Appendix: A Chance-Constrained Generative Framework for Sequence Optimization

Xianggen Liu [1 2] **Qiang Liu** [3] **Sen Song** [1] **Jian Peng** [2]

A. Lower Bound of Validity that Meets the Constraint

Based on the objective $\mathcal{J}$ of CCGF defined in Eqn. (11) in the manuscript, below, we analyze the lower bound of the generation validity that meets the constraint.

Let v be the mean validity of the sequences generated by G_θ, i.e., $v = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[g(G_\theta(\xi_i)) > T]$, where $\mathbb{1}[\cdot]$ is the indicator function. For notational simplicity, we assume that the function g is a binary indicator, then we will have $v = \frac{1}{N}\sum_{i=1}^{N} g(G_\theta(\xi_i))$, so we know there are Nv valid sequences and $N(1-v)$ invalid ones. Note that it is not hard to obtain a similar analysis for arbitrary g function.

Since the scale of the objective $\mathcal{J}$ is affected by α, α should be updated slowly to make the learning process stable. For a given α, the constraint defined in our CCO problem (i.e., Eqn. (11) in the manuscript) is equivalent to

$$\begin{aligned} &h(a, v) \geq 0, \\ \text{where } h(\alpha, v) &= \epsilon - \frac{1}{N}\sum_{i=1}^{N} e^{-\alpha(g(G_\theta(\xi_i))-T)} \\ &= \epsilon - [ve^{-\alpha(1-T)} + (1-v)e^{\alpha T}]. \end{aligned}$$

The partial derivative of $h(a, v)$ with respect to v is

$$\frac{\partial h(a, v)}{\partial v} = -e^{-\alpha(1-T)} + e^{\alpha T}, \tag{1}$$

Notice that $e^{\alpha T} \geq 1 \geq e^{-\alpha(1-T)}$ for all values of T and $\alpha \geq 0$, leading to $\frac{\partial h(a,v)}{\partial v} \geq 0$. For a certain value of α, let $h(\alpha, v_0) = 0$, we have

$$v_0 = \frac{e^{\alpha T} - \epsilon}{e^{\alpha T}(1 - e^{-\alpha})}. \tag{2}$$

For a given α and any $v \geq v_0$, we have $h(\alpha, v) \geq 0$, indicating v_0 is the lower bound of validity of the generated sequences that meets the constraint.

B. Trend of the Lower Bound of Validity

Eqn. (2) shows that, besides T and ϵ, the lower bound of validity v_0 will also vary with α when the optimization process proceeds.

Considering the partial derivative of v_0 with respect to α,

$$\begin{aligned} \frac{\partial v_0}{\partial \alpha} &= \epsilon e^{-\alpha T}(1 - 2e^{-\alpha}) + e^{-\alpha} &(3)\\ &= \epsilon e^{-\alpha T}(1 - e^{-\alpha}) + e^{-\alpha}(1 - \epsilon e^{-\alpha T}) \geq 0, &(4) \end{aligned}$$

we know the lower bound of validity always keeps pace with the changes of α.

Theorem 1. *For any $T \in (0, 1), \epsilon \in (0, 1)$ and $v \in [0, 1]$, there exists $\hat{\alpha} \in [0, +\infty)$ and once the variable $\alpha \geq \hat{\alpha}$, we have $\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \alpha)}{\partial \alpha} \leq 0$.*

Proof.

$$\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \alpha)}{\partial \alpha} = \lambda \frac{\partial [h(\alpha, v)]_-}{\partial \alpha}. \tag{5}$$

(1) If $h(\alpha, v) > 0$, we have $[h(\alpha, v)]_- = 0$ and $\frac{\partial [h(\alpha,v)]_-}{\partial \alpha} = 0$. Therefore, for all $\alpha \in (0, +\infty)$ we have

$$\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \alpha)}{\partial \alpha} = 0. \tag{6}$$

In this case $\hat{\alpha}$ could be any non-negative value.

