SCALABLE AND COST-EFFICIENT DE NOVO TEMPLATE-BASED MOLECULAR GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in reaction-based molecular generation hold great promise for drug design. Composing a molecule from a predefined set of reaction templates and building blocks keeps the generative modeling in line with what can be synthesized in a real-world wet lab. In this paper, we tackle three important challenges of template-based GFlowNets: 1) reducing the synthesis cost, 2) navigating in a large set of building blocks, and 3) exploiting a small set of building blocks. We propose Cost Guidance for a backward policy that uses an auxiliary machine-learning model to approximate the synthesis cost. Our approach limits the costs of proposed molecules, while drastically improving their diversity and quality in large-scale settings. Moreover, we design a Dynamic Library mechanism that allows the generation of full synthesis trees, boosting the results in small-scale settings. The resulting generative model establishes state-of-the-art results in template-based molecular generation in a benchmark concerning synthesis cost and diversity of high-rewarded molecules.

1 INTRODUCTION

Generative models offer a promise of accelerating drug discovery by vastly expanding the search space of potential molecules through direct sampling from the distribution of compounds with desired properties. Recent research has focused on the difficulty of synthesizing the generated molecules using template-based approach, where generated molecules are assembled from a predefined set of chemical building blocks in a sequence of reactions Gottipati et al. (2020); Horwood & Noutahi (2020). Recent studies Koziarski et al. (2024); Cretu et al. (2024); Seo et al. (2024) have integrated this concept into the GFlowNet framework Bengio et al. (2021), which is especially appealing for drug discovery given its natural mode-seeking tendencies and ability to explore massive chemical spaces effectively. Despite the emergence of multiple GFlowNet-based methods aimed at synthesizability, challenges persist in scalability and cost-awareness. An ideal generative framework would explore the entire space of synthesizable molecules while favoring those that are simpler and cheaper to synthesize. Existing methods do not yet achieve this balance.

In this paper, we propose SCENT (**S**calable and **C**ost-**E**fficient de **N**ovo **T**emplate-Based Molecular Generation) which tackles the aforementioned limitations of reaction-based GFlowNets. Our contributions can be summarized as follows:

- We design a Recursive Cost Guidance for backward policy that helps SCENT navigate the vast molecular space, improving the diversity of generated molecules and reducing their cost, especially in settings with a large number of building blocks.
- We design a Dynamic Building Block Library framework to enable the generation of fully-shaped synthesis trees. Dynamic Library is shown to increase the number of modes discovered in settings with a small number of initial building blocks.
- We investigate the mechanism of cost reduction in SCENT and develop a simple yet effective strategy that mitigates the exploitative nature of our Cost Guidance, called Exploitation Penalty. We show that it further improves the SCENT results by discouraging revisits of the same molecular scaffolds.

We benchmark recent reaction-based GFlowNets across three building block settings, demonstrating that SCENT efficiently identifies high-reward molecules at a lower cost than alternative approaches. A graphical summary of our contributions can be found in Figure 1.

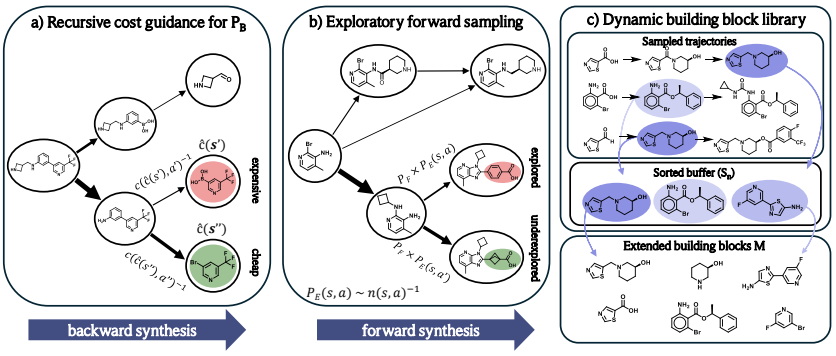

Figure 1: Recursive Cost Guidance (a) uses a machine learning model $\hat{c}$ to estimate the recursively defined cost of the previous molecules. Our backward policy P_B prefers cheaper synthesis paths and then guides the forward policy P_F. To boost the exploration during the training, we augment the forward sampling with a simple Exploitation Penalty technique (b). Dynamic Library (c) gathers intermediate molecules with the highest expected reward and adds them to the building block library M, enabling full-tree synthesis.

2 METHOD

In this section, we describe the details of SCENT and introduce 1) Recursive Cost Guidance, 2) Dynamic Building Block Library and 3) Exploitation Penalty.

2.1 RECURSIVE COST GUIDANCE

In this section, we introduce a new way of incorporating the preference over trajectories in GFlowNets Shen et al. (2023). Let $c(\tau) \in \mathbb{R}$ be a function that assigns a cost to a partial trajectory $\tau = (s_0, a_0, \ldots, s_n)$ starting from source state s_0. In the context of reaction-based GFlowNet, $c(\tau)$ may correspond to the cost of running a sequence of reactions from τ to synthesize the final molecule s_n. Furthermore, let $c(s) = \min_{\tau \in \mathcal{T}} c(\tau_{\to s})$ denote the minimum cost of arriving at state s.

Having those definitions, we can guide the GFlowNet towards cheaper trajectories using the recursive definition:

$$P_B((s'_i, a'_i)|s) = \sigma_i^{|SA'|}(s), \;\; s_i = -\alpha f(c(s'_i), s'_i, a'_i, s) \tag{1}$$

where SA' is a set of all parent state-action tuples (s', a') and f is a function defining the cost of arriving to s through s'_i and a'_i. A key feature of our framework is handling the cost when it's **recursive**, making explicit computation *intractable*.

