Tristan Deleu as well as the entire research team at Recursion's Emerging ML Lab for providing insightful comments on this work. We also acknowledge funding in support of this work from Fonds de Recherche Nature et Technologies (FRQNT), Institut de valorisation des données (IVADO) and Recursion Pharmaceuticals.

References

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Atanackovic, L., Tong, A., Hartford, J., Lee, L. J., Wang, B., and Bengio, Y. Dyngfn: Bayesian dynamic causal discovery using generative flow networks. *arXiv preprint arXiv:2302.04178*, 2023.

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., and Salomon, L. Performance indicators in multiobjective optimization. *European journal of operational research*, 292 (2):397–422, 2021.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021a.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. Gflownet foundations. *arXiv preprint arXiv:2111.09266*, 2021b.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Brown, N., McKay, B., and Gasteiger, J. A novel workflow for the inverse qspr problem using multiobjective optimization. *Journal of computer-aided molecular design*, 20:333–341, 2006.

Brown, N., Fiscato, M., Segler, M. H., and Vaucher, A. C. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

Coello, C. A. C. and Cortés, N. C. Solving multiobjective optimization problems using an artificial immune system. *Genetic programming and evolvable machines*, 6:163–190, 2005.

Deleu, T., Góis, A., Emezue, C., Rankawat, M., Lacoste-Julien, S., Bauer, S., and Bengio, Y. Bayesian structure learning with generative flow networks. In *Uncertainty in Artificial Intelligence*, pp. 518–528. PMLR, 2022.

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

Emmerich, M. T. and Deutz, A. H. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, 17:585–609, 2018.

Ertl, P. and Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Hu, E., Malkin, N., Jain, M., Everett, K., Graikos, A., and Bengio, Y. Gflownet-em for learning compositional latent variable models. *arXiv preprint arXiv:2302.06576*, 2023.

Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C. W., Xiao, C., Sun, J., and Zitnik, M. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548*, 2021.

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

Jain, M., Raparthy, S. C., Hernandez-Garcia, A., Rector-Brooks, J., Bengio, Y., Miret, S., and Bengio, E. Multiobjective gflownets. *arXiv preprint arXiv:2210.12765*, 2022b.

Jin, W., Barzilay, R., and Jaakkola, T. Multi-objective molecule generation using interpretable substructures. In *International conference on machine learning*, pp. 4849–4859. PMLR, 2020.

Keeney, R., Raiffa, H., L, K., and Meyer, R. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Wiley series in probability and mathematical statistics. Applied probability and statistics. Cambridge University Press, 1993. ISBN 9780521438834. URL https://books.google.ca/books?id=GPE6ZAqGrnoC.

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

Lahlou, S., Deleu, T., Lemos, P., Zhang, D., Volokhova, A., Hernández-García, A., Ezzine, L. N., Bengio, Y., and Malkin, N. A theory of continuous generative flow networks. *arXiv preprint arXiv:2301.12594*, 2023.

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. Pareto multi-task learning. *Advances in neural information processing systems*, 32, 2019.

Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A., Bosc, T., Bengio, Y., and Malkin, N. Learning gflownets from partial episodes for improved convergence and stability. *arXiv preprint arXiv:2209.12782*, 2022.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in gflownets. *arXiv preprint arXiv:2201.13259*, 2022a.

Malkin, N., Lahlou, S., Deleu, T., Ji, X., Hu, E., Everett, K., Zhang, D., and Bengio, Y. Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*, 2022b.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Pan, L., Malkin, N., Zhang, D., and Bengio, Y. Better training of gflownets with local credit and incomplete trajectories. *arXiv preprint arXiv:2302.01687*, 2023.

Papadopoulos, A. I. and Linke, P. Multiobjective molecular design for integrated process-solvent systems synthesis. *AIChE Journal*, 52(3):1057–1070, 2006.

Pardalos, P. M., Žilinskas, A., Žilinskas, J., et al. *Non-convex multi-objective optimization*. Springer, 2017.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Roy, J., Girgis, R., Romoff, J., Bacon, P.-L., and Pal, C. Direct behavior specification via constrained reinforcement learning. *arXiv preprint arXiv:2112.12228*, 2021.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320. PMLR, 2015.

