GRAPHEBM: MOLECULAR GRAPH GENERATION WITH ENERGY-BASED MODELS

Meng Liu, Keqiang Yan, Bora Oztekin & Shuiwang Ji
Department of Computer Science & Engineering
Texas A&M University
College Station, TX 77843, USA
{mengliu,keqiangyan,bora,sji}@tamu.edu

ABSTRACT

We note that most existing approaches for molecular graph generation fail to guarantee the intrinsic property of permutation invariance, resulting in unexpected bias in generative models. In this work, we propose GraphEBM to generate molecular graphs using energy-based models. In particular, we parameterize the energy function in a permutation invariant manner, thus making GraphEBM permutation invariant. We apply Langevin dynamics to train the energy function by approximately maximizing likelihood and generate samples with low energies. Furthermore, to generate molecules with a desirable property, we propose a simple yet effective strategy, which pushes down energies with flexible degrees according to the properties of corresponding molecules. Finally, we explore the use of GraphEBM for generating molecules with multiple objectives in a compositional manner. Comprehensive experimental results on random, goal-directed, and compositional generation tasks demonstrate the effectiveness of our proposed method.

1 INTRODUCTION

A fundamental problem in drug discovery and material science is to find novel molecules with desirable properties. One way is to search in the chemical space based on molecular property prediction (Gilmer et al., 2017; Wu et al., 2018; Yang et al., 2019; Stokes et al., 2020; Wang et al., 2020). Recently, molecular graph generation has provided an alternative and promising way for this problem by directly generating desirable molecules, thus circumventing the expensive search of the chemical space. Despite intensive efforts recently, molecular graph generation remains challenging since the underlying graphs are discrete, irregular, and permutation invariant to node order.

Existing approaches (Appendix A.1) have achieved promising success by generating molecular graphs based on various generative methods, including variational autoencoders (VAEs) (Kingma & Welling, 2013), generative adversarial networks (GANs) (Goodfellow et al., 2014), flow models (Dinh et al., 2014; Rezende & Mohamed, 2015) and recurrent neural networks (RNNs). However, as analyzed in Appendix A.1, most of them fail to preserve the intrinsic property of permutation invariance, which might yield different likelihoods for different permutations of the same graph.

Notably, energy-based models (EBMs) (LeCun et al., 2006) can also be naturally used as generative models since data points near the underlying data manifold are assigned lower energies than other data points, which defines an unnormalized probability distribution over the data. Given a data point x, let $E_\theta(x) \in \mathbb{R}$ be the corresponding energy, where θ denotes the learnable parameters of the energy function. Then, the energy function defines a data distribution via the Boltzmann distribution as

$$p_\theta(x) = \frac{e^{-E_\theta(x)}}{Z(\theta)} \propto e^{-E_\theta(x)}, \tag{1}$$

where $Z(\theta) = \int e^{-E_\theta(x)} dx$ is the normalization constant and usually intractable. Related works on EBMs are reviewed in Appendix A.2.

In this work, we propose GraphEBM to generate molecular graphs with energy-based models. Since our parameterized energy function is permutation invariant, we can show that our GraphEBM preserves the permutation invariance property. We use Langevin dynamics (Welling & Teh, 2011) to train the energy function by maximizing likelihood approximately and generate samples from the trained energy function. To our knowledge, our GraphEBM is the first energy-based model that can generate attributed molecular graphs. Furthermore, in order to generate molecules with a specific desirable property, we propose a novel, simple, and effective strategy to train our GraphEBM for goal-directed generation by pushing down energies with flexible degrees according to the property values of corresponding molecules. Significantly, we show that GraphEBM can generate molecules with multiple objectives in a compositional manner, which cannot be achieved by any existing methods. This provides a new and promising way for multi-objective molecule generation. Experimental results on random, goal-directed, and compositional generation tasks show that our proposed method is effective.

2 The Proposed GraphEBM

In this section, we present GraphEBM by describing the internal structure of the parameterized energy function (Section 2.1), showing that GraphEBM satisfies the desirable property of permutation invariance (Section 2.2), and describing the training (Section 2.3) and generation (Section 2.4) process of GraphEBM. Then, we introduce our proposed strategy for goal-directed generation based on GraphEBM (Section 2.5). Finally, we explore the potential of compositional generation using GraphEBM (Section 2.6).

Molecules can be naturally represented as graphs by considering atoms and bonds as nodes and edges, respectively. We formally represent a molecular graph as $\mathcal{G} = (X, A)$, where X is the node feature matrix and A is the adjacency tensor. Let k be the number of nodes in the graph. b and c denote the number of possible types of nodes and edges, respectively. Then we have $X \in \{0,1\}^{k\times b}$ and $X_{(i,p)} = 1$ if node i belongs to type p. $A \in \{0,1\}^{k\times k\times c}$ and $A_{(i,j,q)} = 1$ denotes that an edge with type q exists between node i and node j. Following Madhawa et al. (2019) and Zang & Wang (2020), we let n denote the maximum number of atoms that a molecule has in a given dataset. We insert virtual nodes into molecular graphs that have less than n nodes such that the dimensions of X and A keep the same for all molecules. Also, for any two nodes that are not connected in the molecule, we add a virtual edge between them. We can consider the virtual node and the virtual edge as an additional node type and edge type, respectively. Hence, for all molecules in a certain dataset, $X \in \{0,1\}^{n\times(b+1)}$ and $A \in \{0,1\}^{n\times n\times(c+1)}$.

