Efficient Graph Generation with Graph Recurrent Attention Networks

Renjie Liao[1,2,3], Yujia Li[4], Yang Song[5], Shenlong Wang[1,2,3], Charlie Nash[4],
William L. Hamilton[6,7], David Duvenaud[1,3], Raquel Urtasun[1,2,3], Richard Zemel[1,3,8]

University of Toronto[1], Uber ATG Toronto[2], Vector Institute[3], DeepMind[4], Stanford University[5],
McGill University[6], Mila - Quebec Artificial Intelligence Institute[7], Canadian Institute for Advanced Research[8]

Generative Model of Graphs

Model the distribution of graph $G = (V, E)$:

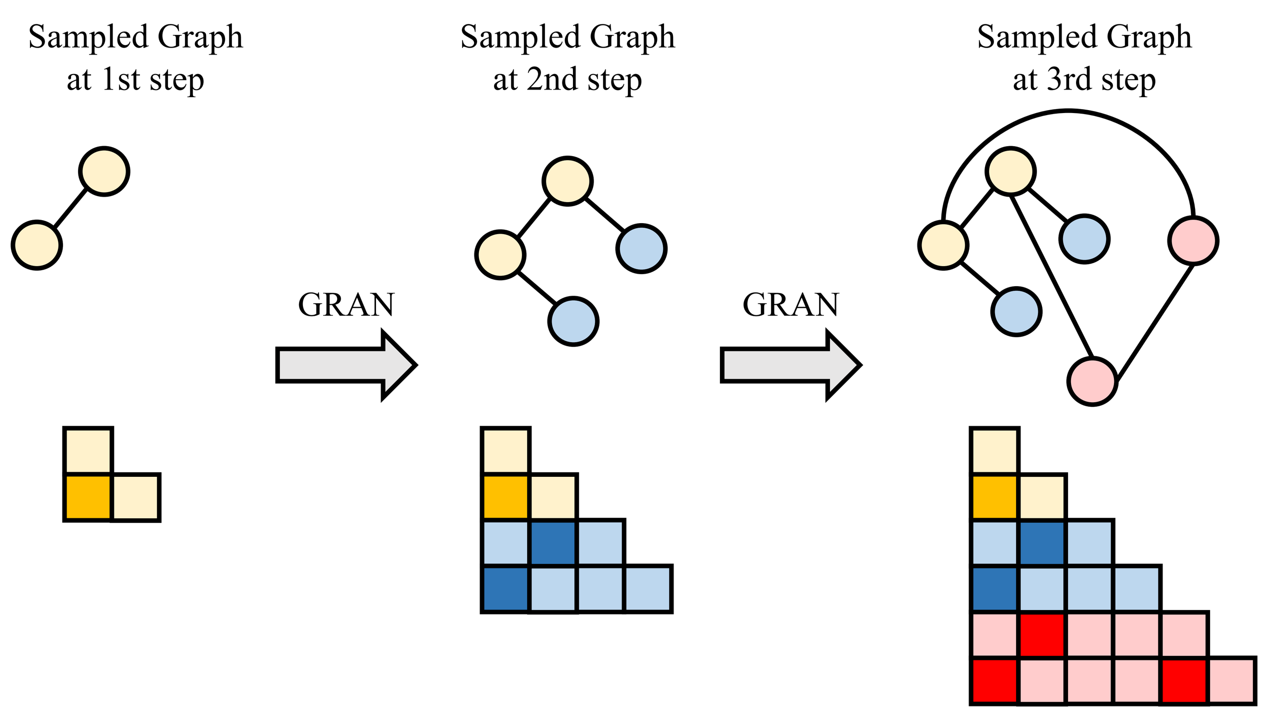
$$P(G) = \sum_{\pi} P(L^{\pi}, \pi),$$

where L^{π}: (binary) adjacency matrix
π: node ordering.

Contributions

- Our approach consists of $O(N)$ **auto-regressive** generation steps, where a block of nodes and associated edges are generated per step.
- We propose an **attention-based GNN** that better utilizes the topology of the already generated graph, reduces the dependency on the node ordering, and distinguishes multiple newly added nodes.
- We capture the correlation between multiple generated edges via a **mixture of Bernoullis output distribution** per step.
- We approximate the likelihood by marginalizing over a family of **canonical node orderings**, *e.g.*, DFS, BFS, or k-core.