Ståhl, N., Falkman, G., Karlsson, A., Mathiason, G., and Bostrom, J. Deep reinforcement learning for multiparameter optimization in de novo drug design. *Journal of*

chemical information and modeling, 59(7):3166–3176, 2019.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

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

Zhang, D., Chen, R. T., Malkin, N., and Bengio, Y. Unifying generative models with gflownets. *arXiv preprint arXiv:2209.02606*, 2022.

Zhang, D. W., Rainone, C., Peschl, M., and Bondesan, R. Robust scheduling with gflownets. *arXiv preprint arXiv:2302.05446*, 2023.

Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

A. Task and Training Details

We use the GFlowNet framework (Bengio et al., 2021a;b) to train discrete distribution samplers over the space of molecules that can be assembled from a set of pre-defined molecular fragments (Kumar et al., 2012). A state is represented as a graph in which each node represents a fragment from the fragment library and where each edge has two attributes representing the attachment point of each connected fragment to its neighbor. The state representation is augmented with a fully-connected virtual node, whose features are an embedding of the conditioning information computed from the conditioning vector that represents the preferences w and/or the goal direction d_g. To produce the state-conditional distribution over actions, the model processes the state using a graph transformer architecture (Yun et al., 2019) for a predefined number of message-passing steps (number of layers). Our GFlowNet sampler thus starts from the initial state s_0 representing an empty graph. It iteratively constructs a molecule by either adding a node or an edge to the current state s_t until it eventually selects the 'STOP' action.

To maintain some amount of exploration throughout training, at each construction step t, the model samples a random action with probability ϵ and otherwise samples from its forward transition distribution. The model is trained using the trajectory balance criterion (Malkin et al., 2022a) and thus is parameterised by a forward action distribution P_F and an estimation of the partition function $Z := \sum_x R(x)$. Forbidden actions are masked out from the forward transition distribution (for example, the action of adding an edge to the empty state). We use a uniform distribution for the backward policy P_B. To prevent the sampling distribution from changing too abruptly, we collect new trajectories from a sampling model $P_F(\cdot\,|\theta_{\text{sampling}})$ which uses a soft update with hyperparameter τ to track the learned GFN at update k: $\theta^{(k)}_{\text{sampling}} \leftarrow \tau \cdot \theta^{(k-1)}_{\text{sampling}} + (1-\tau)\cdot\theta^{(k)}$. This is akin to the target Q-functions and target policies used in actor-critic frameworks (Mnih et al., 2015; Fujimoto et al., 2018).

The hyperparameters used for training both methods are listed in Table 3.

Table 3. Hyperparameters used in our conditional-GFN training pipeline

Hyperparameters	**Values**	
	Goal-conditioned GFN	**Preference-conditioned GFN**
Batch size	64	64
GFN temperature parameter β	60	60
Number of training steps	40,000	40,000
Number of GNN layers	2	2
GNN node embedding size	256	256
Learning rate for GFN's P_F	10^{-4}	10^{-4}
Learning rate for GFN's Z-estimator	10^{-3}	10^{-3}
Sampling moving average τ	0.95	0.95
Random action probability ϵ	0.01	0.01
Focus region cosine similarity threshold c_g	0.98	-
Limit reward coefficient m_g	0.20	-
Replay buffer length	100,000	-
Number of replay buffer trajectory warmups	1,000	-
Hindsight ratio	0.30	-
Conditioning-vector sampling distribution	$d_g \sim \begin{cases} \text{Uniform-GS} & \text{(Sec 4.2)} \\ \text{Tab-GS} & \text{(Sec 4.3)} \end{cases}$	$w \sim Dirichlet(1)$

B. Failure Modes and Filtering

While using goal regions as hard constraints offers a more precise tool for controllable generation, it faces the additional challenge that not all goals may be feasible (or that reaching some goals may be much easier to learn than others). When a model is conditioned with an infeasible goal, all the samples that it will observe will have a reward $R(x) = 0$. The proper behavior, in that case, is to sample any possible molecule with equal weight, thus sampling uniformly across the entire molecular state space. Such molecules generally won't be of any interest and can be discarded. Thus, in our experiments, we filter out such *out-of-focus* samples (molecules falling outside the focus region) and evaluate the candidates that were inside their prescribed focus region. Figure 5 shows the conditional distributions learned by a single model trained on the 2-objective task. The picture on the last row, second column showcases such an occurrence of difficult focus region which results in many samples simply belonging to the uniform distribution over the state space.