2.1 Parameterized Energy Function

Following the above notations, the energy function for molecular graphs can be denoted as $E_\theta(X, A)$. Specifically, we model $E_\theta(X, A)$ by a graph neural network, where θ denotes parameters in the network. Many deep learning methods on graphs have been proposed and have achieved great success in many tasks, such as node classification (Kipf & Welling, 2017; Hamilton et al., 2017; Monti et al., 2017; Veličković et al., 2018; Gao et al., 2018; Xu et al., 2018; Klicpera et al., 2019; Wu et al., 2019; Liu et al., 2020b; Jin et al., 2020a; Chen et al., 2020; Liu et al., 2020c; Hu et al., 2020), graph classification (Zhang et al., 2018; Ying et al., 2018; Maron et al., 2018; Xu et al., 2019; Gao & Ji, 2019; Ma et al., 2019; Yuan & Ji, 2020; You et al., 2020; Gao et al., 2020), and link prediction (Zhang & Chen, 2018; Cai et al., 2020). In this work, we use a variant of relational graph convolutional networks (R-GCN) (Schlichtkrull et al., 2018) to learn the node representations since molecular graphs have categorical edge types. Formally, the layer-wise forward-propagation is defined as

$$H^{\ell+1} = \sigma\left(\sum_{k=1}^{c+1}\left(A_{(:,:,k)} H^{\ell} W_k^{\ell}\right)\right). \tag{2}$$

$A_{(:,:,k)}$ is the k-th channel of the adjacency tensor. $H^\ell \in \mathbb{R}^{n\times d_\ell}$ is the node representation matrix at layer ℓ, where d_ℓ denotes the hidden dimension at layer ℓ. $W_k^\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ represents the trainable weight matrix for edge type k at layer ℓ. $\sigma(\cdot)$ denotes a non-linear activation function. The initial node representation matrix $H^0 = X$. In each layer, message passing is conducted among the nodes independently for each type of edge. Then, the information is integrated together by a sum operator. We stack L such layers. Hence, the final node representation matrix is $H^L \in \mathbb{R}^{n\times d}$, where d is the hidden dimension. Then, the representation of the whole graph can be derived by a readout function.

In this work, we use the sum operation to compute the graph-level representation h_G as

$$h_G = \sum_{i=1}^{n} H^L_{(i,:)} \quad \in \mathbb{R}^d. \tag{3}$$

Finally, the scalar energy associated with the molecular graph can be obtained by applying a transformation as

$$E = h_G^T W \quad \in \mathbb{R}, \tag{4}$$

where $W \in \mathbb{R}^d$ is the trainable parameters.

2.2 Permutation Invariance

Permutation invariance is an intrinsic and desirable inductive bias for graph modeling. We note that our proposed GraphEBM satisfies this fundamental property due to our permutation invariant energy function. Specifically, each layer of our graph neural network in Eq. (2) is permutation equivariant. In addition, the readout operation in Eq. (3) is permutation invariant. Therefore, our parameterized energy function is permutation invariant thus satisfying $E_\theta(X, A) = E_\theta(X^\pi, A^\pi)$, where π denotes any permutation of node order. For simplicity, we use the superscript π to denote that the corresponding matrix or tensor is arranged according to the node order given by π. According to Eq. (1), the energy function defines a distribution over data. Specifically, the likelihood is proportional to the negative exponential of the corresponding energy. Hence, we can further obtain $p_\theta(X, A) = p_\theta(X^\pi, A^\pi)$. Thus, our GraphEBM can preserve permutation invariance by modeling graphs in a permutation invariant manner.

2.3 Training

Intuitively, a good energy function should assign lower energies to data points that correspond to real molecular graphs and higher energies to other data points. Hence, a straightforward idea is to train the parameterized energy function by maximizing the likelihood of real data defined in Eq. (1). Let p_D be the distribution of the real data. We can achieve maximum likelihood by minimizing the negative log likelihood of real data. Formally,

$$\mathcal{L}_{ML} = \mathbb{E}_{(X,A)\sim p_D}\left[-\log p_\theta\left(X, A\right)\right], \tag{5}$$

where $-\log p_\theta\left(X, A\right) = E_\theta(X, A) + \log Z(\theta)$ according to Eq. (1). It has been shown (Hinton, 2002; Turner, 2005; Song & Kingma, 2021) that the objective in Eq. (5) has the below gradient:

$$\nabla_\theta \mathcal{L}_{ML} = \mathbb{E}_{(X^\oplus, A^\oplus)\sim p_D}\left[\nabla_\theta E_\theta(X^\oplus, A^\oplus)\right] - \mathbb{E}_{(X^\odot, A^\odot)\sim p_\theta}\left[\nabla_\theta E_\theta(X^\odot, A^\odot)\right]. \tag{6}$$

As defined in Eq. (1), p_θ is the distribution given by the energy function. Following Du et al. (2020b), we refer to $(X^\odot, A^\odot)$ as hallucinated samples. Obviously, this gradient pushes down the energies of positive samples $(X^\oplus, A^\oplus)$ and pushes up the energies of hallucinated samples $(X^\odot, A^\odot)$. However, sampling $(X^\odot, A^\odot)$ from p_θ is challenging, since $Z(\theta)$ in Eq. (1) is intractable.

To overcome this issue, we follow Du & Mordatch (2019) to sample $(X^\odot, A^\odot)$ from an approximated p_θ using Langevin dynamics (Welling & Teh, 2011). Particularly, a sample $(X^\odot, A^\odot)$ is initialized randomly and refined iteratively by

$$X^k = X^{k-1} - \frac{\lambda}{2}\nabla_X E_\theta\left(X^{k-1}, A^{k-1}\right) + w^k; \quad A^k = A^{k-1} - \frac{\lambda}{2}\nabla_A E_\theta\left(X^{k-1}, A^{k-1}\right) + \eta^k, \tag{7}$$

where w^k and η^k are added noise sampled from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. k denotes the iteration step, and $\frac{\lambda}{2}$ is the step size. As demonstrated by Welling & Teh (2011), the obtained samples (X^k, A^k) approach samples from p_θ as $k \to \infty$ and $\frac{\lambda}{2} \to 0$. In practice, we let K denote the number of iteration steps of Langevin dynamics and use the resulting sample (X^K, A^K) as $(X^\odot, A^\odot)$ in Eq. (6).

