A. Generation Algorithm

Algorithm 1 Generation Algorithm of GraphDF

Input: GraphDF model, latent distribution p_{Z_a}, p_{Z_b}, maximum number of nodes n, number of node types k, number of edge types c

Initialize empty graph G_0
for $i = 1$ **to** n **do**
  $z_i \sim p_{Z_a}$
  $z_i^0 = z_i$
  $H^L = \text{R-GCN}(G_{i-1})$
  $h = \text{sum}(H^L)$
  for $d = 1$ **to** D **do**
    $\mu_i^d = \arg\max \text{MLP}_a^d(h)$
    $z_i^d = (z_i^{d-1} + \mu_i^d) \bmod k$
  end for
  $a_i = z_i^D$
  Add a new node with type a_i to G_{i-1} and set the updated graph as $G_{i-1,1}$
  for $j = 1$ **to** $i - 1$ **do**
    repeat
      $z_{ij} \sim p_{Z_b}$
      $z_{ij}^0 = z_{ij}$
      $H^L = \text{R-GCN}(G_{i-1,j})$
      $h = \text{sum}(H^L)$
      for $d = 1$ **to** D **do**
        $\mu_{ij}^d = \arg\max \text{MLP}_b^d\left(\text{Con}(h, H_i^L, H_j^L)\right)$
        $z_{ij}^d = (z_{ij}^{d-1} + \mu_{ij}^d) \bmod (c+1)$
      end for
      $b_{ij} = z_{ij}^D$
    until check_valency($G_{i-1,j}$, b_{ij}) is true
    Add a new edge with type b_{ij} connecting the node i and j to $G_{i-1,j}$ and set the updated graph as $G_{i-1,j+1}$
  end for
  if $G_{i-1,i}$ is not connected **then**
    Delete the i-th node from $G_{i-1,i}$ and set it as G_i
    Output G_i
  end if
  $G_i = G_{i-1,i}$
end for
Output G_n

B. Training Algorithm

Algorithm 2 Generation Algorithm of GraphDF

Input: Molecular graph dataset $\mathcal{M}$, GraphDF model with trainable parameter θ, latent distribution p_{Z_a}, p_{Z_b}, number of node types k, number of edge types c, learning rate α, batch size B

repeat
  Sample a batch of B molecular graphs $\mathcal{G}$ from $\mathcal{M}$
  $L = 0$
  for $G \in \mathcal{G}$ **do**
    Set n as the number of nodes in G
    Find $S_G = (a_1, a_2, b_{21}, a_3, \dots)$ by BFS on G
    for $i = 1$ **to** n **do**
      $z_i^D = a_i$
      Set G_{i-1} as the graph formed by all elements previous to a_i in S_G, or an empty graph if $i = 1$
      $H^L = \text{R-GCN}(G_{i-1})$
      $h = \text{sum}(H^L)$
      for $d = D$ **to** 1 **do**
        $\mu_i^d = \arg\max \text{MLP}_a^d(h)$
        $z_i^{d-1} = (z_i^d - \mu_i^d) \bmod k$
      end for
      $z_i = z_i^0$
      $L = L - \log p_{Z_a}(z_i)$
      for $j = 1$ **to** $i - 1$ **do**
        $z_{ij}^D = b_{ij}$
        Set $G_{i-1,j}$ as the graph formed by all elements previous to b_{ij} in S_G
        $H^L = \text{R-GCN}(G_{i-1,j})$
        $h = \text{sum}(H^L)$
        for $d = D$ **to** 1 **do**
          $\mu_{ij}^d = \arg\max \text{MLP}_b^d\left(\text{Con}(h, H_i^L, H_j^L)\right)$
          $z_{ij}^{d-1} = (z_{ij}^d - \mu_{ij}^d) \bmod (c+1)$
        end for
        $z_{ij} = z_{ij}^0$
        $L = L - \log p_{Z_b}(z_{ij})$
      end for
    end for
  end for
  $L = \frac{L}{B}$
  $\theta = \theta - \alpha \nabla_\theta L$
until θ is converged
Output GraphDF model with parameter θ