(2) If $h(\alpha, v) \leq 0$, the partial derivatives of $h(\alpha, v)$ with respect to α are given by

$$\begin{aligned} \frac{\partial h(\alpha, v)}{\partial \alpha} &= e^{\alpha T}(v(1-T)e^{-\alpha} - (1-v)T), &(7)\\ \frac{\partial}{\partial \alpha}(\frac{\partial h(\alpha, v)}{\partial \alpha}) &= -[e^{\alpha T}(v(1-T)^2 e^{-\alpha} + (1-v)T^2] < 0, &(8) \end{aligned}$$

[1]Laboratory for Brain and Intelligence and Department of Biomedical Engineering, Tsinghua University, Beijing, China. [2]Department of Computer Science, University of Illinois at Urbana Champaign, IL, USA. [3]Department of Computer Science, University of Texas at Austin, TX, USA. Correspondence to: Xianggen Liu <liuxg16@mails.tsinghua.edu.cn>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

where we notice the second partial derivative of $h(\alpha, v)$ regarding α is negative, leading to the monotonically decreasing trend of $\frac{\partial h}{\delta \alpha}$ along α. Let $\frac{\partial h}{\delta \alpha}(\alpha_1, v_1) = 0$, we have

$$\alpha_1 = -\log \frac{(1-v_1)T}{v_1(1-T)}. \tag{9}$$

Thus, when $\alpha \geq \alpha_1$, we have $\frac{\partial h}{\delta \alpha}(\alpha, v_1) \leq 0$. It also lead to $\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \alpha)}{\partial \alpha} \leq 0$ when $v = v_1$. To make $\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \alpha)}{\partial \alpha} \leq 0$ for all v, we choose the maximum of α_1 to be $\hat{\alpha}$, given by

$$\hat{\alpha} = \max_{v_1 \in [0,1]} [\alpha_1] \tag{10}$$

$$s.t., \frac{\partial h}{\delta \alpha}(\alpha_1, v_1) = 0 \text{ and } h(\alpha_1, v_1) \leq 0. \tag{11}$$

$\hat{\alpha}$ is essentially the intersection of $h(\alpha, v) = 0$ and $\frac{\partial h(\alpha, v)}{\partial \alpha} = 0$, which can be numerically solved (See Appendix C for more details).

Therefore, there exists an $\hat{\alpha}$ that makes $\frac{\partial \mathcal{J}(\boldsymbol{\theta}, \alpha)}{\partial \alpha} \leq 0$ for all T, ϵ and v when $\alpha \geq \hat{\alpha}$. □

Remarks 1. The above theorem shows that, once $\alpha \geq \hat{\alpha}$, both the value of α and the lower bound of validity v_0 will continue to increase to minimize $\mathcal{J}(\boldsymbol{\theta}, \alpha)$. That is to say, under the condition of $\alpha \geq \hat{\alpha}$, our CCGF framework will gradually improve the lower bound of validity as the optimization process proceeds. According to Eqn. (2), when $\alpha \to +\infty$, $v_0 \to 1$. This coincides with the idea of simulated annealing (Kirkpatrick et al., 1983): the initial low validity requirement enables the searching agent escape from the realm of locality in the beginning of learning. Afterwards, the stricter validity requirement would reduce the search space and accelerate the optimization.

To make Theorem 1 work throughout the optimization process, one of the easiest ways is to initialize α with the value of $\hat{\alpha}$. Fortunately, the value of $\hat{\alpha}$ can be numerically obtained and its determination is provided in Appendix C.

C. Determination of $\hat{\alpha}$

We are going to solve the following problem

$$\hat{\alpha} = \max_{v_1 \in [0,1]} [\alpha_1] \tag{12}$$

$$s.t., \alpha_1 = -\log \frac{(1-v_1)T}{v_1(1-T)} \tag{13}$$

$$h(\alpha_1, v_1) \leq 0. \tag{14}$$

However, we are not able to derive the analytic solution of $\hat{\alpha}$ or the exact value of $\hat{\alpha}$ since the function h is a transcendental function. We resort to obtain α numerically. For example, we enumerate the values of v_1 from 0 to 1 and keep the maximum value of α_1 that meets Eqn. 14 in a greedy manner. At the end of searching, the kept α_1 is the approximate numeric solution of $\hat{\alpha}$.