We illustrate the training process of our GraphEBM in Figure 1. Since Langevin dynamics is for continuous data, we model the hallucinated samples by continuous format. For consistency, we can also use dequantization techniques (Dinh et al., 2016; Kingma & Dhariwal, 2018) to convert the

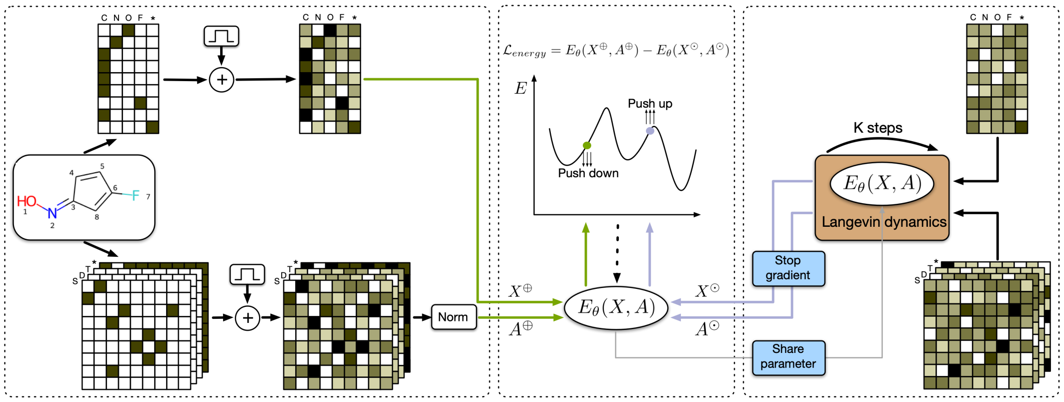

Figure 1: The training process of our GraphEBM. The left part and right part illustrate the processes of obtaining the positive sample and the hallucinated sample, respectively. The middle part shows the forward and backward propagation of the training process. Green and purple arrows represent the forward computation of energy value for the positive sample and the hallucinated sample, respectively. The black dashed arrow denotes the gradient backpropagation. In this example, $n = 9$, $k = 8$, $b = 4$, and $c = 3$. The annotations of node representation matrices denote the atom types, including carbon (C), nitrogen (N), oxygen (O), fluorine (F), and virtual atom ($\star$). Note that we remove hydrogen atoms, which is a common technique in the community. The annotations of adjacency tensor indicate the bond types, including single (S), double (D), triple (T), and virtual bond ($\star$).

discrete positive samples to continuous data by adding uniform noise, as shown in the left part of Figure 1. The dequantization can be formally expressed as

$$X' = X + tu, \quad u \sim [0, 1)^{n\times(b+1)}; \quad A' = A + tu, \quad u \sim [0, 1)^{n\times n\times(c+1)}. \tag{8}$$

$t \in [0, 1)$ is a scaling hyperparameter. We then apply a normalization to the adjacency tensor, which is a common step in modern graph neural networks (Kipf & Welling, 2017). Formally,

$$A^\oplus_{(:,:,k)} = D^{-1} A'_{(:,:,k)}, \quad k = 1, \cdots, c + 1, \tag{9}$$

where D is the diagonal degree matrix in which $D_{(i,i)} = \sum_{j,k} A'_{(i,j,k)}$. We treat the above $A^\oplus$ and $X^\oplus = X'$ as the input for the energy function. In our case, each element of $X^\oplus$ is in $[0, 1 + t)$ and each element of $A^\oplus$ is in $[0, 1)$.

Note that the above dequantization for positive samples is optional. This indicates that we can set $t = 0$ and keep the positive data discrete since Langevin dynamics is only required for obtaining hallucinated samples. Applying dequantization to positive samples can be viewed as a data augmentation technique and we can easily convert the dequantized continuous data back to the original one-hot discrete data by simply applying the argmax operation.

To keep the same value range as $(X^\oplus, A^\oplus)$, the hallucinated sample $(X^\odot, A^\odot)$ is initialized as

$$X^\odot \sim [0, 1 + t)^{n\times(b+1)}, \quad A^\odot \sim [0, 1)^{n\times n\times(c+1)}. \tag{10}$$

Then we apply K steps of Langevin dynamics as Eq. (7) to refine the sample, as illustrated in the right part of Figure 1. After each step of refinement, we clamp the data to guarantee that the values are still in the desirable ranges.

As demonstrated in Eq. (6), the energies of positive samples are expected to be pushed down and the energies of hallucinated samples should be pushed up. Hence, to shape the energy function as expected, our loss function is defined as

$$\mathcal{L}_{energy} = E_\theta(X^\oplus, A^\oplus) - E_\theta(X^\odot, A^\odot). \tag{11}$$

As shown in the middle part of Figure 1, the gradient backpropagated from $\mathcal{L}_{energy}$ can update the parameters θ, thus pushing the energy function $E_\theta(X, A)$ to approach our expected shape. Notably, the gradient from $\mathcal{L}_{energy}$ will not be propagated to the energy function used in Langevin dynamics. We apply parameter sharing to keep the energy function used in Langevin dynamics up-to-date.

To stabilize training, we also apply a regularization technique to the energy magnitudes. Specifically, we use the same regularization as Du & Mordatch (2019). Formally,

$$\mathcal{L}_{reg} = E_\theta(X^\oplus, A^\oplus)^2 + E_\theta(X^\odot, A^\odot)^2. \tag{12}$$

Hence, the total loss function is $\mathcal{L} = \mathcal{L}_{energy} + \alpha\mathcal{L}_{reg}$, where $\alpha \in \mathbb{R}$ is a hyperparameter.

2.4 GENERATION

Let $E_{\theta^\star}(X, A)$ denote the trained energy function, where $\theta^\star$ represents the obtained parameters. Intuitively, if an energy function is well-shaped, the configurations with low energies should correspond to desirable molecular graphs. Hence, the generation process is to generate molecules based on the configurations (X, A) that yield low energies.

An overview of the generation process is given in Figure 5 in Appendix B. The steps are as follows. First, we initialize a data point as in Eq. (10) and then apply K steps of Langevin dynamics as in Eq. (7) to obtain data points that have low energy. We denote the obtained configuration as $(X^\star, A^\star)$. Second, since molecular graphs are undirected, we make the adjacency tensor to be symmetric by using $A^\star + A^{\star T}$ as the new adjacency tensor. Third, we convert the continuous data to discrete ones by applying the argmax operation in the dimensions of atom types and bond types. Finally, we use validity correction introduced by Zang & Wang (2020) to refine the corresponding molecule so that the valency constraint is satisfied.