To mitigate this exponential nature of f, we estimate the underlying recursive component using a neural network $\hat{c}(s')$ that is trained on data collected during the training to satisfy the recursivity constraint. Then we can plug in the estimate $\hat{c}(s')$ in Equation (1) instead:

$$P_B((s'_i, a'_i)|s) = \sigma_i^{|SA'|}(s), \;\; s_i = -\alpha f(\hat{c}(s'_i), s'_i, a'_i, s) \tag{2}$$

Note that f is not a learnable function as it describes our preference over the trajectories. The only learnable part of P_B is $\hat{c}(s')$ which is trained jointly with P_F.

2.1.1 RECURSIVE SYNTHESIS COST GUIDANCE

In this paper, we define c as the synthesis cost, which allows us to obtain a P_B that favors cheaper synthesis paths. The synthesis cost is defined as follows:

$$f(c(s'), a', s', s) = (c(a') + c(s'))y(s', a', s)^{-1}, \tag{3}$$

where $c(a')$ denotes the cost of fragments (recall that $a' = (M', r)$ is a pair of fragments M' and reaction template r), and $y(s', a', s)$ is the yield (the expected percentage of reacted molecules)) of the reaction r leading from molecule s' to s along with additional M' fragments.

Details on how we estimate the yield and fragments' cost can be found in Appendix C.

2.1.2 RECURSIVE DECOMPOSABILITY GUIDANCE

An important issue in reaction-based GFlowNets is to ensure that the parent molecule s' in $P_B(s', a'|s)$ can be recursively decomposed all the way back into building blocks from M. The authors of RGFN Koziarski et al. (2024) checked it explicitly, leading to a huge computational overhead. On the other hand, Cretu et al. (2024) proposed two implicit strategies to deal with the decomposability issue: an RL-based approach in which the backward reward is positive if the backward trajectory arrives at s_0 and negative when stuck on a non-decomposable molecule; and a pessimistic backward policy that trains P_B with maximum likelihood objective over trajectories sampled by P_F.

In this paper, we take a different approach to tackle the decomposability problem and leverage our Recursive Cost-Guided Backward Policy framework. We define cost $d(\tau) = 0$ if τ starts with s_0 and $d(\tau) = \infty$ otherwise. Then the cost $d(s') = \min_{\tau \in \mathcal{T}}(d(\tau_{\to s})) = 0$ if d' is decomposable and ∞ otherwise. The recursive cost definition f for Decomposability Guidance takes a simple form: $f(d(s'), s', a', s) = d(s')$ and can be efficiently approximated using $\hat{d}(s')$.

To separately deal with the cost associated with the decomposability of a molecule and the synthesis cost, we define two separate backward sub-policies, $P_B^c(a, s|s')$ for the Cost Guidance and $P_B^d(a, s|s')$ for Decomposability Guidance:

$$P_B(a, s|s') \propto P_B^c(a, s|s') P_B^d(a, s|s') \tag{4}$$

2.2 DYNAMIC BUILDING BLOCKS LIBRARY

During training, we gather intermediate molecules encountered in the trajectories sampled by P_F (and possibly P_B using a prioritized replay buffer) in a buffer. Let m be a molecule from the buffer and $s(m) \in \mathbb{R}$ its score. Let $\mathcal{T}_m$ be the set of trajectories that contain m and let s_τ be the final state of a trajectory τ. Then:

$$s(m) = \frac{1}{|\mathcal{T}_m|} \sum_{\tau \in \mathcal{T}_m} R(s_\tau) \tag{5}$$

$s(m)$ is thus interpreted as the "expected utility" of a molecule m and captures the usefulness of that molecule in generating high-reward synthesizable products which is the essence of library learning. As such, the Dynamic Library algorithm can be summarized as follows:

1. For each trajectory τ, update the scores for all intermediate molecules in τ and save in a buffer
2. Filter the buffer:
   (a) Keep molecules that have been obtained using no more than `max_reaction=2` reactions.
   (b) Keep molecules that match at least `min_matches=10` reactant patterns in our predefined set of templates.
   (c) Finally, Keep molecules that are not already present in the Dynamic Library
3. We sort the filtered buffer and keep the L highest-expected utility molecules.
4. Finally, we extend the Dynamic Library with these L molecules.

The above steps are repeated every T iteration for a maximum N_{add} iterations.

2.3 EXPLOITATION PENALTY

Appendix E.2 shows that Recursive Cost Guidance increases the exploitative behavior of SCENT. To mitigate that, we develop a strategy that discourages SCENT from re-taking the same actions at a given state during the training. We augment P_F using an exploratory policy P_E:

$$P_F'(a|s) \propto P_F(a|s) P_E(a|s) \tag{6}$$

We define Exploitation Penalty P_E using the following count-based Bellemare et al. (2016); Tang et al. (2017); Ostrovski et al. (2017) approach:

$$P_E(a|s) = \frac{(n(s, a) + \epsilon)^{-\gamma}}{\sum_{a'}(n(s, a') + \epsilon)^{-\gamma}}, \tag{7}$$

where $n(s, a)$ is the number of (s, a) occurrences in the sampled trajectories. Note that as opposed to pseudo-count methods our method directly assesses the number of visits and omits the erroneous

density estimation Pan et al. (2022); Burda et al. (2018); Pathak et al. (2017). The simplicity comes with the cost of seemingly poor scaling capabilities - we need to store every visited state and action which may be intractable when exploring the state space. However, Appendix E.2 shows that using the exploitation penalty is highly beneficial even when used in the initial 1000 iterations, making the memory needed to store the visited states constant with respect to the number of training iterations.

