# PARD: Permutation-invariant AutoRegressive Diffusion

for Graph Generation

***Lingxiao Zhao***, *Xueying Ding, Leman Akoglu*

# AR vs Diffusion for Graph Generation

- Autoregressive approach
  - Simple, efficient, and fast inference
  - Permutation sensitive, sample inefficient

- Diffusion method
  - Permutation invariant, sample efficient
  - Slow inference with ~1000 steps, requiring additional domain-specific features (DiGress)

# How to Get the Full Benefits of Both?

- ???
  - Simple, efficient, and fast inference
  - Permutation invariant, sample efficient

**PARD**

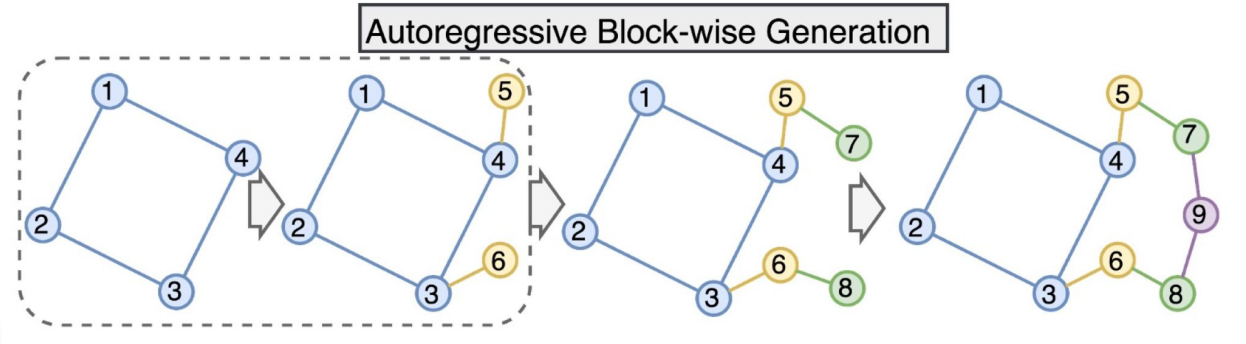
# From AR to PARD

- Why is AR permutation sensitive?
  - Finding unique node/edge ordering is impossible (graph canonization, NP-intermediate)
    Vikraman Arvind, Bireswar Das, and Johannes Köbler. The space complexity of k-tree isomorphism. Springer, 2007.
  - Non-unique, non-deterministic

- However, graphs are not set, as nodes are not completely unordered.
  - Nodes are not “fully equivalent” considering edges.