2.5 GOAL-DIRECTED GENERATION

For drug discovery and material design, we also need to generate molecules with desirable chemical properties. This task is termed as goal-directed generation. As noted in Appendix C, it is not straightforward to apply existing strageties to our GraphEBM for goal-directed generation. To generate molecules with desirable chemical properties, we propose a novel, simple, and effective strategy to achieve goal-directed generation based on our GraphEBM. Our basic idea is to push down energies with flexible degrees according to the property values of corresponding molecules. If a molecule has a higher value of desirable property, we push down the corresponding energy harder. Formally, in goal-directed generation, the loss function defined in Eq. (11) becomes

$$\mathcal{L}_{energy} = f(y)E_\theta(X^\oplus, A^\oplus) - E_\theta(X^\odot, A^\odot), \tag{13}$$

where $y \in [0, 1]$ is the normalized property value and $f(y) \in \mathbb{R}$ determines the degree of the push down. We use $f(y) = 1 + e^y$ in this work. Thus, energies of molecules with higher property values are pushed down harder. After training, the generation process is the same as described in Section 2.4. Note that $f(y)$ could also be a learnable function and we leave this as future work.

2.6 COMPOSITIONAL GENERATION

In addition to single property constraints, it is commonly necessary to generate molecules with multiple property constraints in drug discovery (Jin et al., 2020c). We observe that compositional generation with EBMs (Hinton, 2002), which has been shown to be effective in the image domain (Du et al., 2020a), can be naturally applied to generate molecules with multiple constraints based on our GraphEBM. Thus, we investigate compositional generation in the graph domain.

Suppose we have two energy functions $E_{\theta_1^\star}(X, A)$ and $E_{\theta_2^\star}(X, A)$ trained towards two property goals respectively, as described in Section 2.5. According to Hinton (2002) and Du et al. (2020a), we can obtain a new energy function by summing the above two energy functions since the product of probabilities is equivalent to the sum of corresponding energies, according to Eq. (1). Formally,

$$E_{\theta^\star}(X, A) = E_{\theta_1^\star}(X, A) + E_{\theta_2^\star}(X, A). \tag{14}$$

Then we can apply the generation process described in Section 2.4 to $E_{\theta^\star}(X, A)$ to generate molecules towards multiple objectives in a compositional manner.

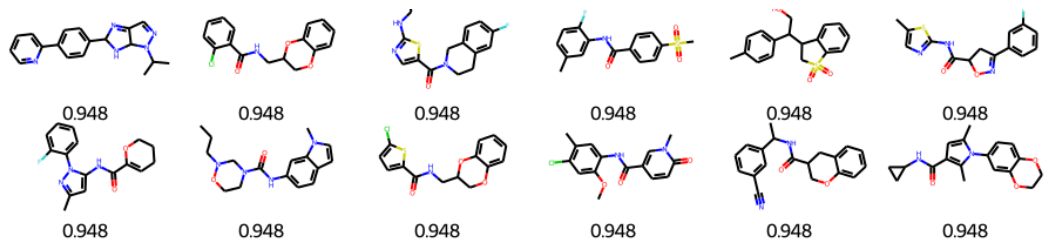

Figure 2: Discovered examples with high QED scores.

3 Experiments

We evaluate our proposed method for molecule generation under three settings: random generation, goal-directed generation, and compositional generation. We consider two widely used molecule datasets, QM9 (Ramakrishnan et al., 2014) and ZINC250k (Irwin et al., 2012). The details of the experimental setup for each setting are included in Appendix D. Our implementation is included in DIG[1] (Liu et al., 2021), a library for graph deep learning research.

Random generation. The results on QM9 and ZINC250k are shown in Table 4 and Table 5 respectively in Appendix E. We can observe that GraphEBM performs competitively with existing methods, which is significant considering that the study of EBMs is still in its early stage and GraphEBM is the first EBM for molecule generation. Generated samples are visualized in Figure 6 in Appendix E, which further demonstrates that GraphEBM can generate non-trivial molecules.

To better understand the implicit generation through Langevin dynamics, we visualize this process for an example in Figure 7 in Appendix E. We can observe that Langevin dynamics effectively refines the random initialized sample to approach a data point that corresponds to a realistic molecule.

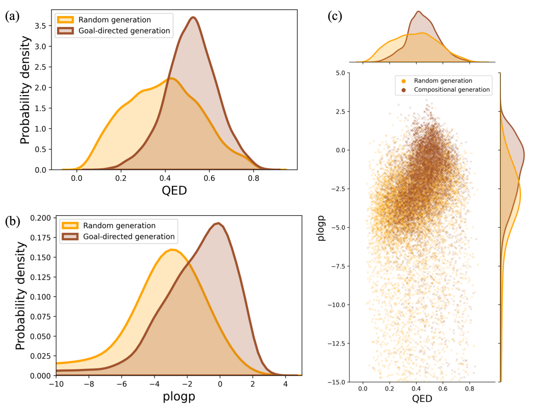

Figure 3: (a)&(b) Comparison of QED and plogp distributions between goal-directed generation and random generation, respectively. (c) Comparison of distributions on QED and plogp between compositional generation and random generation.

Goal-directed generation. Figure 3 (a) and (b) compare the property value distribution between goal-directed generation and random generation. It can be observed that goal-directed generation can generate more molecules with a high property value, indicating that our proposed strategy for goal-directed generation, which aims to assign lower energies to molecules with higher property values, is effective.

The property optimization results are shown in Table 1. We observe that GraphEBM can find more novel molecules with the best QED score (0.948) than baselines. This strongly demonstrates the effectiveness of our proposed goal-directed generation method. Examples of discovered novel molecules with high QED scores are illustrated in Figure 2

Table 1: Property optimization results.

