FLOYDNET: PIVOTAL ATTENTION FOR RELATION COMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning algorithmic computation often requires explicit relational intermediate states, yet many graph processors maintain their primary states on individual entities. We introduce FloydNet and **Pivotal Attention** (PA), which maintain ordered pair states and update a target relation (i, k) by attending over candidates formed from (i, j) and (j, k) for every pivot j. Motivated by the pair-and-pivot structure of Floyd–Warshall, PA learns relation composition and pivot weighting in parallel rather than executing its ordered min-plus recurrence. The k-FloydNet framework extends this operation to ordered k-tuples, with Self-Attention and PA as its $k = 1$ and $k = 2$ cases at the attention-operation level. Under atomic tuple initialization and invariant readout, we show that k-FloydNet is no more graph-discriminative than k-FWL; on BREC, each evaluated variant matches the success set of its corresponding WL reference. FloydNet further achieves 96.64% mean accuracy under the reported CLRS-30 protocol and a 99.8% optimality rate with 10 samples on held-out non-metric TSP instances.

1 INTRODUCTION

Neural Algorithmic Reasoning asks whether neural networks can learn representations of algorithmic computation that transfer beyond the instances seen during training (Veličković & Blundell, 2021; Veličković et al., 2022). A central challenge is the choice of intermediate state: a processor must represent not only the input entities, but also the subproblems and partial results created during computation.

Most graph processors keep their primary states on individual nodes. Local message passing propagates these states along graph edges, while Graph Transformers allow global interactions among them (Gilmer et al., 2017; Xu et al., 2019; Dwivedi & Bresson, 2021; Ying et al., 2021). Both are effective designs, but global communication alone does not make relations explicit computational objects. Many algorithms are naturally expressed through pair- or tuple-indexed states: shortest-path algorithms maintain pairwise distances, geometric systems compose frame transformations, and combinatorial solvers make pairwise decisions. Encoding these states only through node representations leaves the relevant relations implicit.

Explicit relation states also require an update rule that reflects how they interact. In many algorithms, a relation from i to j and a relation from j to k jointly inform the relation from i to k. Treating all pair states as unrelated tokens does not expose this shared-pivot structure. This motivates an operator whose candidates are relations that can be composed through an intermediate entity.

We introduce **Pivotal Attention** (PA). FloydNet maintains an ordered pair state $\boldsymbol{R}_{ik}$. For each pivot j, PA forms a candidate from $\boldsymbol{R}_{ij}$ and $\boldsymbol{R}_{jk}$, then uses $\boldsymbol{R}_{ik}$ to weight and aggregate these candidates. Self-Attention attends over entity states; PA attends over candidate relation compositions. Pair-state and triangle-indexed architectures provide important precedents (Jumper et al., 2021; Abramson et al., 2024; Bergen et al., 2021); PA differs in making both composable relations part of each candidate key and value.

The abstraction is captured by k-FloydNet. Its states are indexed by ordered k-tuples, and a common pivot generates one candidate by replacing each tuple coordinate in turn. At the level of state space and attention operation, $k = 1$ gives entity-state Self-Attention, $k = 2$ gives pair-state Pivotal Attention, and $k > 2$ represents higher-order relations. This unifies their attention operations, not the complete architectures.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Our design is inspired by dynamic programming (DP), which organizes computation through recurrences over reusable subproblem states (Bellman, 1966). Floyd–Warshall and its earlier closure lineage provide a canonical example of all-pairs states composed through intermediate vertices (Roy, 1959; Floyd, 1962; Warshall, 1962):

$$D_{ij}^{(k)} = \min\Big\{ D_{ij}^{(k-1)},\ D_{ik}^{(k-1)} + D_{kj}^{(k-1)} \Big\}. \tag{1}$$

Here stage k admits vertex k as an additional intermediate. FloydNet uses this pair-and-pivot pattern but not the ordered min-plus recurrence: PA learns the composition and pivot weights and evaluates all candidates in parallel.

The common-pivot construction also mirrors k-FWL. Under atomic tuple initialization and invariant readout, k-FloydNet is upper-bounded by k-FWL; on BREC, the evaluated variants match their WL references' success sets under RPC. We further evaluate algorithmic computation, pair-structured optimization, and graph prediction.

Our primary contributions are:

1. We introduce Pivotal Attention, in which each target pair relation attends over pivot-indexed compositions of aligned incoming and outgoing relations.
2. We express PA as the $k = 2$ case of a common-pivot ordered-tuple attention framework that includes entity-state Self-Attention at $k = 1$, and provide a fused pairwise implementation that avoids materializing cubic pivot-message tensors.
3. We prove a k-FWL upper bound under explicit initialization and readout assumptions and evaluate the relation-first inductive bias through BREC, CLRS-30, TSP, and graph prediction benchmarks.

2 The FloydNet Framework

2.1 Relational State Spaces for Attention

Attention refines a target by comparing it with candidate representations. Standard Self-Attention (SA) operates on entity states $\mathbf{H}_i$, using projected entity keys and values. FloydNet instead maintains an ordered pair state $\boldsymbol{R}_{ik} \in \mathbb{R}^{d_r}$ and refines this relation from candidates composed through intermediate entities.