3 EXPERIMENTS

Table 1: Online discovery results.

	model	modes $> 8.0 \uparrow$	scaff. $> 8.0 \uparrow$
SMALL	RGFN	538 ±21	5862 ±354
	SynFlowNet	10.0 ±0.8	27.0 ±8.6
	RxnFlow	6.0 ±2.45	9.0 ±3.74
	SCENT (w/o C)	510 ±26	5413 ±334
	SCENT (C)	478 ±11	5150 ±87
	SCENT (C+D)	596 ±19	7148 ±497
	SCENT (C+P)	595 ±20	6839 ±360
	SCENT (C+D+P)	**715 ±15**	**8768 ±363**
MEDIUM	RGFN	4755 ±541	10638 ±918
	SynFlowNet	288 ±28	299 ±32
	RxnFlow	26.3 ±3.9	27.3 ±2.5
	SCENT (w/o C)	9310 ±863	11478 ±823
	SCENT (C)	17705 ±4224	52340 ±4303
	SCENT (C+P)	**37714 ±3430**	**90068 ±9010**
LARGE	SynFlowNet	278 ±282	385 ±401
	RxnFlow	24.5 ±2.5	25.0 ±2.0
	SCENT (w/o C)	7171 ±291	8767 ±429
	SCENT (C)	12375 ±264	36930 ±5455
	SCENT (C+P)	**26367 ±3193**	**46202 ±10242**

Table 2: Inference sampling results.

	model	reward $\uparrow$	sim. $> 8.0 \downarrow$	scaff. $> 8.0 \uparrow$	cost $> 8.0 \downarrow$
SMALL	RGFN	7.34 ±0.06	0.24 ±0.02	90.3 ±25.6	37.7 ±2.0
	SynFlowNet	6.89 ±0.29	-	4.33 ±4.19	-
	RxnFlow	6.13 ±0.03	-	0.33 ±0.47	-
	SCENT (w/o C)	7.38 ±0.02	0.25 ±0.02	97.3 ±14.3	37.1 ±1.0
	SCENT (C)	7.43 ±0.02	0.26 ±0.01	106 ±8	23.7 ±4.0
	SCENT (C+D)	7.56 ±0.05	**0.23 ±0.01**	202 ±34	**19.7 ±2.0**
	SCENT (C+P)	7.42 ±0.05	0.24 ±0.01	103 ±26	20.7 ±2.6
	SCENT (C+D+P)	**7.66 ±0.03**	0.24 ±0.0	**240 ±11**	20.7 ±3.0
MEDIUM	RGFN	7.08 ±0.08	0.21 ±0.02	46.3 ±13.5	1268 ±71
	SynFlowNet	6.38 ±0.03	-	1.67 ±0.47	1952 ±548
	RxnFlow	6.3 ±0.02	-	0.0 ±0.0	-
	SCENT (w/o C)	7.31 ±0.09	**0.19 ±0.01**	74.3 ±19.1	1463 ±62
	SCENT (C)	7.74 ±0.04	0.22 ±0.01	303 ±40	**1163 ±147**
	SCENT (C+P)	**7.81 ±0.04**	0.2 ±0.01	**360 ±21**	1230 ±121
LARGE	SynFlowNet	6.39 ±0.19	-	2.33 ±2.62	-
	RxnFlow	6.14 ±0.08	-	0.0 ±0.0	-
	SCENT (w/o C)	7.13 ±0.12	**0.19 ±0.01**	53.7 ±25.8	1678 ±63
	SCENT (C)	7.52 ±0.09	0.21 ±0.01	209 ±74	**1267 ±159**
	SCENT (C+P)	**7.76 ±0.03**	0.2 ±0.0	**326 ±47**	1291 ±66

In this section, we evaluate SCENT against recent template-based GFlowNets, namely RGFN Koziarski et al. (2024), SynFlowNet Cretu et al. (2024) and RxnFlow Seo et al. (2024). The chosen models explicitly output a reaction path using a shared backbone of reaction templates and building blocks. This way we can directly compare the costs of the proposed molecules without the need for erroneous retro-synthesis. We focus on GFlowNet-based models in this paper as this framework outperforms other approaches in mode-seeking metrics that we report in the paper.

Our SCENT method consists of four components: 1) Decomposability Guidance, 2) Cost Guidance (C), 3) Dynamic Library (D), and 4) Exploitation Penalty (P). The first component is used in all experiments with SCENT. The rest of the components may be turned on or off, depending on the experimental setting, which we indicate explicitly.

3.1 SETUP

Note that the results previously reported for those methods are not directly comparable: the methods use different sets of templates and building blocks. We propose a fair benchmark that aligns those two important factors. The models are evaluated in three settings: 1) SMALL: templates and building blocks described in Appendix C, 2) MEDIUM: templates from SynflowNet along with 64k molecules from Enamine, and 3) LARGE: templates from SynflowNet along with 128k molecules from Enamine. These settings cover a wide range of potential applications of SCENT. The SMALL setting contains a curated set of cheap fragments and is easier for automated quick-chemistry or rapid experimental turnover. MEDIUM and LARGE settings can be used for extensive molecular space exploration with building blocks that need to be ordered from an external supplier.