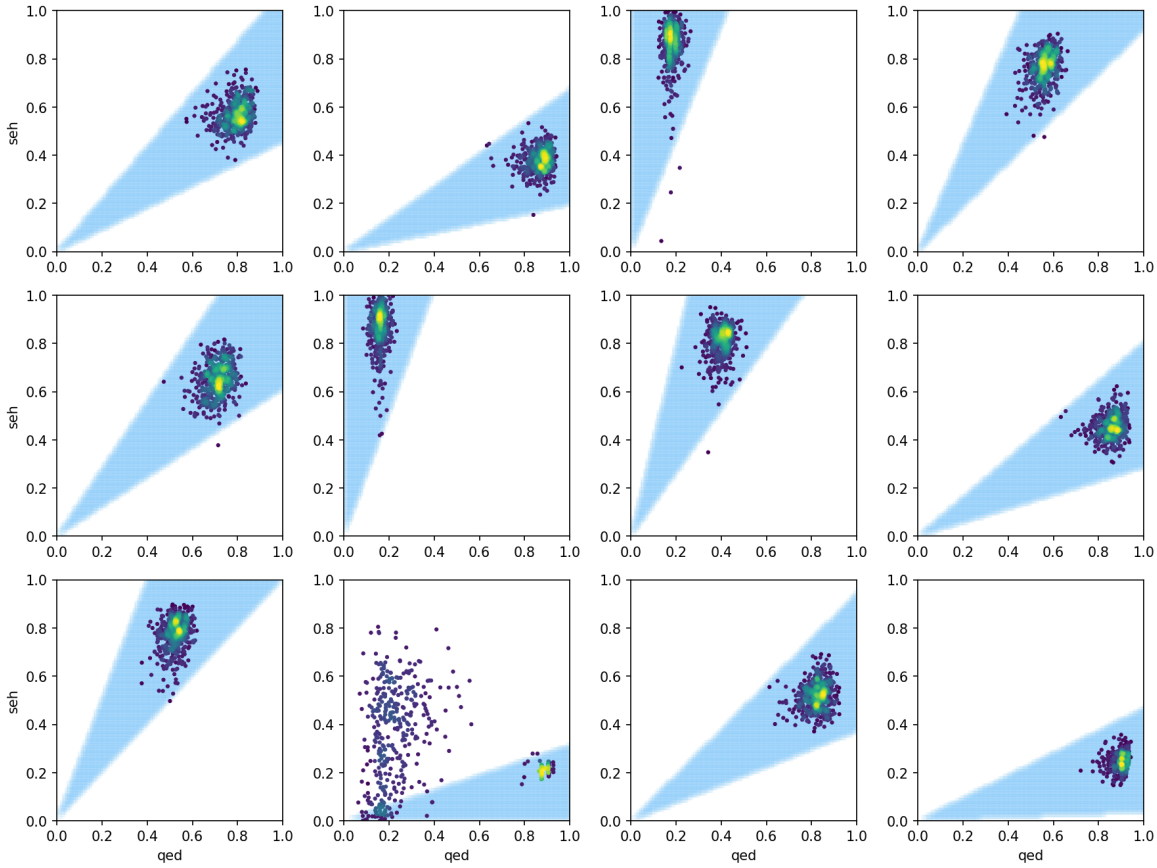

Figure 5. Learned conditional-distributions for different focus regions passed as input to the same model. Each dot marks the image of a generated molecule in the objective space. The colors indicate how densely populated a particular area of the objective space is (brighter is denser). The focus regions (goal regions) are depicted in light blue. The distribution on the last row, second column, showcases a focus region which seems difficult to reach and may not contain as large a population of molecules in the state space. In such cases, the model cannot learn to consistently produce samples from that goal region when conditioned on this goal direction d_g and will instead produce several samples very similar to the sampling distribution of an untrained model (uniform across the state space).