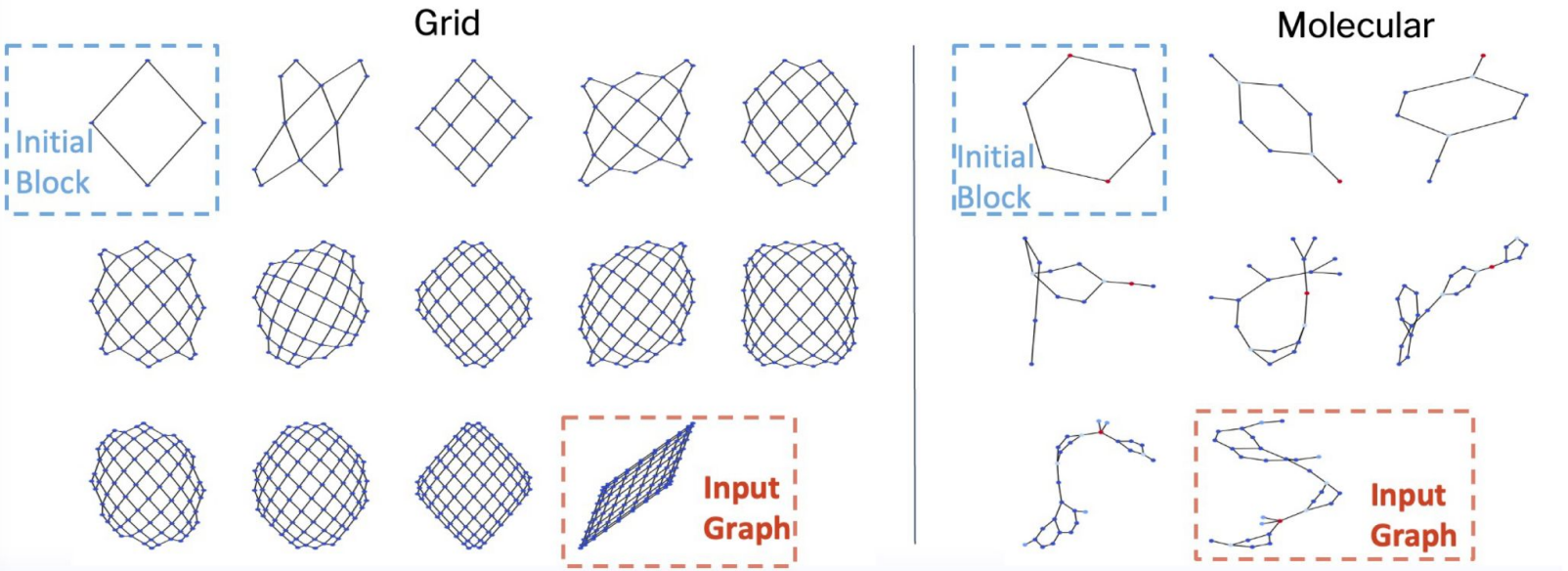
# Permutation-equivariant Partial Order

- No unique node order
- Unique **partial** order can be easily defined!
  - **Why partial**: structurally-equivalent nodes MUST have the same rank/order.
- **Structural partial ordering**: sequence of blocks
  - **Unique, deterministic, permutation-equivariant.**



# The Structural Partial Order

**Thm.** *For any $G$, structural partial order $\phi$ is deterministic, unique, and permutation equivariant:* $\phi(\boldsymbol{P} \star \mathrm{G}) = \boldsymbol{P} \star \phi(\mathrm{G})$



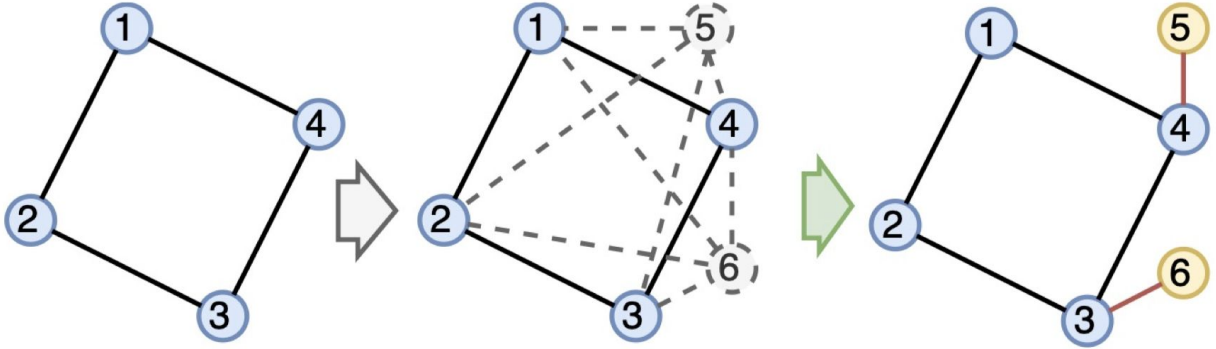
# Blockwise Autoregression

- Let block $\mathcal{B}_j = \{i \in \mathcal{V}(G) | \phi(i) = j\}$ and $\mathcal{B}_{1:i} := \cup_{j=1}^{i} \mathcal{B}_j$, can decompose the joint distribution of G with AR

$$p_\theta(\mathbf{G}) = \prod_{i=1}^{K_B} p_\theta\Big(\underbrace{\mathbf{G}[\mathcal{B}_{1:i}] \setminus \mathbf{G}[\mathcal{B}_{1:i-1}]}_{\text{All nodes and edges in block i, not in blocks 1…(i–1)}} \Big| \mathbf{G}[\mathcal{B}_{1:i-1}]\Big)$$

  - Each block's conditional distribution is easier to model
  - Permutation invariant under certain condition of block's conditional distribution

# Modeling Block Conditional Distribution




$G[\mathcal{B}_{1:i}]$ $G\left[\mathcal{B}_{1:i}, |\mathcal{B}_{1:i+1}|\right]$ $G[\mathcal{B}_{1:i+1}]$

$$p_\theta\left(\mathrm{G}[\mathcal{B}_{1:i}] \setminus \mathrm{G}[\mathcal{B}_{1:i-1}] \,\middle|\, \mathrm{G}[\mathcal{B}_{1:i-1}]\right)$$