Method	1st	2nd	3rd	4th
JT-VAE	0.925	0.911	0.910	-
GCPN	0.948	0.947	0.946	-
GraphAF	0.948	0.948	0.947	0.946
MoFlow	0.948	0.948	0.948	0.948
GraphEBM	**0.948**	**0.948**	**0.948**	**0.948**

For constraint property optimization, as demonstrated in Table 2, GraphEBM can obtain higher property improvements over JT-VAE, GCPN, and MoFlow by significant margins. In terms of success rate, our GraphEBM is not as strong as the methods using reinforcement learning (*i.e.*, GCPN, GraphAF), and achieves results comparable to JT-VAE

[1]https://github.com/divelab/DIG

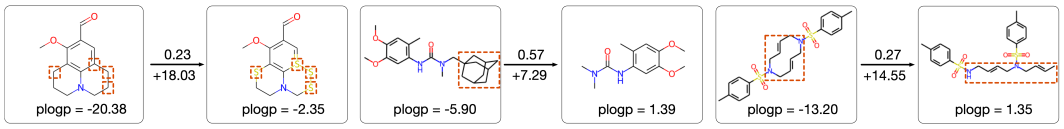

Figure 4: Examples of constraint property optimization. The values above and below arrows denote the similarity scores and improvements, respectively. The modifications are highlighted with red rectangles.

and MoFlow. Although GraphAF performs better than GraphEBM, it can be observed from Shi et al. (2019) that GraphAF learns to improve plogp by simply adding long carbon chains, while our GraphEBM learns more advanced chemical knowledge. Several examples of constraint property optimization are shown in Figure 4. It is interesting that the modifications are interpretable to some degree. Specifically, in the first example, our model optimizes the plogp score with a remarkable margin of 18.03 by replacing several carbon atoms with sulfur atoms, which could make the molecule more soluble in water, thus leading to a larger logP value. Additionally, plogp is highly related to the number of long cycles and synthetic accessibility. As shown in the second and third examples, our model improves the synthetic accessibility and reduces the number of long cycles by removing or breaking them. These facts indicate that our goal-directed generation method can explore the underlying chemical knowledge related to the corresponding property.

Compositional generation. The comparison of the distributions on QED and plogp between compositional generation and random generation is illustrated in Figure 3 (c). We can observe that compositional generation tends to generate more molecules with high QED and plogp scores. Additionally, the distribution of QED or plogp is similar to the corresponding distribution obtained by goal-directed generation towards a single objective (Figure 3 (a) and (b)). These facts demonstrate that our GraphEBM is able to generate molecules towards multiple objectives in a composition manner, which provides a novel and promising way for multi-objective generation.

Table 2: Constrained property optimization results. Imp., Sim., and Suc. denote Improvement, Similarity, and Success Rate, respectively.

	JT-VAE			GCPN			GraphEBM		
δ	**Imp.**	**Sim.**	**Suc.**	**Imp.**	**Sim.**	**Suc.**	**Imp.**	**Sim.**	**Suc.**
0.0	1.91	0.28	98%	4.20	0.32	100%	**5.76**	0.08	98%
0.2	1.68	0.33	97%	**4.12**	0.34	100%	3.97	0.35	92%
0.4	0.84	0.51	84%	2.49	0.47	100%	**2.84**	0.53	88%
0.6	0.21	0.69	46%	0.79	0.68	100%	**1.52**	0.68	64%

	GraphAF			MoFlow			GraphEBM		
δ	**Imp.**	**Sim.**	**Suc.**	**Imp.**	**Sim.**	**Suc.**	**Imp.**	**Sim.**	**Suc.**
0.0	13.13	0.29	100%	8.61	0.30	99%	**15.75**	0.01	99%
0.2	**11.90**	0.33	100%	7.06	0.43	97%	8.40	0.35	94%
0.4	**8.21**	0.49	100%	4.71	0.61	86%	4.95	0.54	79%
0.6	**4.98**	0.66	97%	2.10	0.79	58%	3.15	0.67	45%

4 Conclusion and Outlook

In this paper, we propose GraphEBM, the first energy-based model for generating molecular graphs that preserves the intrinsic property of permutation invariance. We propose to flexibly push down energies for goal-directed generation and explore to generate molecules towards multiple objectives in a compositional manner, leading to a promising method for multi-objective generation. Experimental results demonstrate that our GraphEBM can generate realistic molecules and the proposals of goal-directed generation and compositional generation are effective and promising. Since EBMs have unique advantages and have rarely been explored for graph generation, we hope our exploratory work would open the door for future research in this area.

5 Acknowledgments

We thank Yilun Du and Chengxi Zang for their helpful discussions. This work was supported in part by National Science Foundation grant DBI-1922969.

REFERENCES

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Xavier Bresson and Thomas Laurent. A two-step graph convolutional decoder for molecule generation. *arXiv preprint arXiv:1906.03412*, 2019.

Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. Line graph neural networks for link prediction. *arXiv preprint arXiv:2010.10046*, 2020.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Barry A Cipra. An introduction to the ising model. *The American Mathematical Monthly*, 94(10): 937–959, 1987.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, volume 32, pp. 3608–3618, 2019.

Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. *Advances in Neural Information Processing Systems*, 33, 2020a.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020b.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *International Conference on Machine Learning*, pp. 2083–2092, 2019.

Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1416–1424, 2018.

Hongyang Gao, Yi Liu, and Shuiwang Ji. Topology-aware graph pooling networks. *arXiv preprint arXiv:2010.09834*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th international conference on machine learning*, pp. 1263–1272, 2017.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pp. 599–619. Springer, 2012.

Shion Honda, Hirotaka Akita, Katsuhiko Ishiguro, Toshiki Nakanishi, and Kenta Oono. Graph residual flow for molecular graph generation. *arXiv preprint arXiv:1909.13521*, 2019.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Fenyu Hu, Yanqiao Zhu, Shu Wu, Weiran Huang, Liang Wang, and Tieniu Tan. Graphair: Graph representation learning with neighborhood aggregation and interaction. *Pattern Recognition*, pp. 107745, 2020.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52 (7):1757–1768, 2012.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 66–74, 2020a.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pp. 2323–2332, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning*, 2020b.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*, pp. 4849–4859. PMLR, 2020c.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pp. 10215–10224, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

MJ Kusner, B Paige, and JM Hernández-Lobato. Grammar variational autoencoder. In *Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017*, volume 70, pp. 1945–1954. ACM, 2017.