The models were trained on sEH proxy Bengio et al. (2021). We report the number of molecules with high reward whose mutual Tanimoto distance for ECFP6 fingerprints Rogers & Hahn (2010) is lower than 0.5; and the number of unique Bemis-Murcko scaffolds discovered during the training. Moreover, we sample 1000 synthesis paths using the trained forward policies and report the overall mean reward. Then filter the high-reward molecules and report their average Tanimoto similarity, the number of unique Bemis-Murcko scaffolds, and the average cost of the corresponding synthesis paths. The training details and hyperparameters used are reported in Appendix B.

3.2 COMPARISON TO TEMPLATE-BASED GENERATIVE MODELS

Table 1 and 2 show the evaluation results for the considered GFlowNet approaches on sEH proxy. SCENT with Cost Guidance (C) reduces cost in all settings, while drastically increasing the number of discovered modes and scaffolds in MEDIUM and LARGE. Enabling Dynamic Library (C+D) boosts the online discovery performance in the SMALL setting and maintains the low cost of generated molecules obtained with Cost Guidance. The number of discovered modes and scaffolds is further improved by the Exploration Penalty (+P). The resulting SCENT (C+D+P) for SMALL setting and (C+P) on MEDIUM and LARGE outperforms all considered models by a large margin in terms of diversity, quality, and cost of discovered molecules.

4 CONCLUSION

In this paper, we presented SCENT, a template-based approach for molecular generation within the GFlowNet framework. SCENT introduces three new mechanisms that collectively enhance exploration of chemical space: (1) recursive Cost Guidance, which focuses sampling on molecules with low synthesis cost, (2) Dynamic Library building blocks, which improve exploration efficiency - especially with smaller fragment libraries - and enable the generation of non-linear synthesis trees, and (3) an Exploration Penalty, which avoids repeated visits to previously explored candidates. Through empirical evaluations, we demonstrated that combining these strategies leads to state-of-the-art results in template-based molecular generation across diverse experimental settings - from small-scale fragment libraries suitable for rapid, automated synthesis, to large-scale libraries that create vast search spaces. We anticipate that this framework will accelerate drug discovery, be integrated into self-driving labs, and potentially extend to other scenarios such as molecular optimization of known hits, which remains an avenue for future work.

REFERENCES

Tomasz Badowski, Karol Molga, and Bartosz A. Grzybowski. Selection of cost-effective yet chemically diverse pathways from the networks of computer-generated retrosynthetic plans. *Chem. Sci.*, 10:4640–4651, 2019. doi: 10.1039/C8SC05611K. URL http://dx.doi.org/10.1039/C8SC05611K.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

Oussama Boussif, Léna Néhale Ezzine, Joseph D Viviano, Michał Koziarski, Moksh Jain, Nikolay Malkin, Emmanuel Bengio, Rim Assouel, and Yoshua Bengio. Action abstractions for amortized sampling. *arXiv preprint arXiv:2410.15184*, 2024.

Matthew Bowers, Theo X Olausson, Lionel Wong, Gabriel Grand, Joshua B Tenenbaum, Kevin Ellis, and Armando Solar-Lezama. Top-down synthesis for library learning. *Proceedings of the ACM on Programming Languages*, 7(POPL):1182–1213, 2023.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Miruna Cretu, Charles Harris, Julien Roy, Emmanuel Bengio, and Pietro Liò. Synflownet: Towards molecule design with guaranteed synthesis pathways. *arXiv preprint arXiv:2405.01155*, 2024.

P Kingma Diederik. Adam: A method for stochastic optimization. *(No Title)*, 2014.

Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3d de novo molecule generation, 2024. URL https://arxiv.org/abs/2404.19739.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. Translation between molecules and natural language, 2022. URL https://arxiv.org/abs/2204.11817.

Kevin Ellis, Lucas Morales, Mathias Sablé-Meyer, Armando Solar-Lezama, and Josh Tenenbaum. Learning libraries of subroutines for neurally-guided bayesian program induction. *Neural Information Processing Systems (NIPS)*, 2018.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, pp. 835–850, 2021.

Yin Fang, Ningyu Zhang, Zhuo Chen, Lingbing Guo, Xiaohui Fan, and Huajun Chen. Domain-agnostic molecular generation with chemical feedback, 2024. URL https://arxiv.org/abs/2301.11259.

Jenna Fromer and Connor Coley. An algorithmic framework for synthetic cost-aware decision making in molecular design. *Nature Computational Science*, 4:1–11, 06 2024. doi: 10.1038/s43588-024-00639-y.

Sai Krishna Gottipati, Boris Sattarov, Sufeng Niu, Yashaswi Pathak, Haoran Wei, Shengchao Liu, Simon Blackburn, Karam Thomas, Connor Coley, Jian Tang, et al. Learning to navigate the synthetically accessible chemical space using reinforcement learning. In *International conference on machine learning*, pp. 3668–3679. PMLR, 2020.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d, 2022. URL https://arxiv.org/abs/2203.17003.

Julien Horwood and Emmanuel Noutahi. Molecular design in synthetically accessible chemical space via deep reinforcement learning. *ACS omega*, 5(51):32984–32994, 2020.

Liang Huang, Tianfan Xu, Yadi Yu, et al. A dual diffusion model enables 3d molecule generation and lead optimization based on target pockets. *Nature Communications*, 15:2657, 2024. doi: 10.1038/s41467-024-46569-1. URL https://doi.org/10.1038/s41467-024-46569-1.

Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, jan 2022. doi: 10.1088/2632-2153/ac3ffb. URL https://dx.doi.org/10.1088/2632-2153/ac3ffb.

Ross Irwin, Alessandro Tibo, Jon Paul Janet, and Simon Olsson. Efficient 3d molecular generation with flow matching and scale optimal transport, 2024. URL https://arxiv.org/abs/2406.07266.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pp. 9786–9801. PMLR, 2022.

Moksh Jain, Tristan Deleu, Jason Hartford, Cheng-Hao Liu, Alex Hernandez-Garcia, and Yoshua Bengio. Gflownets for ai-driven scientific discovery. *Digital Discovery*, 2(3):557–577, 2023a.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International Conference on Machine Learning*, pp. 14631–14653. PMLR, 2023b.

Michał Koziarski, Andrei Rekesh, Dmytro Shevchuk, Almer van der Sloot, Piotr Gaiński, Yoshua Bengio, Cheng-Hao Liu, Mike Tyers, and Robert A Batey. Rgfn: Synthesizable molecular generation using gflownets. *arXiv preprint arXiv:2406.08506*, 2024.

Micha Livne, Zulfat Miftahutdinov, Elena Tutubalina, Maksim Kuznetsov, Daniil Polykovskiy, Annika Brundyn, Aastha Jhunjhunwala, Anthony Costa, Alex Aliper, Alán Aspuru-Guzik, and Alex Zhavoronkov. nach0: multimodal natural and chemical languages foundation model. *Chemical Science*, 15(22):8380–8389, 2024. ISSN 2041-6539. doi: 10.1039/d4sc00966e. URL http://dx.doi.org/10.1039/D4SC00966E.

Hannes H Loeffler, Jiazhen He, Alessandro Tibo, Jon Paul Janet, Alexey Voronov, Lewis H Mervin, and Ola Engkvist. Reinvent 4: Modern ai–driven generative molecule design. *Journal of Cheminformatics*, 16(1):20, 2024.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022a.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. Gflownets and variational inference. *arXiv preprint arXiv:2210.00580*, 2022b.

Elia Mazuz, Gabi Shtar, Bar Shapira, and Lior Rokach. Molecule generation using transformers and policy gradient reinforcement learning. *Scientific Reports*, 13:8799, 2023. doi: 10.1038/s41598-023-35648-w. URL https://doi.org/10.1038/s41598-023-35648-w.

Amy McGovern and Richard S Sutton. Macro-actions in reinforcement learning: An empirical analysis. *Computer Science Department Faculty Publication Series*, pp. 15, 1998.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Ling Pan, Dinghuai Zhang, Aaron Courville, Longbo Huang, and Yoshua Bengio. Generative augmented flow networks. *arXiv preprint arXiv:2210.03308*, 2022.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

Nicholas T Runcie and Antonia SJS Mey. Silvr: Guided diffusion for molecule generation. *Journal of Chemical Information and Modeling*, 63(19):5996–6005, 2023.

Seonghwan Seo, Minsu Kim, Tony Shen, Martin Ester, Jinkyoo Park, Sungsoo Ahn, and Woo Youn Kim. Generative flows on synthetic pathway for drug design. *arXiv preprint arXiv:2410.04542*, 2024.

Max W Shen, Emmanuel Bengio, Ehsan Hajiramezanali, Andreas Loukas, Kyunghyun Cho, and Tommaso Biancalani. Towards understanding and improving gflownet training. In *International Conference on Machine Learning*, pp. 30956–30975. PMLR, 2023.

Yuxuan Song, Jingjing Gong, Minkai Xu, Ziyao Cao, Yanyan Lan, Stefano Ermon, Hao Zhou, and Wei-Ying Ma. Equivariant flow matching with hybrid probability transport, 2023. URL https://arxiv.org/abs/2312.07168.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Clement Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. Midi: Mixed graph and 3d denoising diffusion for molecule generation, 2023. URL https://arxiv.org/abs/2302.09048.

Ziqi Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9:10752, 2019. doi: 10.1038/s41598-019-47148-x. URL https://doi.org/10.1038/s41598-019-47148-x.

A Related works

Generative models for molecular design. Various paradigms exist for small molecule generation using machine learning. These can be classified both by the molecular representations they aim to generate, which include molecular graphs, SMILES strings, and atomic coordinates, and by the deep learning frameworks employed to achieve these goals, which include diffusion Hoogeboom et al. (2022); Runcie & Mey (2023); Huang et al. (2024); Vignac et al. (2023) and flow matching models Song et al. (2023); Irwin et al. (2024); Dunn & Koes (2024), reinforcement learning Zhou et al. (2019); Mazuz et al. (2023); Loeffler et al. (2024), and large language models Irwin et al. (2022); Edwards et al. (2022); Fang et al. (2024); Livne et al. (2024). One such framework, Generative Flow Networks (GFlowNets) Bengio et al. (2021); Malkin et al. (2022b); Jain et al. (2023a; 2022; 2023b), has gained popularity as an alternative to traditional reinforcement learning for iterative molecular graph and SMILES generation due to its theoretical emphasis on diversity in sampling, which is critical to hit-finding in drug discovery. Notably, recent GFlowNet Cretu et al. (2024); Koziarski et al. (2024); Seo et al. (2024) and reinforcement learning Gottipati et al. (2020); Horwood & Noutahi (2020) work has explored synthesizable molecular design, wherein molecular building blocks are combined along known high-yield chemical reactions to generate molecules along plausible chemical synthesis pathways.