To intuitively understand the process of searching α, we show the approximate location of $\hat{\alpha}$ in Figure 1 under the setting of $T = \epsilon = 0.5$. The gray shadow indicates the region that satisfies $h(\alpha_1, v_1) \leq 0$. The red dash line stand for the function α_1 with respect to v_1. We notice the maximum of α_1 that satisfies $h(\alpha_1, v_1) \leq 0$ locates at the intersection of the two functions, i.e., $h(\alpha_1, v_1) = 0$ and $\alpha_1 = -\log \frac{(1-v_1)T}{v_1(1-T)}$.

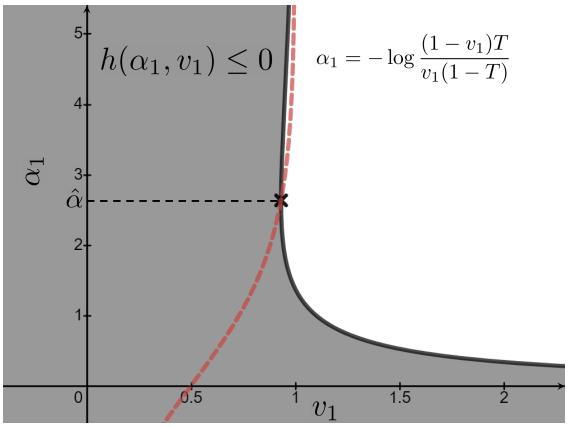

Figure 1. The approximate location of $\hat{\alpha}$

D. Analysis on the classification threshold T_{acc}

We investigate the influences of classification threshold T_{acc} to the performance of CCGF on the task of optimizing the penalized logP of molecules. We choose T_{acc} from $\{0.05, 0.10, 0.15, 0.20\}$ and fix other hyperparameters of CCGF. As shown in Table 1, a lower classification threshold leads to higher property scores but lower diversity. To make the diversity of generated sequences by CCGF higher or comparable to those of other methods, we choose 0.1 to be the value classification threshold T_{acc}. The effects of classification threshold T_{acc} on the domains of expressions and programs are similar. Therefore, we set the global value of T_{acc} to 0.1.

Table 1. The diversity and the optimization performance of the sequences generated by CCGF with different classification thresholds T_{acc}. We use the pair-wise distance to indicate the diversity. It is calculated by $1 - s$, where s is the similarity score of two molecules based on their MorganFp fingerprints.

T_{acc}	Diversity	Top 50 Avg.	Validity
0.20	0.68 ± 0.09	4.47	93.6%
0.15	0.65 ± 0.16	8.21	96.4%
0.10	0.57 ± 0.29	10.15	98.8%
0.05	0.39 ± 0.31	11.58	99.6%

E. Hyperparameter Selection

The hyperparameters tuning process involves the constant ϵ in the constraint, the two weights γ and λ in objective functions of CCGF, hidden size m of RNN, the learning rate r of generative model $G_{\boldsymbol{\theta}}$ and the learning rate r_α of α, two weights γ_c and λ_c in objective functions of RL-Linear, where γ_c, similar to γ, is a weight that coordinates the importance of the property score. We perform grid search procedures to calibrate the optimal values of all the above hyperparameters for each task on a small subset of the corresponding dataset (10%). The selected values of all the above hyperparameters are listed in Table 2.

F. Examples

Figure 2 illustrates the 3D structures of the molecules with the highest penalized logP scores found by CCGF and GraphAF, the latter of which is the previously state-of-the-art optimization approach. Following the color convention in chemistry, individual sizes and colors of atoms stand for different chemical elements, where black atoms stand for carbon and red for oxygen. We notice that the molecules generated by our CCGF framework possess higher penalized logP scores.

G. Implementation Details of individual baselines in Section 4.5

In Section 4.5, our goal is to optimize the penalized logP score and in the meantime satisfy the Lipinski rule. We include four baselines into comparison with our framework, namely, MolDQN (Zhou et al., 2019), GVAE (Kusner et al., 2017), SD-VAE (Dai et al., 2018), and GraphAF (Shi et al., 2020).