C. Ablations

C.1. Replay Buffer

While both the un-conditional and the preference-conditioned GFN models are learning stably even in a purely on-policy setting, we found that the goal-conditioned models were more prone to instabilities and mode-collapse when employed purely on-policy (see Figure 6). This could be because imposing these hard constraints on the generative behavior of the model drastically changes the reward landscape from one set of goals to another. While larger batches could potentially alleviate this problem, sampling uniformly from a replay buffer of the last trajectories proved effective, as observed in many works stemming from Mnih et al. (2015). As described in Section 3.1, we also use hindsight experience replay (Andrychowicz et al., 2017). Specifically, for every batch of data, we randomly select a subset of trajectories (hindsight-ratio * batch-size), among which we re-label both the goal direction d_g and the corresponding reward for the examples that didn't reach their goal.

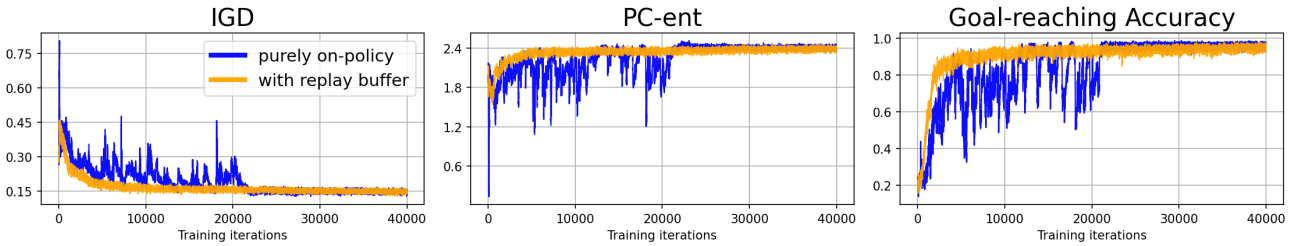

Figure 6. Learning curves for goal-conditioned models either trained purely on-policy (in blue) or using a replay buffer of past trajectories (in orange) on the 2-objective (seh, qed) task.

C.2. Limit Reward Coefficient

While the GFN model is given the goal direction d_g as input, the width of the goal region, which depends on the cosine-similarity threshold c_g is fixed, and the model adapts to producing samples within the region over time by trial-and-error. One can trade off the level of controllability of the goal-conditioned model with the difficulty of reaching those goals by increasing or reducing c_g. Another way to increase the controllability *and* goal-reaching accuracy without drastically affecting the difficulty of reaching such goals is to make the model preferentially generate samples near the center of the focus region, thus reducing the risk of producing an out-of-focus sample due to epistemic uncertainty. To do so, we modify Equation 2 and add a reward-coefficient α_g, which further modulates the magnitude of the scalar reward based on how close to the center of the focus region the sample was generated. While many shaping functions could be devised, we choose the following form:

$$R_g(x) = \begin{cases} \alpha_g \sum_k r_k, & \text{if } r \in g \\ 0, & \text{otherwise} \end{cases}, \qquad \alpha_g = \left(\frac{r \cdot d_g}{||r|| \cdot ||d_g||} \right)^{\frac{\log m_g}{\log c_g}} \tag{3}$$

In words, the reward coefficient α_g is equal to the cosine similarity between the reward vector r and the goal direction d_g exponentiated in such a way that $\alpha_g = m_g$ at the limit of the focus region. So for example, setting $m_g = 0.2$ means that the reward is maximal at the center of the focus region, is at 20% of that magnitude at the limit of the focus region, and follows a sharp sigmoid-like profile in between. Figure 7 showcases the reward coefficient as a function of the angle between r and d_g for different values of m_g and the corresponding distributions learned by the model. We can see that a smaller value of m_g encourages the model to produce samples in a more focused way towards the center of the goal region. Importantly, with a large enough value of m_g, this design preserves the notion of a well-defined goal region (positive reward inside the region and zero reward outside) and thus also preserves our ability to reason about goal-reaching accuracy, a beneficial concept for monitoring the model, filtering out-of-focus samples, etc.