Cost-aware molecular synthesis In addition to properties like binding affinity, practical factors such as the cost of starting materials, reaction conditions, and the ability to conduct parallel chemistry are crucial in prioritizing molecules within generative frameworks. Synthetic feasibility is influenced by factors like reaction complexity, number of steps, and reagent accessibility. These factors and their often nonlinear influence on one another contribute to total cost being notoriously difficult to quantify. Existing methods by Fromer & Coley (2024) and Badowski et al. (2019) typically involve selecting from existing retrosynthetic trees a subset that minimizes a single- or multi-objective cost criterion. In contrast, our approach incorporates the cost prediction mechanism into the backward policy of the template-based GFlowNet, increasing the flow on the cheaper pathways. This way, the forward sampling (synthesis) is implicitly biased towards cheaper molecules and the backward retrosynthesis proposes cheaper pathways for already sampled high-reward molecules.

Library learning is a well-studied method for navigating large search spaces, particularly in program induction, where it has achieved significant success. For example, DreamCoder Ellis et al. (2021; 2018); Bowers et al. (2023) employs a two-step process: programs sampled during training are collected into a corpus, and a library synthesis algorithm is applied to extract the most common subroutines, which are then added to the library of primitives. In the Options framework Sutton et al. (1999), the library or set of actions is kept fixed, and a set of policies (called options) is learned instead. Another closely related framework involves the learning of macro-actions McGovern & Sutton (1998)—sequences of primitive actions that form a temporal abstraction. In settings where macro-actions are beneficial, they have been shown to improve both credit assignment and exploration. Recently, Boussif et al. (2024) introduced ActionPiece, a framework for applying library learning to any task by compressing the actions using BPE in a set of trajectories to extract the most common subsequences and augment the action space. Their approach has demonstrated success in probabilistic modeling, achieving state-of-the-art results for GFlowNet-based methods. Our approach is closely related to the latter, with the key distinction being that we modify the library of fragments by focusing on intermediate states rather than actions.

B Models Training

All the models sampled 320000 forward trajectories during the training in total in SMALL and MEDIUM settings and 256000 in LARGE. All the forward policies were trained using the Trajectory Balance Objective Malkin et al. (2022a) and their parameters were optimized using Adam Diederik (2014). All the methods were trained using their built-in action embedding mechanisms and on the maximum number of reactions equal to 4. The hyperparameters were chosen semi-manually or using small grid searches to maximize the number of high-reward modes in the SMALL setting on sEH proxy.

B.1 SCENT

We followed most of the hyperparameter choices from the official implementation of RGFN Koziarski et al. (2024). We set the number of sampled forward trajectories to 64, and the number of trajectories sampled from the prioritized replay buffer to 32. We set the β to 64 for sEH, and 48 for GSK3β and JNK3. We used uniform ϵ-greedy exploration with $\epsilon = 0.05$ for all our experiments, except for those with other exploration techniques (e.g. Exploitation Penalty).

All SCENT instances were trained using decomposability guidance, either solely or with Cost Guidance using Equation (4).

Decomposability Guidance To predict the decomposability of the molecule, we use the model described in Appendix D. We set the cost temperature (Equation (1)) to $\alpha = 5$.

Synthesis Cost Guidance The model for synthesis Cost Guidance is described in Appendix D. We set the cost temperature to $\alpha = 5$.

Dynamic Building Blocks Library We update the dynamic building blocks library every $T = 1000$ iterations for a maximum of $N_{\text{add}} = 10$ additions (meaning that we do this until the end of training). The number of molecules added to the library every time it's updated is $L = 400$.

Exploitation Penalty We set $\epsilon = 3$ in Equation (7) and schedule γ in two ways: 1) so that it linearly decays to zero after N iterations, and 2) it grows with the trajectory length (each step increases it by some constant factor $\Delta\gamma$). We set $N = \infty$ for SCENT that uses Dynamic Library and $N = 1000$ for the rest of the runs. The growing delta $\Delta\gamma = 0.2$, while the initial s_0 temperature $\gamma_0 = 1.0$. Note that for JNK3 in the MEDIUM setting, we used 4500 iterations for exploitation penalty.

B.2 RGFN

We use the official implementation of RGFN Koziarski et al. (2024) with mostly default parameters. We changed the number of sampled forward trajectories to 64, and the number of trajectories sampled from the prioritized replay buffer to 32. We set the β to 64 for sEH, and 48 for GSK3β and JNK3.

B.3 SynFlowNet

We followed the setting from the paper Cretu et al. (2024) and the official repository. We used **MaxLikelihood** or REINFORCE backward policy with **disabled** or enabled replay buffer. We grid searched GFlowNet reward temperature β: $\{\mathbf{32}, 64\}$ for sEH and $\beta \in \{\mathbf{16}, 32\}$ for GSK3β and JNK3.

We trained SynFlowNet using 32 forward trajectories and 8000 iterations on LARGE setting, and 64 forward trajectories and 5000 iterations for SMALL and MEDIUM.

B.4 RxnFlow

We used the default parameters from the official implementation and adapted 1) the `action_subsampling_ratio` so that the number of sampled actions is close to the number used in their paper, and 2) GFlowNet beta temperatures. We set `action_subsampling_ratio=1.0` for the SMALL setting, 0.2 for MEDIUM, and 0.1 for LARGE. We set the β sampling distribution to Uniform$(16, 64)$ for sEH, and Uniform$(16, 48)$ for GSK3β and JNK3.