However, the latter three models (i.e., GVAE, SD-VAE and GraphAF) could not be directly applied into this task since they are originally designed to optimize only one property of molecules. Following Zhou et al. (2019), we extend their objectives by imposing the Lipinski rule into the their original objective function, which is given by

$$\mathcal{O}(x) = \frac{1}{N}\sum_{i}^{N} w\mathcal{O}_{\text{ori}}(x) + (1-w)\text{Lipinski}(x), \quad (15)$$

where $\mathcal{O}_{\text{ori}}(x)$ is the original objective function and $\text{Lipinski}(x)$ is the binary function that checks whether the sequence x meets the Lipinski rule. w is the weight that coordinates the importance of Lipinski rule. For each competitive method, we perform grid search procedures on $w \in \{0.02, 0.05, 0.1, 0.2, 0.4\}$ to obtain best results.

To obtain the results of MolDQN (Zhou et al., 2019), we use its published source code[1] and only modify its objective as described above. The selected value of w is 0.02.

As for GVAE (Kusner et al., 2017) and SD-VAE (Dai et al., 2018), we use their published codes[2,3] and published pretrained models to conduct this experiment. Concretely, based on the latent spaces learned by the pretrained models, we apply Bayesian optimization algorithm to search sequences with highest scores (evaluated by the modified objectives). The selected values of w are 0.1 for both models.

GraphAF (Shi et al., 2020) is a flow-based autoregressive model that dynamically generates the nodes and edges based on existing sub-graph structures. When it is required to optimize a desirable property, GraphAF is fine-tuned through reinforcement learning algorithm (in practice, proximal policy optimization (Schulman et al., 2017)). We use its published code[4] and we fine-tune the pretrained model based on modified objective function. The value of w is set to 0.2 by validation.

[1] https://github.com/google-research/google-research/tree/master/mol_dqn

[2] GVAE: https://github.com/mkusner/grammarVAE.

[3] SD-VAE: https://github.com/PfizerRD/sdvae

[4] https://chenceshi.com/

Table 2. The hyperparameters of CCGF and RL-Linear on individual tasks. We perform grid search procedures to calibrate their optimal values in each task over $\epsilon \in \{0.1, 0.5, 0.9\}$, $\lambda \in \{0.01, 0.02, \ldots, 0.1\}$, $\lambda_c \in \{0.5, 1, 2, 3\}$, $r \in \{1 \times 10^{-4}, 5 \times 10^{-4}\}$, $r_\alpha \in \{0.01, 0.03, 0.05, 0.07\}$, and $m \in \{256, 512\}$. The searching ranges of γ and γ_c are task dependent.

Task	Methods	ϵ	γ	λ	γ_c	λ_c	r	r_α	m	Range of γ/γ_c
Expressions	CCGF	0.5	0.2	0.03	-	-	1×10^{-4}	0.03	512	$\{0.1, 0.2, 0.3, 0.4, 0.5\}$
Programs	CCGF	0.5	0.25	0.04	-	-	1×10^{-4}	0.03	512	$\{0.1, 0.15, 0.2, 0.25, 0.3\}$
	RL-Linear	-	-	-	0.25	2	1×10^{-4}	-	512	$\{0.1, 0.15, 0.2, 0.25, 0.3\}$
Druglikeness	CCGF	0.5	1	0.05	-	-	1×10^{-4}	0.01	512	$\{0.5, 1, 1.5, 2, 2.5\}$
Solubility	CCGF	0.5	0.4	0.05	-	-	1×10^{-4}	0.01	512	$\{0.1, 0.2, 0.3, 0.4, 0.5\}$
	RL-Linear	-	-	-	0.4	2	1×10^{-4}	-	512	$\{0.1, 0.2, 0.3, 0.4, 0.5\}$
Solubility & Lipinski	CCGF	0.5	0.08	0.1	-	-	1×10^{-4}	0.01	512	$\{0.02, 0.04, 0.06, 0.08, 0.1\}$