Greg Landrum et al. RDKit: Open-source cheminformatics. 2006.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. In *International Conference on Machine Learning*, 2018.

Phillip Lippe and Efstratios Gavves. Categorical normalizing flows via continuous transformations. *arXiv preprint arXiv:2006.09790*, 2020.

Jenny Liu, Will Grathwohl, Jimmy Ba, and Kevin Swersky. Graph generation with energy-based models. *ICML Workshop on Graph Representation Learning and Beyond (GRL+)*, 2020a.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, 2020b.

Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *arXiv preprint arXiv:2005.14612*, 2020c.

Meng Liu, Youzhi Luo, Limei Wang, Yaochen Xie, Hao Yuan, Shurui Gui, Zhao Xu, Haiyang Yu, Jingtun Zhang, Yi Liu, Keqiang Yan, Bora Oztekin, Haoran Liu, Xuan Zhang, Cong Fu, and Shuiwang Ji. DIG: A turnkey library for diving into graph deep learning research. *arXiv preprint arXiv:2103.12608*, 2021.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31: 7795–7804, 2018.

Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 7113–7124, 2018.

Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 723–731, 2019.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2017.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. *Artificial Intelligence and Statistics*, 2020.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*, 2019.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

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

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pp. 593–607. Springer, 2018.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2019.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pp. 412–422. Springer, 2018.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, pp. 11918–11930, 2019.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackerman, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

Richard Turner. CD notes. 2005.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representation*, 2018.

Zhengyang Wang, Meng Liu, Youzhi Luo, Zhao Xu, Yaochen Xie, Limei Wang, Lei Cai, and Shuiwang Ji. Advanced graph and sequence neural networks for molecular property prediction and drug discovery. *arXiv preprint arXiv:2012.01981*, 2020.

David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pp. 6861–6871, 2019.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Jianwen Xie, Wenze Hu, Song-Chun Zhu, and Ying Nian Wu. Learning sparse frame models for natural image patterns. *International Journal of Computer Vision*, 114(2-3):91–112, 2015.

Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *International Conference on Machine Learning*, pp. 2635–2644, 2016.

Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 7093–7101, 2017.

Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8629–8638, 2018.

Jianwen Xie, Yifei Xu, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. Generative pointnet: Energy-based learning on unordered point sets for 3d generation, reconstruction and classification. *arXiv*, pp. arXiv–2004, 2020.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5449–5458, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pp. 4800–4810, 2018.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pp. 6410–6421, 2018.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020.

Hao Yuan and Shuiwang Ji. Structpool: Structured graph pooling via conditional random fields. In *International Conference on Learning Representations*, 2020.

Chengxi Zang and Fei Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 617–626, 2020.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

Table 3: Summary and comparison of existing molecular graph generation methods.

Method	**Generative method**					**Generation process**		**Permutation invariance**	**Compositional generation**
	VAE	*GAN*	*Flow*	*RNN*	*EBM*	*One-shot*	*Sequential*		
GraphVAE (Simonovsky & Komodakis, 2018)	✓	-	-	-	-	✓	-	✗	-
DeepGMG (Li et al., 2018)	-	-	-	✓	-	-	✓	✗	-
CGVAE (Liu et al., 2018)	✓	-	-	-	-	-	✓	✗	-
MolGAN (De Cao & Kipf, 2018)	-	✓	-	-	-	✓	-	-	-
RVAE (Ma et al., 2018)	✓	-	-	-	-	✓	-	✗	-
GCPN (You et al., 2018)	-	✓	-	-	-	-	✓	✗	-
JT-VAE (Jin et al., 2018)	✓	-	-	-	-	-	✓	✗	-
MolecularRNN (Popova et al., 2019)	-	-	-	✓	-	-	✓	✗	-
GraphNVP (Madhawa et al., 2019)	-	-	✓	-	-	✓	-	✗	-
Bresson & Laurent (2019)	✓	-	-	-	-	✓	-	✗	-
GRF (Honda et al., 2019)	-	-	✓	-	-	✓	-	✗	-
GraphAF (Shi et al., 2019)	-	-	✓	-	-	-	✓	✗	-
HierVAE (Jin et al., 2020b)	✓	-	-	-	-	-	✓	✗	-
MoFlow (Zang & Wang, 2020)	-	-	✓	-	-	✓	-	✗	-
GraphCNF (Lippe & Gavves, 2020)	-	-	✓	-	-	✓	-	✓	-
GraphEBM	-	-	-	-	✓	✓	-	✓	✓

APPENDIX

A RELATED WORK

A.1 MOLECULAR GRAPH GENERATION

Since molecules can be represented as SMILES strings (Weininger, 1988), early studies generate molecules based on SMILES strings, such as CVAE (Gómez-Bombarelli et al., 2018), GVAE (Kusner et al., 2017), and SD-VAE (Dai et al., 2018). Recent studies mostly represent and generate molecules as graphs (Simonovsky & Komodakis, 2018; De Cao & Kipf, 2018; Madhawa et al., 2019). We can categorize existing molecular graph generation methods based on the underlying generative methods or the generation processes. Current molecular graph generation approaches can be grouped into four categories according to their underlying generative models, *i.e.*, VAEs, GANs, flow models, and RNNs. They can also be classified into two primary types based on their generation processes; those are, sequential generation and one-shot generation. The sequential process generates nodes and edges in a sequential order by adding nodes and edges one by one. The one-shot process generates all nodes and edges at one time.

To facilitate comparison, we summarize existing methods in Table 3. We can observe that most of them fail to satisfy an intrinsic property of graphs; that is, permutation invariance. Specifically, a generative model should yield the same likelihood for different permutations of the same graph. Currently, permutation invariance remains to be a challenging goal to achieve. The sequential generation approaches have to choose a specific order of nodes, thus failing to preserve permutation invariance. Among the one-shot methods, Bresson & Laurent (2019) also use the specific node order given by the SMILES representation. GraphVAE and RVAE perform an approximate and expensive graph matching to train the VAE model, and they cannot achieve exact permutation invariance. MolGAN circumvents this issue by using a likelihood-free method. The recent one-shot flow methods have the potential to satisfy this property. However, GraphNVP, GRF, and MoFlow cannot preserve this property since the masking strategies in the coupling layers are sensitive to node order. An exception is GraphCNF, which achieves permutation invariance by assigning likelihood independent of node ordering via categorical normalizing flows.