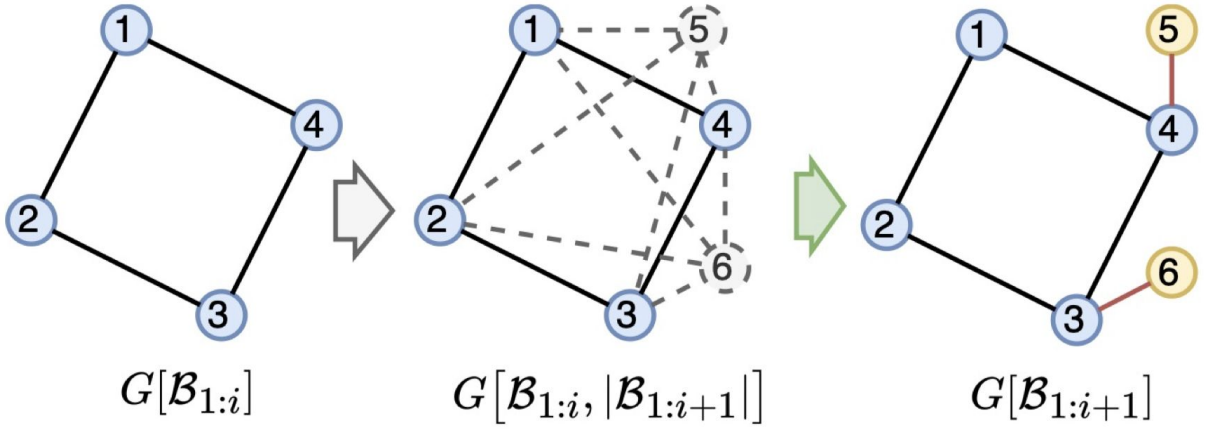
Next block's size

$$p_\theta\left(|\mathcal{B}_i| \,\middle|\, \mathrm{G}[\mathcal{B}_{1:i-1}]\right) \prod_{\mathbf{x} \in \mathrm{G}[\mathcal{B}_{1:i}] \setminus \mathrm{G}[\mathcal{B}_{1:i-1}]} p_\theta\left(\mathbf{x} \,\middle|\, \mathrm{G}[\mathcal{B}_{1:i-1}] \cup \emptyset[\mathcal{B}_{1:i}]\right)$$

Nodes & edges added by next block

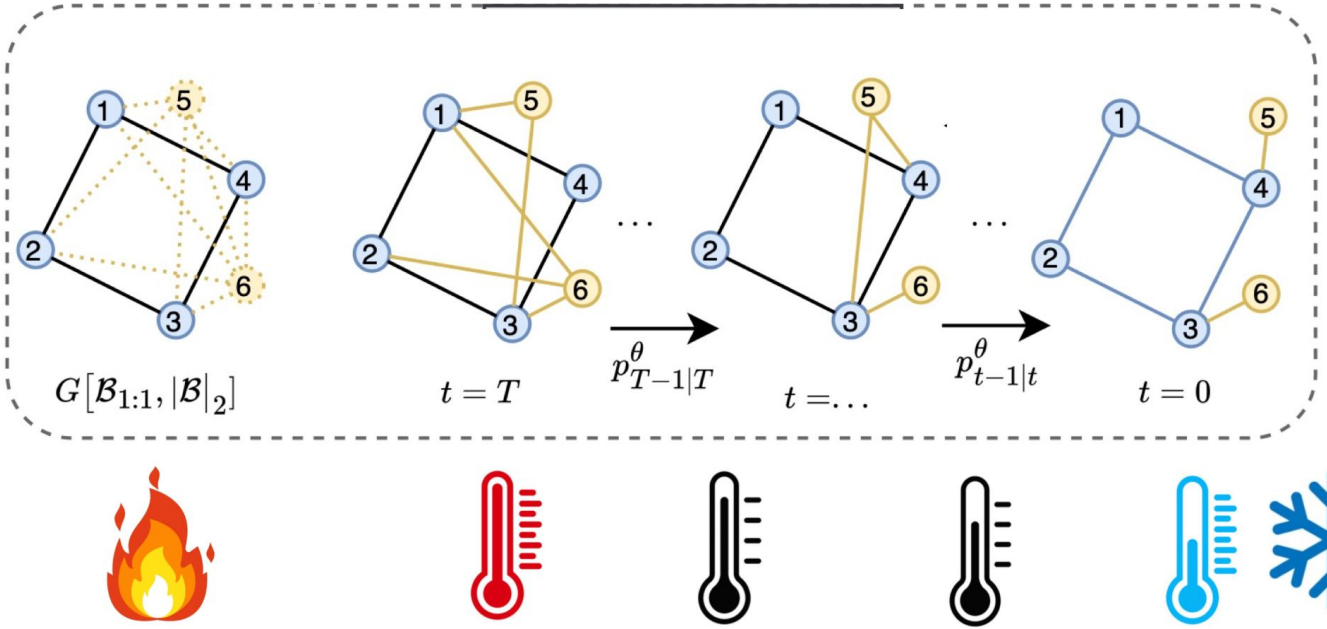
# Issue

- **Thm.:** next block prediction cannot be solved directly with ***any*** equivariant GNN
  - due to structural equivalences: all dashed edges (middle) will receive SAME prediction