C Reaction Yields and Fragments Costs Estimates

SMALL setting The reaction set for the SMALL setting is the extended reaction set from Koziarski et al. (2024). We added Bishler-Napieralski and Pictet-Spengler reactions, along with

various aryl functionalizations, benzoxazole, benzimidazole, and benzothiazole formation reactions, Hantzsch thiazole synthesis, alkylation of aromatic nitrogen, and Williamson ether synthesis.

Building block prices for SMALL were obtained from the vendors online. If multiple vendors offered the same building block, only the cost of the cheapest option was considered. For compounds available in varying amounts, both the price per gram and the alternative size (e.g., 5, 10, 25 grams) were considered, and the smallest price per mmol was used. Compounds that were exclusively accessible in other forms, such as salts, related functional groups, or containing a protected functional group were used with the SMILES fitted to the corresponding reactions. The cost of a product includes the total cost of the building blocks used, while additional expenses such as solvents, catalysts, and reagents are not accounted for.

Reaction yield estimates for amide coupling, nucleophilic additions to isocyanates, Suzuki reaction, Buchwald-Hartwig coupling, Sonogashira coupling, and azide-alkyne cycloaddition were obtained by analyzing in-house data. For all other reactions, SMARTS templates were used to search for reaction substructure in Reaxys and SciFinder, and the obtained literature data was analyzed to produce yield estimates.

MEDIUM and LARGE setting We used subsets of building blocks from Enamine of size 64000 for LARGE and 128000 for MEDIUM settings. We downloaded the fragments' costs from their website.

D Cost Prediction Model Training

We use two cost prediction models in SCENT: synthesis cost prediction model $\hat{c}(s)$ and decomposability prediction model (let us override the notation and denote it as $\hat{d}(s)$). The size of mini-batches, the dataset size N, and the negative sampling ratio were manually chosen to maximize the cost prediction validation losses gathered during the training.

D.1 Synthesis Cost Model

The cost prediction model $\hat{c}(s)$ is a simple multi-layer perception on top of ECFP4 fingerprint $e(s) \in \mathbb{R}^{2048}$:

$$\hat{c}(s) = W_1\phi(W_2 e(s)),\ W_1 \in \mathbb{R}^{1\times 128},\ W_2 \in \mathbb{R}^{128\times 2048}, \tag{8}$$

where θ is the GELU activation function Hendrycks & Gimpel (2016).

To train the model we maintain a dataset D with current estimates of the states/molecules $c(s)$ denoted as c_s. The dataset is updated with trajectories sampled from GFlowNet during the training and its size is kept below $N = 10000$. Given a trajectory τ, we update all c_s for states $s \in \tau$ using the cost of the partial trajectory τ_s leading to state s: $c'_s = \min(c_s, c(\tau_s))$. To stabilize training, we standardize all costs in D using the mean and variance of costs from the building block library M.

Given an updated dataset D, we perform five mini-batch updates of $\hat{c}$ using mini-batches of size 1024, learning rate of 0.01 and mean-square-error loss $\mathcal{L}_{cost} = ||c(s) - \hat{c}(s)||_2^2$.

Importantly, when computing the recursive costs in our cost-guided backward policy P_B from Equation (2), we use the most optimistic estimate of the cost by using $\min(\hat{c}(s), c_s)$ if the considered cost $c_s \in D$.

D.2 Decomposability Model

The decomposability model $\hat{d}(s)$ is very similar to the synthesis cost model, except that it encodes the current number of used reactions in the molecule s. The "cost" in the context of decomposability is 0 if the given molecule can be decomposed into building blocks from M (within some number of steps) and ∞ otherwise. In practice, we train the model $\hat{d}(s)$ to predict decomposability label $\in \{0, 1\}$ using binary cross entropy $\mathcal{L}_{cost}$ (the "cost" is then $-\hat{d}(s)$). The labels are gathered similarly to synthesis cost values, using the trajectories encountered in the GFlowNet's forward sampling. We update the $\hat{d}(s)$ with five mini-batches of size 1024 and learning rate of 0.005. We additionally ensure that at least 20% of the sampled molecules for $\hat{d}(s)$ training are not decomposable. In the

inference, we use the most optimistic estimate of cost by using $\min(-\hat{c}(s), c_s)$ in P_B computation if considered cost $c_s \in D$.

E STUDIES OF RECURSIVE SYNTHESIS COST GUIDANCE

In this section, we study the proposed synthesis Cost Guidance mechanism in SCENT.

E.1 HOW DOES COST REDUCTION WORK?

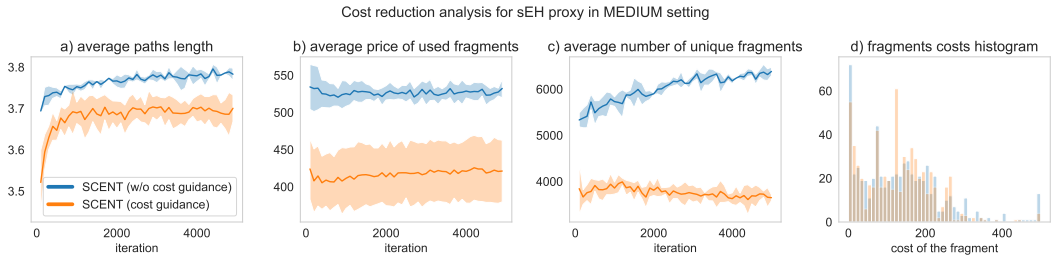

Figure 2: Cost reduction analysis for sEH proxy and MEDIUM setting. Cost Guidance decreases the average length of the sampled trajectory (a), and the average cost of used fragments (b) which directly reduces the synthesis path cost. It also focuses on a smaller fraction of fragments during generation (c) and minimizes the usage of the most expensive ones (d). The plots are smoothened by averaging results from the last 100 iterations.