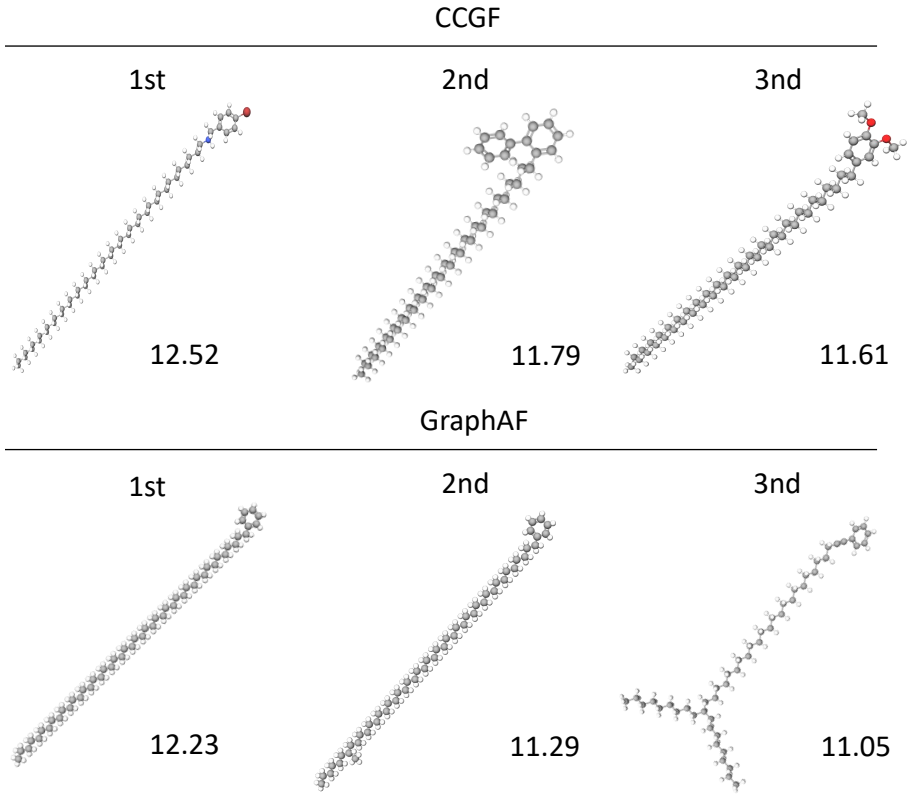

Figure 2. Molecules with high penalized logP scores generated by CCGF and GraphAF. Following the color convention in chemistry, individual sizes and colors of atoms stand for different chemical elements, where black atoms stand for carbon and red for oxygen.

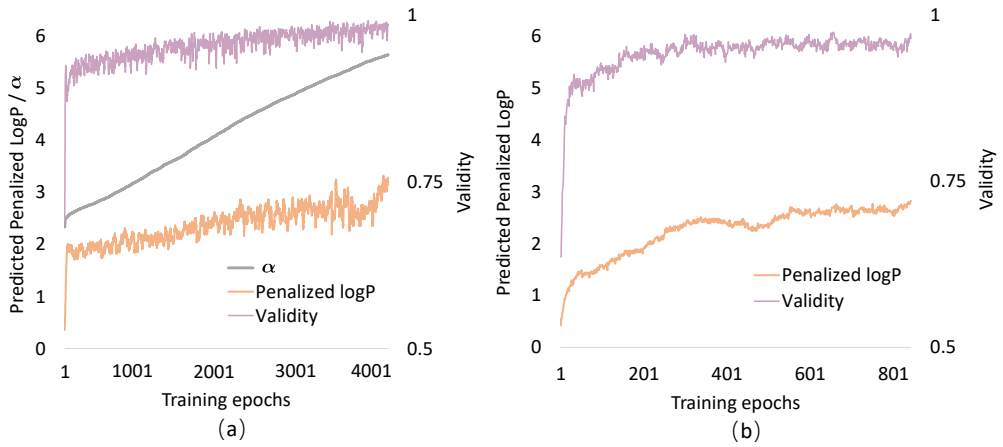

Figure 3. The learning curves of CCGF (a) and RL-Linear (b) regarding the average score of logP, average validity and α.

References

Dai, H., Tian, Y., Dai, B., Skiena, S., and Song, L. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

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

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. Grammar variational autoencoder. *Proceedings of the 34th International Conference on Machine Learning*, 70: 1945–1954, 2017.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. GraphAF: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020.

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.