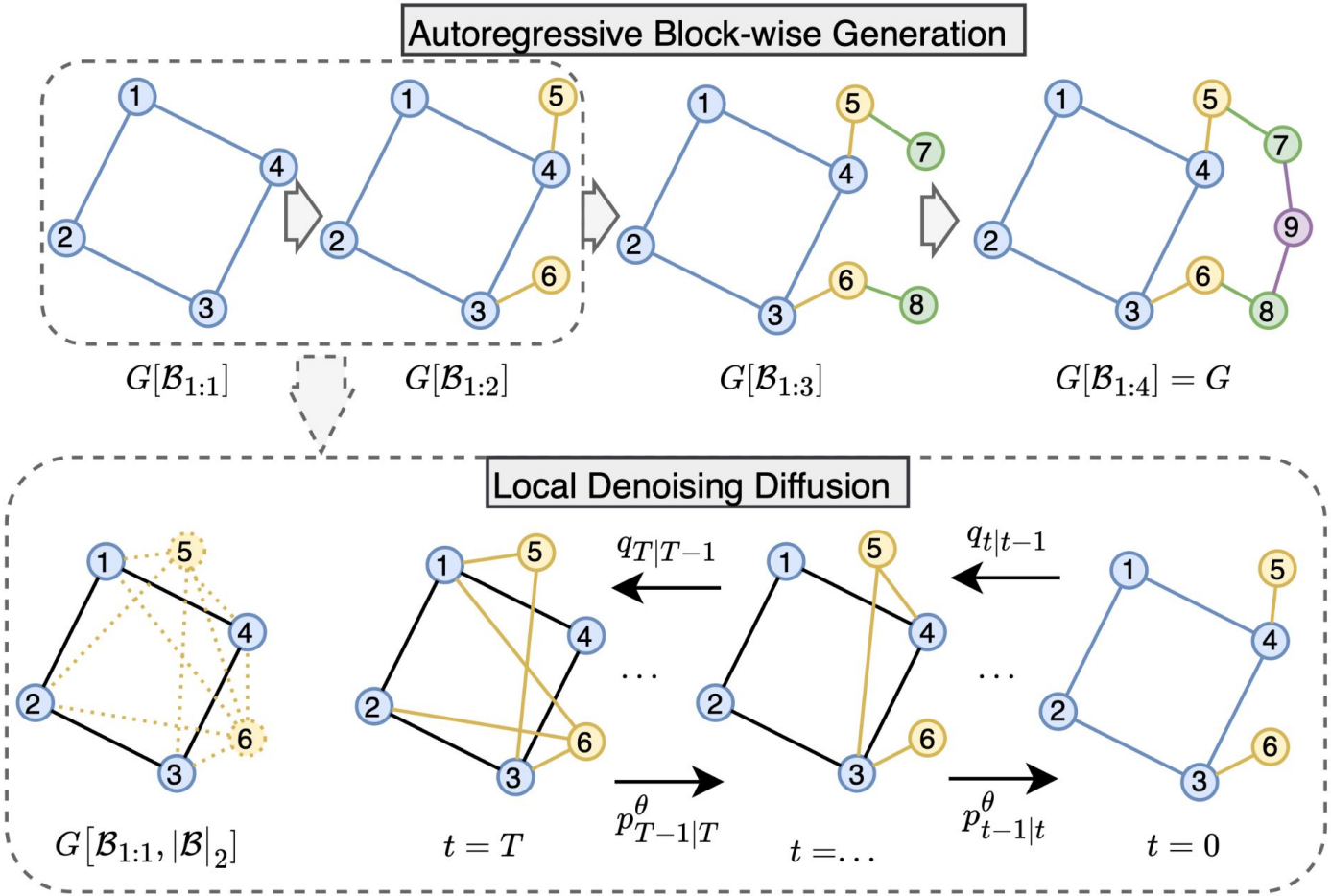


# Solution: Annealing / Diffusion

To “increase energy”, **add random noise** to nodes edges (**heat**), then **reduce noise** (**cool**) to attain desired target



# PARD


Autoregressive Block-wise Generation
$G[\mathcal{B}_{1:1}]$
$G[\mathcal{B}_{1:2}]$
$G[\mathcal{B}_{1:3}]$
$G[\mathcal{B}_{1:4}] = G$
Local Denoising Diffusion
$G[\mathcal{B}_{1:1}, |\mathcal{B}|_2]$
$t = T$
$t = \ldots$
$t = 0$
$q_{T|T-1}$
$q_{t|t-1}$
$p^{\theta}_{T-1|T}$
$p^{\theta}_{t-1|t}$

# Important Techniques for Training PARD

- Efficient and expressive architecture
  - PPGN + transformer (reduce memory cost)

- ***Parallel*** training of diffusion for **ALL** blocks
  - Similar as causal transformer
  - Non-trivial (for PPGN). See paper for the detail.