C. Data Information

Molecule datasets. For random generation of molecular graphs, we use three datasets ZINC250K (Irwin et al., 2012), QM9 (Ramakrishnan et al., 2014), and MOSES (Polykovskiy et al., 2020). ZINC250K contains around 250K drug-like molecules selected from ZINC, which is a free public chemical library for drug discovery. The maximum number of nodes among all molecules in ZINC250K is 38, and all nodes belong to 9 different types of heavy atoms. QM9 collects around 130K molecules with up to 9 heavy atoms for quantum chemistry research. MOSES provides a benchmarking platform particularly for evaluating molecule generation models, containing 1.9M molecules in total. The information about three molecule datasets are summarized below in Table 8. All molecules are transformed to kekulized form before training, that is, removing hydrogen atoms and replacing aromatic bonds by double bonds. Hence, there are three edge types in total, corresponding to single bonds, double bonds and triple bonds in molecules.

Table 8. Information of molecule datasets.

Dataset	Number of molecules	Maximum number of nodes	Number of node types
ZINC250K	249,455	38	9
QM9	133,885	9	4
MOSES	1,936,962	30	7

COMMUNITY-SMALL and EGO-SMALL. Following GNF (Liu et al., 2019), we evaluate GraphDF on two generic graph datasets, COMMUNITY-SMALL and EGO-SMALL. COMMUNITY-SMALL contains 100 synthetic 2-community graphs, and EGO-SMALL has 200 graphs which are small sub-graphs of Citeseer network dataset (Sen et al., 2008). We calculate the MMD under two cases. One is calculating MMD between the graphs in the dataset and the set of generated 1024 graph. The other is evaluating on selected graphs from generated 1024 graphs with node distribution matching. If there are N graphs in the dataset, the node distribution matching is done by first computing the distribution over node numbers in the dataset, then selecting N graphs from all generated graphs that closely match this distribution. We use the open source code of You et al. (2018b) to do evaluation.

D. Experiment Details

Random generation. On ZINC250K, QM9 and MOSES, the GraphDF model is trained with Adam optimizer for 10 epochs, where the fixed learning rate is 0.001 and the batch size is 32. On COMMUNITY-SMALL and EGO-SMALL, the GraphDF model is trained with Adam optimizer for 1000 epochs, where the fixed learning rate is 0.001 and the batch size is 16. As for generation, a widely used strategy for improving generation quality is to apply temperature parameters in prior distribution. For instance, GraphAF (Shi* et al., 2020) and MoFlow (Zang & Wang, 2020) both generate graphs by sampling from a spherical multivariate Gaussian distribution whose standard deviation is multiplied by a tunable temperature parameter t. We adopt the similar strategy in our discrete prior distribution. Specifically, for p_{Z_a} with parameters $(\alpha_0, \ldots, \alpha_{k-1})$ and p_{Z_b} with parameters $(\beta_0, \ldots, \beta_c)$, we will sample discrete latent variables as

$$\begin{aligned} p_{Z_a}(z_i = s) &= \frac{\exp(t_1 \alpha_s)}{\sum_{t=0}^{k-1} \exp(t_1 \alpha_t)}, \\ p_{Z_b}(z_{ij} = s) &= \frac{\exp(\beta_s / t_2)}{\sum_{t=0}^{c} \exp(\beta_t / t_2)}, \end{aligned} \tag{16}$$

where t_1, t_2 are tunable temperature parameters. Note that there is only one node type in COMMUNITY-SMALL and EGO-SMALL, so only t_2 is needed. The temperature parameters used for each dataset are listed below in Table 9.

Table 9. Temperature parameters for each dataset.

Dataset	t_1	t_2
ZINC250K	0.35	0.2
QM9	0.35	0.23
MOSES	0.3	0.3
COMMUNITY-SMALL	n/a	0.65
EGO-SMALL	n/a	0.5

Property optimization. The model is first pretrained on ZINC250K dataset with the same setting of random generation task for 1000 epochs. Then we apply reinforcement learning to fine-tune it for 200 iterations with a learning rate of 0.0001 and a batch size of 8 using Adam optimizer. During generation, we set the temperature parameters of prior distribution as $t_1 = 0.8, t_2 = 0.1$.

Constrained optimization. Same as property optimization, GraphDF model is first pretrained on ZINC250K dataset for 1000 epochs and fine-tuned for 200 iteration. We fine-tune the model with a learning rate of 0.0001 and a batch size of 16 using Adam optimizer. During optimization, we set the temperature parameters of prior distribution as $t_1 = 1.0, t_2 = 1.0$. Each molecule is optimized for 200 times.