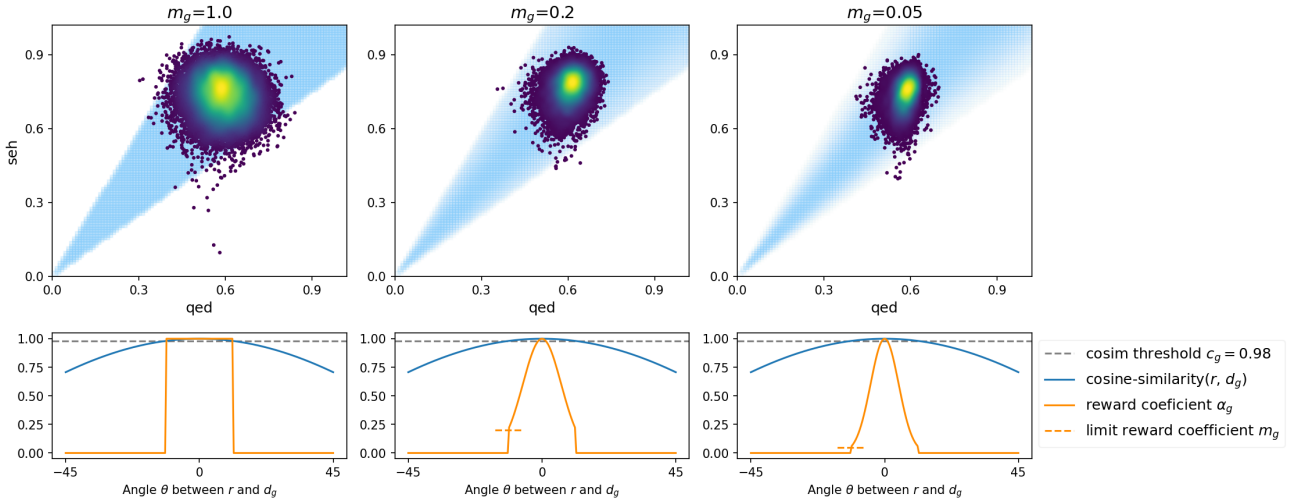

Figure 7. Effect of the hyperparameter m_g on the profile of the reward coefficient α_g and the learned sampling distribution (top row).

C.3. Tabular Goal-Sampler

To cope with the problem of infeasible goal regions described in Section 3.2, we explore the idea of sampling the goal directions d_g from a learned goal distribution rather than sampling all directions uniformly. The idea is that, as the model learns about which goal directions point towards infeasible regions of the objective space, we can attribute a much lower sampling likelihood to these regions in order to focus on more fruitful goals.

We implement a first version of this idea as a tabular goal-sampler (Tab-GS). We first build a dataset of goal directions $\mathcal{D}_G$. This could be done in many different ways such as sampling a large number of positive vectors at the surface of the unit hypersphere in objective space. In our case, we discretise the extreme faces of the unit hypercube and normalize them. At training time, for each direction vector $d_g \in \mathcal{D}_G$, we keep a count of the number of samples which have landed closest to it (closer than any other direction d'_g) and follow this very simple scheme: from the beginning up to 25% of the training iterations, we sample batches of goal directions $\{d_g\}_{i=1}^{N}$ uniformly over $\mathcal{D}_G$. Then starting at 25% of the training iterations, while we keep updating each direction's count, we sample batches of goal directions according to the following (unnormalized) likelihoods:

$$f(d_g) = \begin{cases} 1 & \text{if } d_g \text{ has never been sampled} \\ 1 & \text{if there has been a sample } r \text{ closest to } d_g \text{ than any other goal direction in } \mathcal{D}_G \\ 0.1 & \text{otherwise} \end{cases} \tag{4}$$

Finally, at 75% of the training, we stop updating the goal direction counts to allow the model to fine-tune itself to a now stationary goal-distribution Tab-GS(f). At test time we also sample from that same stationary distribution.

Figure 8 shows the effect of our learned tabular goal-sampler (Tab-GS) on the model's performance and learning dynamics. While the 2-objective problem does not contain a lot of infeasible goal directions, resulting in very similar behaviors for both methods, we can see that in the case of 3 and 4 objectives, the model experiences an important immediate improvement in goal-reaching accuracy at 25% of training when we start sampling d_g's according to our learned goal-sampler and that this improved focus helps the model further improves on these more fruitful goal directions, resulting in an increase IGD and PC-ent scores.