- Simple discrete diffusion framework
  - Zhao, Lingxiao, et al. "Improving and Unifying Discrete&Continuous-time Discrete Denoising Diffusion." *arXiv preprint arXiv:2402.03701* (2024).

# Experiments: SoTA

Table 1: Generation quality on QM9 with explict hydrogens.

| Model | Valid. ↑ | Uni. ↑ | Atom.↑ | Mol. ↑ |
|---|---|---|---|---|
| Dataset (optimal) | 97.8 | 100 | 98.5 | 87.0 |
| ConGress | 86.7 | **98.4** | 97.2 | 69.5 |
| DiGress (uniform) | 89.8 | 97.8 | 97.3 | 70.5 |
| DiGress (marginal) | 92.3 | 97.9 | 97.3 | 66.8 |
| DiGress (marg. + *feat.*) | 95.4 | 97.6 | 98.1 | 79.8 |
| PARD (*no feat.*) | **97.5** | 95.8 | **98.4** | **86.1** |

Table 2: Generation quality on ZINC250K.

| Model | Validity ↑ | FCD ↓ | Uni. ↑ | Model Size |
|---|---|---|---|---|
| EDP-GNN | 82.97 | 16.74 | 99.79 | 0.09M |
| GraphEBM | 5.29 | 35.47 | 98.79 | - |
| SPECTRE | 90.20 | 18.44 | 67.05 | - |
| GDSS | **97.01** | 14.66 | 99.64 | 0.37M |
| GraphArm | 88.23 | 16.26 | 99.46 | - |
| DiGress | 91.02 | 23.06 | 81.23 | 18.43M |
| SwinGNN-L | 90.68 | 1.99 | 99.73 | 35.91M |
| PARD | 95.23 | **1.98** | **99.99** | 4.1M |

Generation quality on MOSES. The top three methods use hard-coded rules

| Model | Val. ↑ | Uni. ↑ | Novel. ↑ | Filters ↑ | FCD ↓ | SNN ↑ | Scaf. ↑ |
|---|---|---|---|---|---|---|---|
| VAE | 97.7 | 99.8 | 69.5 | 99.7 | 0.57 | 0.58 | 5.9 |
| JT-VAE | 100 | 100 | 99.9 | 97.8 | 1.00 | 0.53 | 10.0 |
| GraphINVENT | 96.4 | 99.8 | - | 95.0 | 1.22 | 0.54 | 12.7 |
| ConGress | 83.4 | 99.9 | **96.4** | 94.8 | 1.48 | 0.50 | **16.4** |
| DiGress | 85.7 | **100** | 95.0 | 97.1 | 1.19 | 0.52 | 14.8 |
| PARD | **86.8** | **100** | 78.2 | **99.0** | **1.00** | **0.56** | 2.2 |

# Ablation

Table 5: Ablation study on QM9 with varying maximum hops while keeping the total diffusion steps fixed (first two parts). The last part examines the effect of increasing steps for the no AR case.

| **Setting** | **No AR** | **With AR** | | **No AR, ↑ steps** | |
|---|---|---|---|---|---|
| Total diffusion steps | 140 | 140 | | 280 | 490 |
| Maximum hops | 0 | 1 | 2 | 3 | 0 | 0 |
| Average number of blocks | 1 | 4.3 | 5.6 | 7.75 | 1 | 1 |
| Diffusion steps per block | 140 | 32 | 25 | 20 | 280 | 490 |
| Validity | 93.8 | **97.1** | 96.7 | 97.0 | 94.3 | 95.2 |
| Uniqueness | **96.9** | 96.5 | 96.2 | 96.1 | 96.5 | 96.9 |
| Mol stability | 76.4 | 86.1 | 85.4 | **86.3** | 79.3 | 79.2 |
| Atom Stability | 97.7 | 98.3 | 98.3 | **98.4** | 97.9 | 98.0 |

# Thank You

**Code**: https://github.com/LingxiaoShawn/Pard