In this work, we propose to develop energy-based models (EBMs) (LeCun et al., 2006) for molecular graph generation. EBMs are a class of powerful methods for modeling richly structured data, but their use for graph generation has been under-explored. We show that our method can achieve the desirable property of permutation invariance. Additionally, our method has the potential to generate molecules in a compositional manner towards multiple objectives, which cannot be achieved by any existing methods.

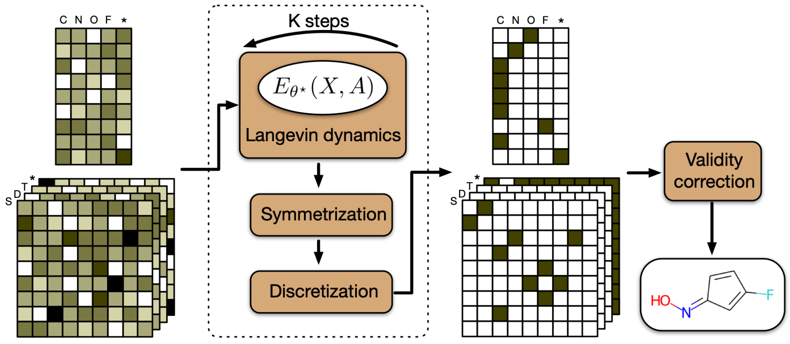

Figure 5: The generation process of our GraphEBM.

A.2 Energy-Based Models

Modeling variables by defining an unnormalized probability density has been explored for decades Hopfield (1982); Ackley et al. (1985); Cipra (1987); Dayan et al. (1995); Cipra (1987); Zhu et al. (1998); Hinton (2012). Such methods are known and unified as energy-based models (EBMs) (LeCun et al., 2006) in machine learning. EBMs capture the dependencies of variables by assigning a scalar energy to each configuration of the variables with a learnable energy function. Given a trained EBM, inference is to find the configurations that yield low energies. Training an EBM aims at obtaining an energy function where observed configurations are associated with lower energies than unobserved ones.

Currently, EBMs have been used as generative models in multiple domains, including images (Xie et al., 2015; 2016; Du & Mordatch, 2019; Du et al., 2020a;b), videos (Xie et al., 2017), 3D objects (Xie et al., 2018), and point sets (Xie et al., 2020).

To date, EBMs have rarely been studied in the graph domain. Liu et al. (2020a) attempt to generate graphs by building EBMs based on graph neural networks. Niu et al. (2020) model graphs using a score-based generative model (Song & Ermon, 2019), a method that is similar to EBMs. However, these two methods can only generate graph structures, and it is not straightforward to use them on attributed graphs. In this work, we propose GraphEBM to generate attributed molecular graphs using EBMs. Hence, we consider GraphEBM to be the first energy-based model capable of generating attributed graphs.

B Generation Process of GraphEBM

The generation process of our GraphEBM is illustrated in Figure 5.

C Existing Goal-Directed Generation Strategies

There are mainly three approaches in the literature for goal-directed generation. First, this task can be modeled as a conditional generation problem, where the property value can be utilized as the condition (Simonovsky & Komodakis, 2018). Second, for methods using the latent space, a predictor can be applied to learn the property value from the latent representation (Gómez-Bombarelli et al., 2018). Third, reinforcement learning can be used to optimize the properties of generated molecules (You et al., 2018). However, it is not straightforward to apply these methods to our GraphEBM for goal-directed generation since GraphEBM generates molecules implicitly using Langevin dynamics and no latent space exists. Also, using EBMs for generation that is conditional on continuous conditions is rarely studied by the community. Hence, it remains challenging to apply EBMs for goal-directed generation.

D Experimental Setup

Dataset. QM9 consists of 134k organic molecules and the maximum number of atoms is 9. It contains 4 atom types and 3 bond types. ZINC250k has 250k drug-like molecules and the maximum number of atoms is 38. It includes 9 atom types and 3 edge types.

Implementation details. We kekulize molecules and remove their hydrogen atoms using RDKit (Landrum et al., 2006). In our parameterized energy function, we adopt a network of $L = 3$ layers with hidden dimension $d = 64$. We use Swish as the activation function. We set $\alpha = 1$ in the loss function and the standard variance $\sigma = 0.005$ in the gaussian noise. For training, we tune the following hyperparameters: the scale t of uniform noise $\in$ [0, 1], the sample step K of Langevin dynamics $\in$ [30, 300], and the step size $\frac{\lambda}{2} \in$ [10, 50]. All models are trained for up to 20 epochs with a learning rate of 0.0001 and a batch size of 128. It is well known that it is difficult to train EBMs. We follow the techniques adopted in Du & Mordatch (2019) to stabilize the training process. Specifically, we add spectral normalization (Miyato et al., 2018) to all layers of the network. In addition, we clip the gradient used in Langevin dynamics so that its value magnitude can be less than 0.01. GraphEBM is implemented with PyTorch (Paszke et al., 2017).

Random generation. We evaluate the ability of our proposed GraphEBM to model and generate molecules. We consider most methods reviewed in Section A.1 as baselines. The following widely used metrics are adopted. *Validity* is the percentage of chemically valid molecules among all generated molecules. *Uniqueness* denotes the percentage of unique molecules among all valid molecules. *Novelty* corresponds to the percentage of generated valid molecules that are not present in the training set. The metrics are computed on $10,000$ randomly generated molecules. Results averaged over 5 runs are reported.

Goal-directed generation. To empirically show the effectiveness of our goal-directed generation method proposed in Section 2.5, we train models on ZINC250k accordingly and compare the distribution of the property score between goal-directed generated molecules and random generated molecules. We consider two chemical properties, including Quantitative Estimate of Druglikeness (QED) (Bickerton et al., 2012) and penalized logP (plogp), which is the water-octanol partition coefficient penalized by the number of long cycles and synthetic accessibility.