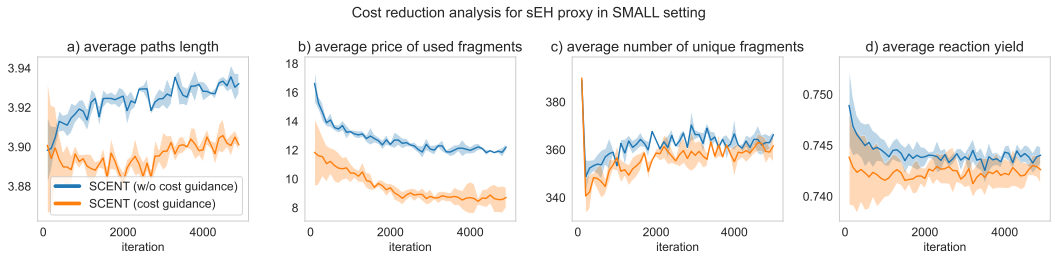

Figure 3: Cost reduction analysis for sEH proxy and MEDIUM setting. Cost Guidance decreases the average length of the sampled trajectory (a), and the average cost of used fragments (b) which directly reduces the synthesis path cost. Importantly, the average reaction yield (d) does not increase when using Cost Guidance, suggesting its low influence in reducing the synthesis costs. The plots are smoothened by averaging results from the last 100 iterations.

We analyze how Cost Guidance reduces the average reaction path cost generated by P_F. Figure 2-a shows that Cost Guidance reduces the trajectory length, which lowers the overall cost of the trajectory. Additionally, Figure 2-b and -d demonstrate that Cost Guidance selects cheaper fragments and significantly limits the use of more expensive ones. Finally, Figure 2 illustrates that Cost Guidance relies on a subset of fragments throughout training, which correlates with the frequent use of cheaper molecules. Figure 3 presents the same analysis for the SMALL setting.

E.2 EXPLOITATION ANALYSIS

According to Table 1 and 2, Cost Guidance increases the number of discovered scaffolds more than it increases the number of discovered modes. By our definition, a set of modes is a set of high-rewarded molecules that has mutual Tanimoto similarity below 0.5. It captures the notion of structural diversity much better than the number of scaffolds also often reported as a number of ”modes”.

Therefore, the divergence in the improvement over the number of scaffolds and modes may indicate that Cost Guidance increases the exploitativeness. In Figure 4, we plot how often P_F resamples previously discovered scaffolds with high-reward. Cost Guidance revisits the scaffolds drastically

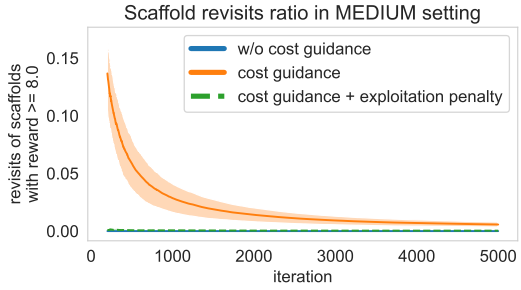

Figure 4: Frequency of revisits of high-rewards scaffolds during training. Cost Guidance drastically increases the ratio of revisits, suggesting its exploitative nature. By incorporating the exploitation penalty into Cost Guidance, we reduce the revisits ratio.

more often than SCENT without Cost Guidance, suggesting its more exploitative nature. However, adding the exploitation penalty limits this behavior.

E.3 Recursive Cost Guidance Helps in Scaling

Encouraged by the results of Cost Guidance in MEDIUM and LARGE settings, we performed additional experiments to show how the size of the building block library M influences the gap between cost guided SCENT, SCENT without Cost Guidance, and SCENT with additional Exploitation Penalty. Figure 5 shows that Cost Guidance drastically increases the number of discovered scaffolds from the smallest (2k molecules) to the largest (256k molecules) M. On the other hand, we observe that the gap in the number of discovered modes is raised up to M of size 128k, but then drops in the largest setting.

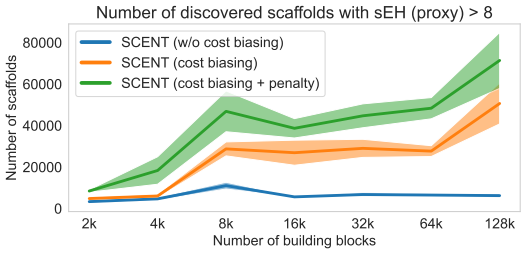

Figure 5: Number of discovered scaffolds as a function of the size of building block library M. The numbers are reported for the 3000th training iteration.

The fact that using Cost Guidance improves the mode-seeking metrics is initially not intuitive. Results from Appendix E.1 suggest that it navigates the fragment space more effectively, reducing the number of used fragments. While limiting the size of the building block library by randomly cropping some fragments may increase the results according to Cretu et al. (2024) and Koziarski et al. (2024), the effect of the reduction is by far less prominent than our performance gains. Similarly, the removal of the most expensive fragments helps the mode-seeking metrics, but to a limited extent. As such, we believe that Cost Guidance learns to select building blocks that are both cheap and useful for the given reward.