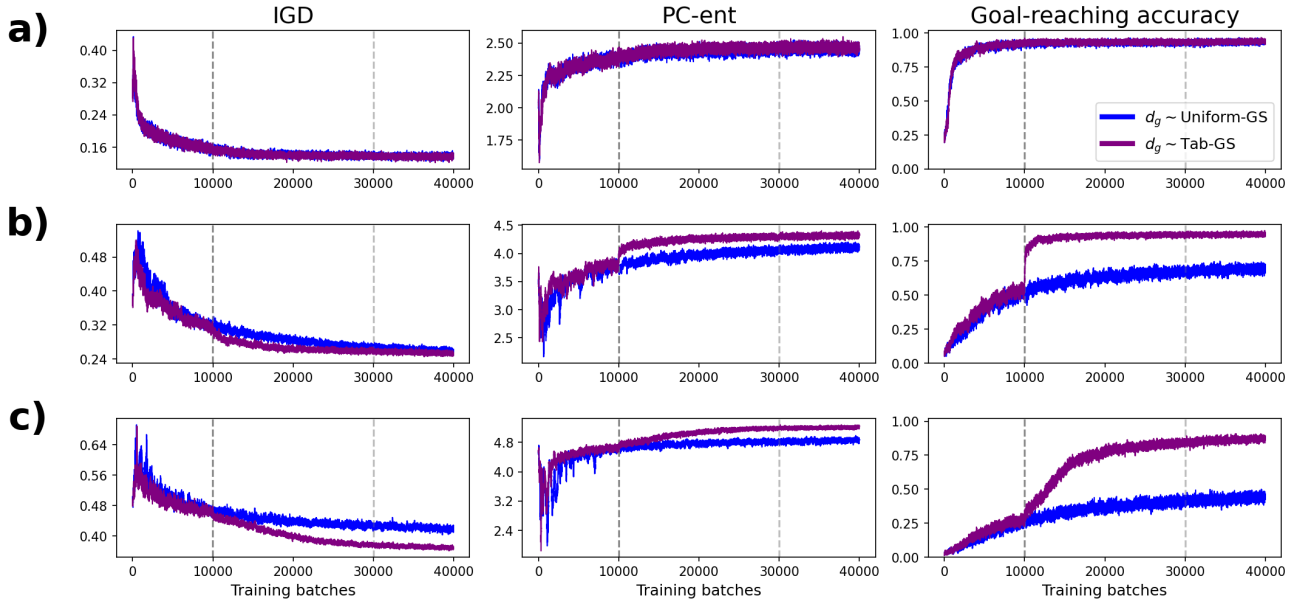

Figure 8. Learning curves for our goal-conditioned model trained by sampling goal directions d_g either uniformly on the positive quadrant of a K-dimensional hypersphere (Uniform-GS) in blue or according to our learned tabular goal-sampler (Tab-GS) in purple on **a)** 2 objectives, **b)** 3 objectives and **c)** 4 objectives. Vertical dotted lines indicate 25% and 75% of training when we start sampling goal directions according to Equation 4 and when we stop updating the learned goal-sampler, respectively.

D. Additional Results

In this section, we present additional plots for experiments on 2, 3 and 4 objectives (Figures 9, 10 & 11).