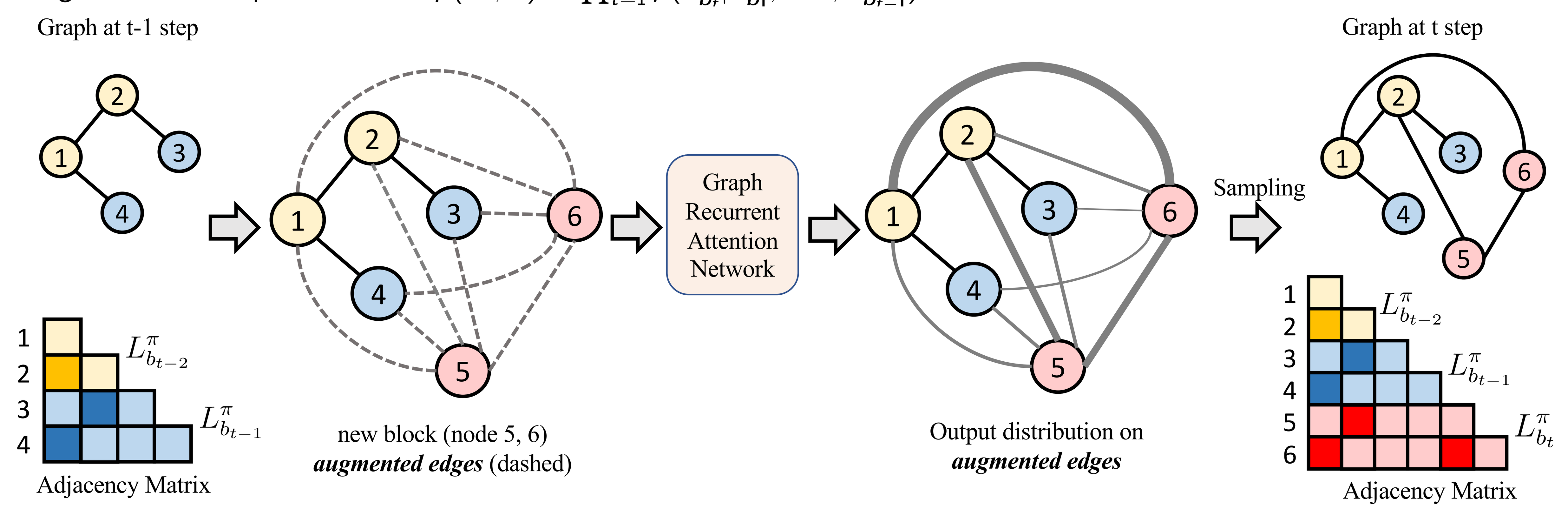
Generation Process



- Varying the block size and the sampling stride permits the **efficiency-quality trade-off**.
- Breaking the dependncy between generation steps allows **parallel training** with **sampled subgraphs**.

Graph Recurrent Attention Networks (GRAN)

Auto-regressive Decomposition: $p(L^{\pi}, \pi) = \prod_{t=1}^{T} p(L^{\pi}_{b_t} | L^{\pi}_{b_1}, \cdots, L^{\pi}_{b_{t-1}})$



- Initial Node Representation: $h^0_{b_i} = WL^{\pi}_{b_i} + b, \quad \forall i < t$
- Message Passing:

$$m^r_{ij} = f(h^r_i - h^r_j), \qquad a^r_{ij} = \text{Sigmoid}\left(g(\tilde{h}^r_i - \tilde{h}^r_j)\right),$$
$$\tilde{h}^r_i = [h^r_i, x_i], \qquad h^{r+1}_i = \text{GRU}(h^r_i, \sum\nolimits_{j \in \mathcal{N}(i)} a^r_{ij} m^r_{ij}).$$