We further verify the effectiveness of our proposed goal-directed generation method by performing molecule optimization, including property optimization and constrained property optimization. Property optimization aims at generating novel molecules with high QED scores. We directly use the model trained for goal-directed generation and leverage the molecules in the training set as initialization for Langevin dynamics, following prior works (Jin et al., 2018; Zang & Wang, 2020). We report the highest QED scores and the corresponding novel molecules discovered by our method. For constrained property optimization, given a molecule m, our task is to obtain a new molecule m' that has a better desired chemical property with the molecular similarity $sim(m, m') \geq \delta$ for some threshold δ. We adopt Tanimoto similarity of Morgan fingerprint (Rogers & Hahn, 2010) to measure the similarity between molecules. We find that there are two different settings in baselines. JT-VAE and GCPN choose 800 molecules with the lowest plogp scores in the test set and use them as initialization, while GraphAF and MoFlow choose from the training set. We report our results on both of these two settings for extensive comparisons.

Compositional generation. As investigated in Section 2.6, our GraphEBM has the potential to conduct compositional generation towards multiple objectives. To verify this, we combine the two energy functions obtained in goal-directed generation experiments, as formulated in Eq. (14). Then we apply the generation process described in Section 2.4 to the resulting energy function to generate molecules.

E Experimental Results

The generation performance on QM9 and ZINC250k is given in Table 4 and Table 5. The generated molecule samples are visualized in Figure 6. The visualization of the implicit generation process of our GraphEBM is shown in Figure 7.

Table 4: Generation performance on QM9. The results of CVAE and GVAE are obtained from Simonovsky & Komodakis (2018). The result of MoFlow is obtained by evaluating its public trained model. All other results are from their original papers.

Method	**Validity**(%)	**Uniqueness**(%)	**Novelty**(%)
CVAE	10.30	67.50	90.00
GVAE	60.20	9.30	80.90
GraphVAE	55.70	76.00	61.60
RVAE	96.60	-	97.50
MolGAN	98.10	10.40	94.20
GraphNVP	83.10$\pm$0.50	99.20$\pm$0.30	58.20$\pm$1.90
GRF	84.50$\pm$0.70	66.00$\pm$1.14	58.60$\pm$0.82
GraphAF	100.00	94.51	88.83
MoFlow	100.00$\pm$0.00	98.53$\pm$0.14	96.04$\pm$0.10
GraphEBM	100.00$\pm$0.00	97.90$\pm$0.14	97.01$\pm$0.17

Table 5: Generation performance on ZINC250k. The results of GCPN and JT-VAE are obtained from Shi et al. (2019). The result of MoFlow is obtained by evaluating its public trained model. All other results are from their original papers.

Method	**Validity**(%)	**Uniqueness**(%)	**Novelty**(%)
GCPN	100.00	99.97	100.00
JT-VAE	100.00	100.00	100.00
MolecularRNN	100.00	99.89	100.00
GraphNVP	42.60$\pm$1.60	94.80$\pm$0.60	100.00$\pm$0.00
GRF	73.40$\pm$0.62	53.7$\pm$2.13	100.00$\pm$0.00
GraphAF	100.00	99.10	100.00
MoFlow	100.00$\pm$0.00	99.99$\pm$0.01	100.00$\pm$0.00
GraphEBM	99.96$\pm$0.02	98.79$\pm$0.15	100.00$\pm$0.00

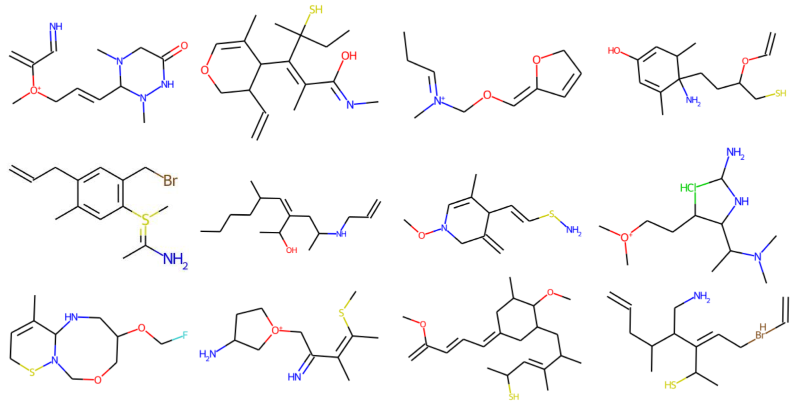

Figure 6: Visualization of molecules generated by GraphEBM.

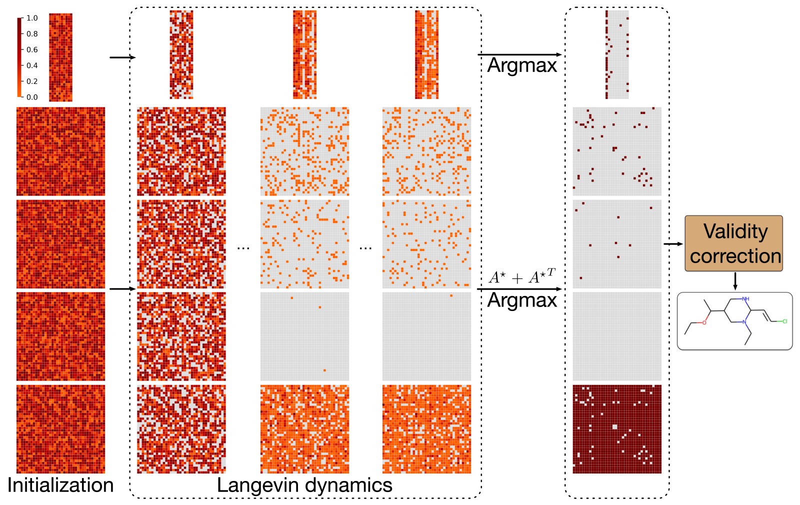

Figure 7: Visualization of the implicit generation process of our GraphEBM. The first row denotes atom matrices and the remaining rows represent fours channels of adjacency tensors, corresponding to single, double, triple, and virtual bonds. For better visual results, each atom matrix and adjacency tensor is normalized by dividing by its maximum value.