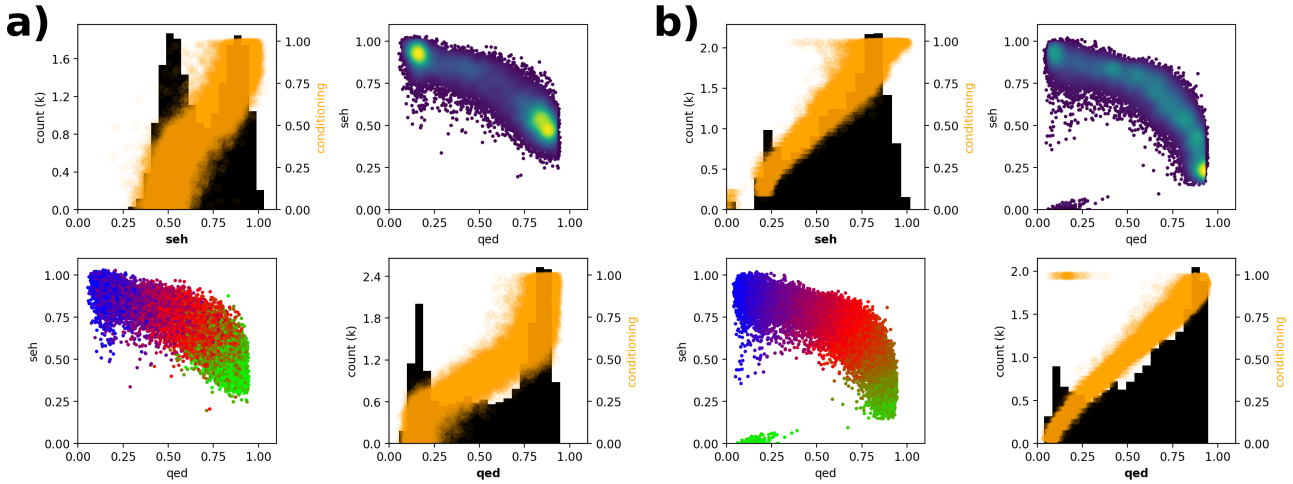

Figure 9. Comparison between **a)** preference-conditioned and **b)** goal-conditioned models trained on the 2-objective problem (seh, qed). Each panel is an assemblage of $K \times K$ plots where K is the number of objectives. **On the diagonal**, each plot focuses on a single objective. They each show a histogram (dark) of the samples' scores $r_{\cdot,k}$ for that objective, overlayed with a scatter plot (orange) in which each point is a distinct sample i with coordinates $(x, y) = (r_{i,k}, c_{i,k})$, where $r_{i,k}$ is the reward attributed to sample i for objective k and $c_{i,k}$ is the corresponding value of the conditioning vector that was used to generate that sample. The histogram thus showcases the distribution and span of our set of samples for a given dimension in objective space while the scatter plot allows us to visualise the correlation between the conditioning vectors and the resulting rewards for that dimension. **Above the diagonal**, each plot shows the density of the learned distribution on the plane corresponding to a pair of objectives. Brighter colors indicate that a region is more densely populated. **Below the diagonal**, each plot shows the controllability of the learned distribution where BRG colors represent the angle between the vector $[1, 0]$ and **a)** the preference-vector w or **b)** the goal direction d_g (this is the same as in Figure 2). Overall, we can see that on the density plot and on the histograms that the goal-conditioned approach produces a more uniformly distributed set of samples while the orange scatter plot and the BRG-colored plots show that they also provide a finer control over the generated samples.

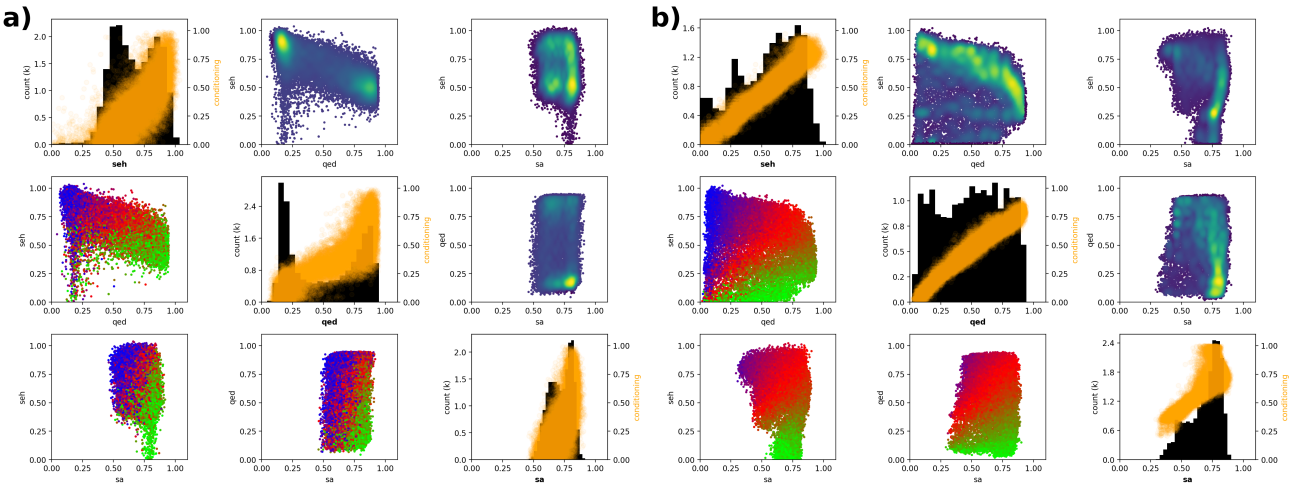

Figure 10. Idem to Figure 10 but with 3 objectives: seh, qed, sa.