- Output Distribution: $p(L^{\pi}_{b_t} | L^{\pi}_{b_1}, ..., L^{\pi}_{b_{t-1}}) = \sum_{k=1}^{K} \alpha_k \prod_{i \in b_t} \prod_{1 \le j \le i} \theta_{k,i,j}$

$$\alpha_1, \ldots, \alpha_K = \text{Softmax}\left(\sum\nolimits_{i \in b_t, 1 \le j \le i} \text{MLP}_{\alpha}(h^R_i - h^R_j)\right), \qquad \theta_{1,i,j}, \ldots, \theta_{K,i,j} = \text{Sigmoid}\left(\text{MLP}_{\theta}(h^R_i - h^R_j)\right)$$

Approximated Likelihood:

$$P(G) = \sum\nolimits_{\pi} P(L^{\pi}, \pi) \ge \sum\nolimits_{\pi \in \mathcal{Q}} P(L^{\pi}, \pi) \qquad e.g. \quad \mathcal{Q} = \{\pi_{\text{BFS}}, \pi_{\text{DFS}}, \pi_{\text{degree descent}}, \pi_{\text{k-core}}, \pi_{\text{default}}\}$$

Visualization

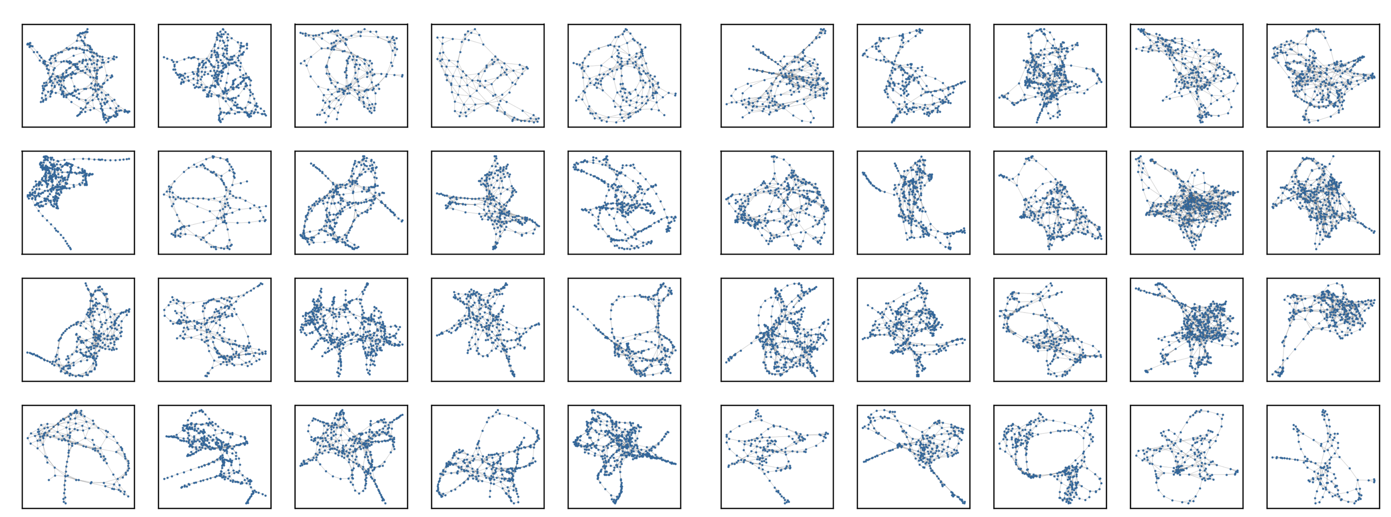

(a) Train (b) GRAN

Figure: Train and sampled graphs on the protein dataset.