2.2 Pivotal Attention

For a target relation $\boldsymbol{R}_{ik}$, every entity j aligns two composable relations, $\boldsymbol{R}_{ij}$ and $\boldsymbol{R}_{jk}$. Their shared endpoint j is the *pivot*. Pivotal Attention (PA) asks how the relation composed through each pivot should refine the target relation.

Let there be h heads of dimension d_h, with $d_r = hd_h$. For head $a \in [h]$, define

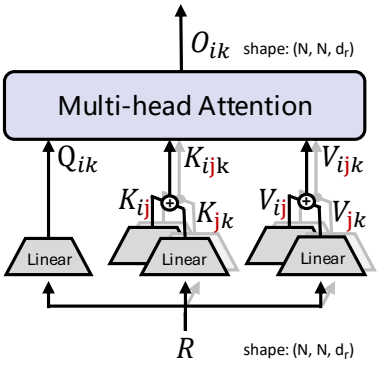
$$\begin{aligned} \mathbf{q}_{ik}^a &= W_Q^a \boldsymbol{R}_{ik}, \\ \boldsymbol{\kappa}_{ijk}^a &= W_{K,L}^a \boldsymbol{R}_{ij} + W_{K,R}^a \boldsymbol{R}_{jk}, \\ \boldsymbol{\nu}_{ijk}^a &= W_{V,L}^a \boldsymbol{R}_{ij} + W_{V,R}^a \boldsymbol{R}_{jk}. \end{aligned} \tag{2}$$

The role-specific projections distinguish the incoming relation (i, j) from the outgoing relation (j, k). The key $\boldsymbol{\kappa}_{ijk}^a$ scores the composition against the target; the value $\boldsymbol{\nu}_{ijk}^a$ supplies its update. PA then normalizes over pivots:

$$\alpha_{ijk}^a = \frac{\exp\Big(\langle \mathbf{q}_{ik}^a, \boldsymbol{\kappa}_{ijk}^a \rangle / \sqrt{d_h}\Big)}{\sum_{p \in V} \exp\Big(\langle \mathbf{q}_{ik}^a, \boldsymbol{\kappa}_{ipk}^a \rangle / \sqrt{d_h}\Big)}, \qquad \mathbf{o}_{ik}^a = \sum_{j \in V} \alpha_{ijk}^a \boldsymbol{\nu}_{ijk}^a, \tag{3}$$

$$\mathrm{PA}(\boldsymbol{R})_{ik} = W_O[\mathbf{o}_{ik}^1 \| \cdots \| \mathbf{o}_{ik}^h].$$

Thus SA learns entity-to-entity interactions, whereas PA learns interactions between a target relation and pivot-aligned compositions of other relations. Both use query–key compatibility and value aggregation; they differ in the state they represent and in how they form candidates. The main model




```
def PivotAttnCore(q_ik, k_ij, k_jk, v_ij, v_jk):
    # Inputs: [batch, n, n, heads, head_dim]
    k_ijk = (k_ij[:, :, :, None, :, :] +
             k_jk[:, None, :, :, :, :])
    v_ijk = (v_ij[:, :, :, None, :, :] +
             v_jk[:, None, :, :, :, :])
    q_ixk = q_ik[:, :, None, :, :, :]

    a_ijk = torch.sum(q_ixk * k_ijk, dim=-1)
    a_ijk = a_ijk / (q_ik.shape[-1] ** 0.5)
    w_ijk = torch.softmax(a_ijk, dim=-3) # pivot
        j
    o_ijk = v_ijk * w_ijk[..., None]
    return torch.sum(o_ijk, dim=-4) # pivot j
```

Figure 1: **Pivotal Attention refines a relation through pivot-specific candidates.** To update (i, k), each pivot j provides a candidate built jointly from (i, j) and (j, k). The target state $\boldsymbol{R}_{ik}$ supplies the query used to weight and aggregate these candidates.

uses the additive composition in Equation (2). Other composition operators, including elementwise multiplication, are discussed in Appendix B.2.

PA follows Floyd–Warshall's pattern of composing pair states through an intermediate vertex (Floyd, 1962; Warshall, 1962), but learns the composition and evaluates all pivots in parallel.