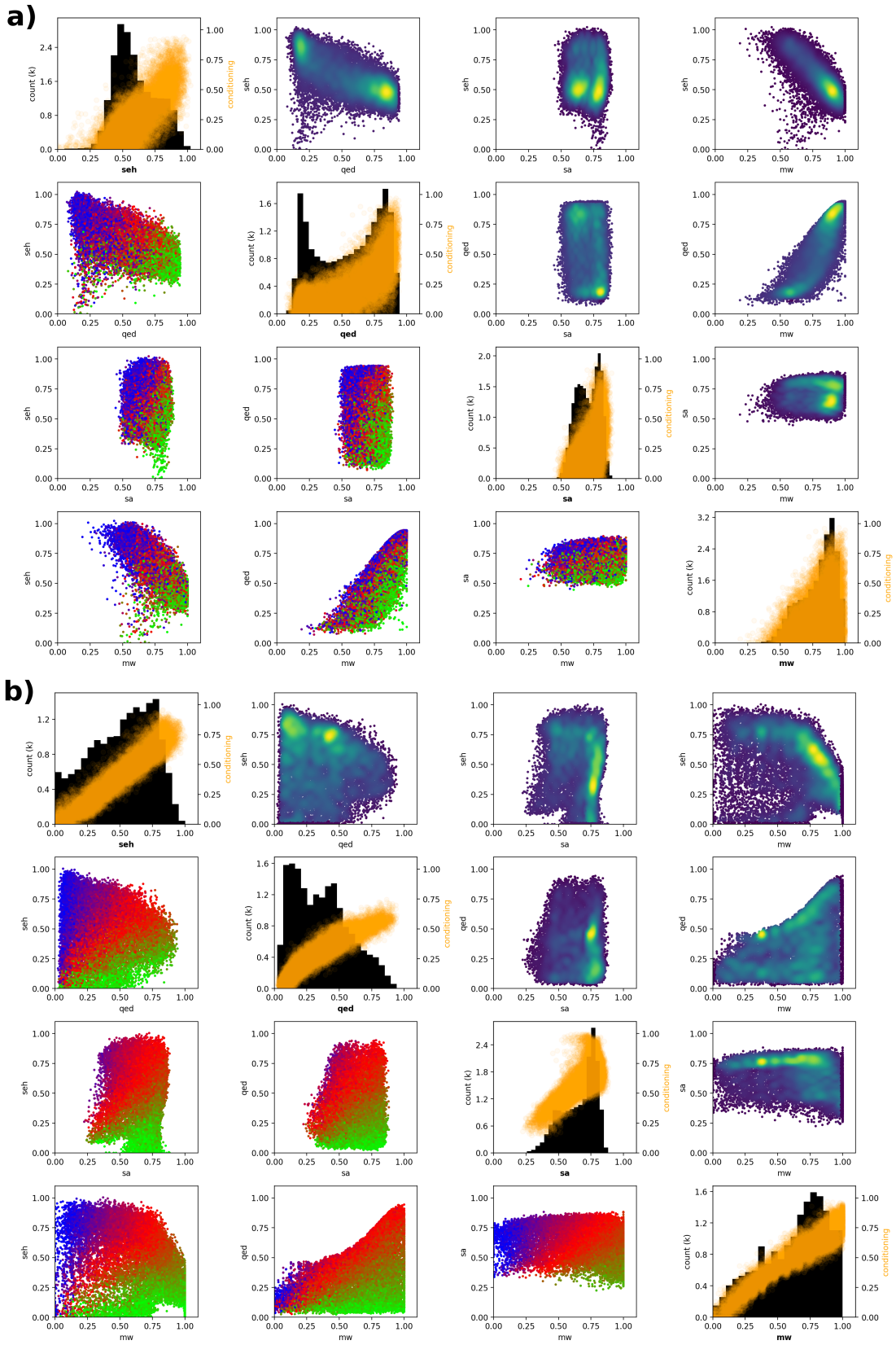

Figure 11. Idem to Figure 9 but with 4 objectives: seh, qed, sa, mw.