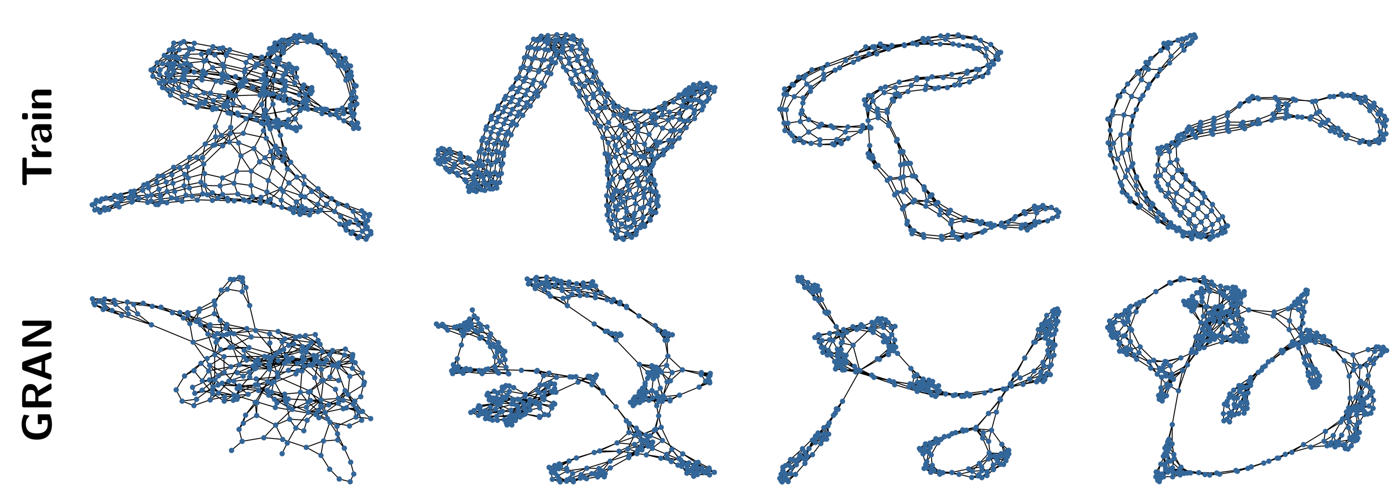


Figure: Train and sampled graphs on the 3D point cloud dataset.

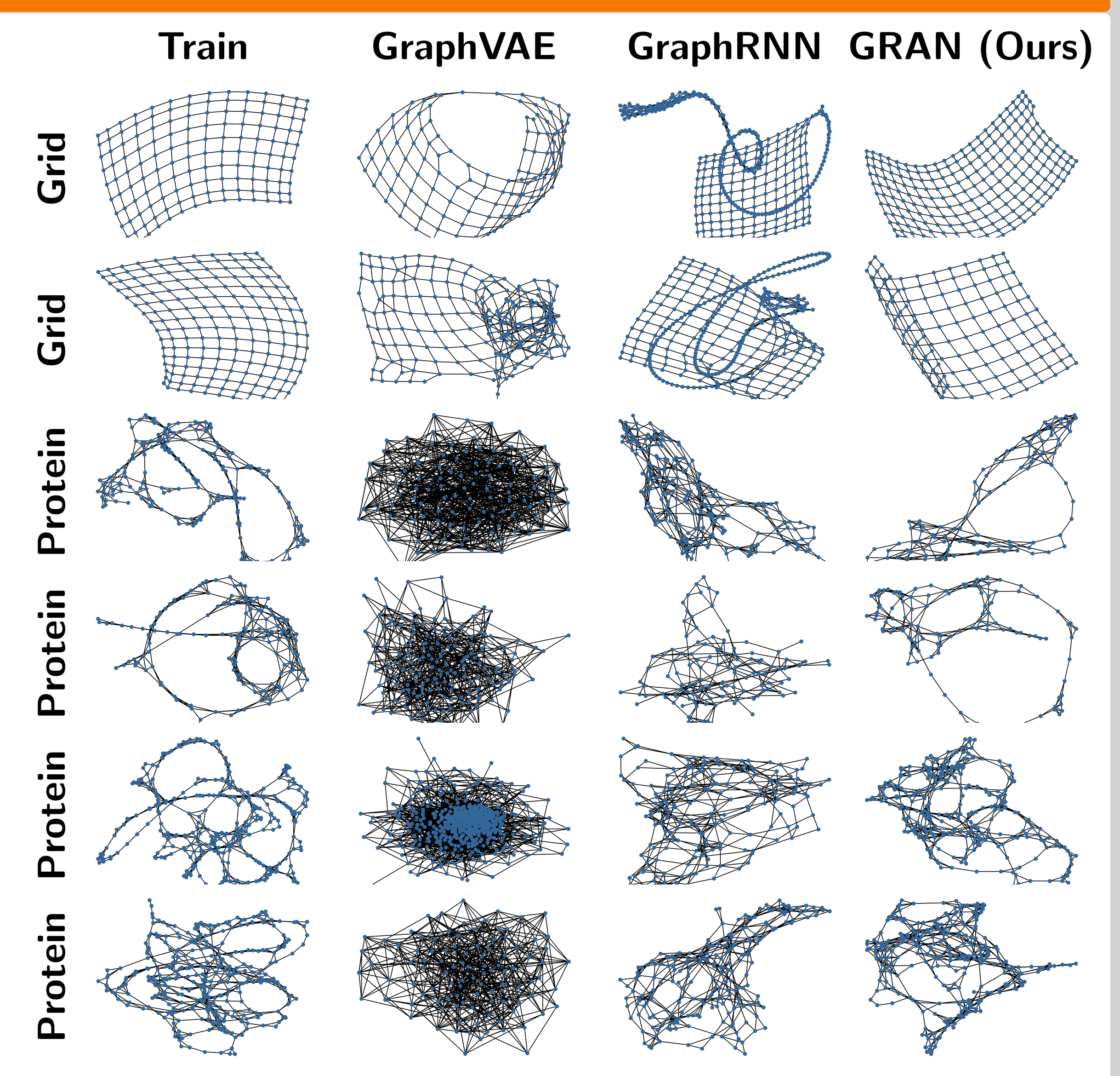


Figure: Train and sampled graphs on the grid and protein datasets.

Experiments

Dataset	$\lvert V\rvert_{avg}$	$\lvert E\rvert_{avg}$	$\lvert V\rvert_{max}$	$\lvert E\rvert_{max}$
Grid	210	392	361	684
Protein	258	646	500	1575
3D Point Cloud	1377	3074	5037	10886

Table: Dataset statistics.

Dataset	Metric	Erdos-Renyi	GraphVAE*	GraphRNN-S	GraphRNN	GRAN
Grid	Deg.	0.79	$7.07e^{-2}$	0.13	$1.12e^{-2}$	$\mathbf{8.23e^{-4}}$
	Clus.	2.00	$7.33e^{-2}$	$3.73e^{-2}$	$\mathbf{7.73e^{-5}}$	$3.79e^{-3}$
	Orbit	1.08	0.12	0.18	$\mathbf{1.03e^{-3}}$	$1.59e^{-3}$
	Spec.	0.68	$1.44e^{-2}$	0.19	$\mathbf{1.18e^{-2}}$	$1.62e^{-2}$
Protein	Deg.	$5.64e^{-2}$	0.48	$4.02e^{-2}$	$1.06e^{-2}$	$\mathbf{1.98e^{-3}}$
	Clus.	1.00	$7.14e^{-2}$	$\mathbf{4.79e^{-2}}$	0.14	$4.86e^{-2}$
	Orbit	1.54	0.74	0.23	0.88	**0.13**
	Spec.	$9.13e^{-2}$	0.11	0.21	$1.88e^{-2}$	$\mathbf{5.13e^{-3}}$
3D Point Cloud	Deg.	0.31	-	-	-	$\mathbf{1.75e^{-2}}$
	Clus.	1.22	-	-	-	**0.51**
	Orbit	1.27	-	-	-	**0.21**
	Spec.	$4.26e^{-2}$	-	-	-	$\mathbf{7.45e^{-3}}$

Table: For all MMD metrics, the smaller the better. *: our own implementation, -: not applicable due to memory issue, Deg.: degree distribution, Clus.: clustering coefficients, Orbit: the number of 4-node orbits, Spec.: spectrum of graph Laplacian.

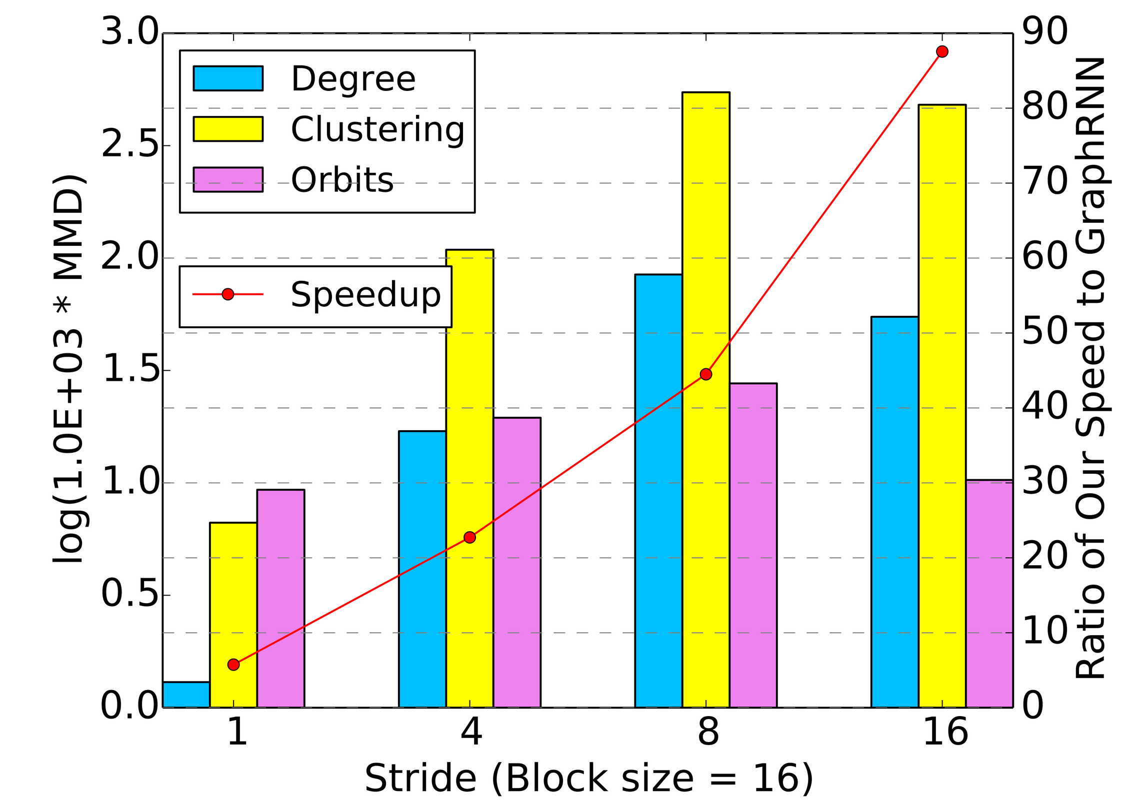


Figure: Efficiency vs. sample quality. The bar and line plots are the MMD (left y-axis) and speed ratio (right y-axis) respectively.

B	K	$\mathcal{Q}$	Deg.	Clus.	Orbit
1	1	$\{\pi_1\}$	$1.51e^{-5}$	0	$2.66e^{-5}$
1	20	$\{\pi_1\}$	$1.54e^{-5}$	0	$4.27e^{-6}$
1	50	$\{\pi_1\}$	$1.70e^{-5}$	0	$9.56e^{-7}$
1	20	$\{\pi_1, \pi_2\}$	$6.00e^{-2}$	0.16	$2.48e^{-2}$
1	20	$\{\pi_1, \pi_2, \pi_3\}$	$8.99e^{-3}$	$7.37e^{-3}$	$1.69e^{-2}$
1	20	$\{\pi_1, \pi_2, \pi_3, \pi_4\}$	$2.34e^{-2}$	$5.95e^{-2}$	$5.21e^{-2}$
1	20	$\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5\}$	$4.11e^{-4}$	$9.43e^{-3}$	$6.66e^{-4}$
4	20	$\{\pi_1\}$	$1.69e^{-4}$	0	$5.04e^{-4}$
8	20	$\{\pi_1\}$	$7.01e^{-5}$	$4.89e^{-5}$	$8.57e^{-5}$
16	20	$\{\pi_1\}$	$1.30e^{-3}$	$6.65e^{-3}$	$9.32e^{-3}$

Table: Ablation study. B: block size, K: number of Bernoulli mixtures, π_1: DFS, π_2: BFS, π_3: k-core, π_4: degree descent, π_5: default.

Code (Pytorch): `https://github.com/lrjconan/GRAN`