2.3 The Unified k-FloydNet Framework

The same operation extends to relations of any order. For $k \geq 1$, let $\mathbf{v} = (v_1, \ldots, v_k) \in V^k$ be an ordered tuple, with repeated vertices allowed, and let $\boldsymbol{R}_{\mathbf{v}}$ denote its state. For coordinate $r \in [k]$ and pivot $p \in V$, define

$$\mathbf{v}[r \leftarrow p] = (v_1, \ldots, v_{r-1}, p, v_{r+1}, \ldots, v_k).$$

A common pivot produces the k coordinate-replacement states $\big(\boldsymbol{R}_{\mathbf{v}[1\leftarrow p]}, \ldots, \boldsymbol{R}_{\mathbf{v}[k\leftarrow p]}\big)$. For head a, k-FloydNet constructs the additive pivot message

$$\mathbf{q}^a_{\mathbf{v}} = W^a_Q \boldsymbol{R}_{\mathbf{v}},$$

$$\boldsymbol{\kappa}^a_{\mathbf{v},p} = \sum_{r=1}^{k} W^a_{K,r} \boldsymbol{R}_{\mathbf{v}[r\leftarrow p]}, \qquad \boldsymbol{\nu}^a_{\mathbf{v},p} = \sum_{r=1}^{k} W^a_{V,r} \boldsymbol{R}_{\mathbf{v}[r\leftarrow p]}, \tag{4}$$

$$\alpha^a_{\mathbf{v},p} = \frac{\exp\big(\langle \mathbf{q}^a_{\mathbf{v}}, \boldsymbol{\kappa}^a_{\mathbf{v},p}\rangle/\sqrt{d_h}\big)}{\sum_{u\in V} \exp\big(\langle \mathbf{q}^a_{\mathbf{v}}, \boldsymbol{\kappa}^a_{\mathbf{v},u}\rangle/\sqrt{d_h}\big)}, \qquad \mathbf{o}^a_{\mathbf{v}} = \sum_{p\in V} \alpha^a_{\mathbf{v},p} \boldsymbol{\nu}^a_{\mathbf{v},p}. \tag{5}$$

Coordinate-specific maps preserve tuple roles. For $k = 1$, the replacement state is simply $\boldsymbol{R}_p$, recovering node-set SA at the operation level; for $k = 2$, the replacements (p, k) and (i, p) recover PA after assigning its right and left roles; and $k > 2$ represents explicit higher-order relations. This is a state-space and operation-level unification, not an identity between complete architectures. Increasing k enlarges the available relational state space; its formal k-FWL placement is summarized below and proved in Appendix D.

2.4 Pairwise Instantiation and Properties

For a graph $G = (V, E, \mathbf{X}, \mathbf{E}, \mathbf{G})$, FloydNet initializes each ordered pair with a shared map

$$\boldsymbol{R}^{(0)}_{ij} = \mathrm{Init}(\mathbf{G}, \mathbf{X}_i, \mathbf{X}_j, \mathbf{E}_{ij}, \mathbf{1}\{i = j\}), \tag{6}$$

where pair features distinguish edges, nonedges, and self-pairs when relevant. A FloydBlock applies PA and a pointwise FFN in a pre-normalized residual form:

$$\widetilde{\boldsymbol{R}} = \boldsymbol{R} + \mathrm{PA}(\mathrm{Norm}(\boldsymbol{R})), \qquad \boldsymbol{R}' = \widetilde{\boldsymbol{R}} + \mathrm{FFN}(\mathrm{Norm}(\widetilde{\boldsymbol{R}})). \tag{7}$$

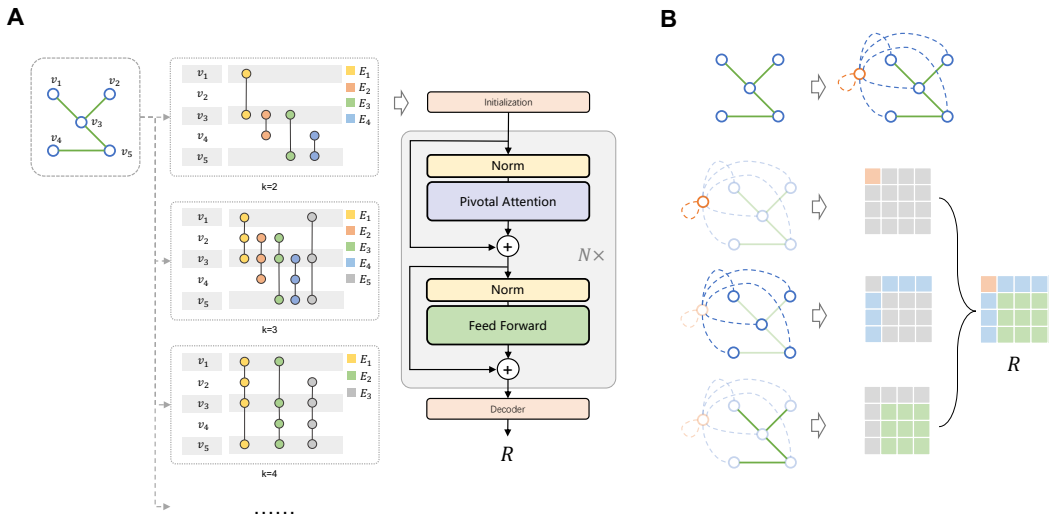


Figure 2: **A unified hierarchy of relational state spaces. (A)** k-FloydNet lifts attention from entity states ($k = 1$) to pair relations ($k = 2$) and higher-order ordered-tuple relations ($k > 2$). **(B)** FloydNet instantiates the pairwise case with stacked FloydBlocks and optional SuperNode readouts.

Pair tasks decode $\boldsymbol{R}_{ij}^{(L)}$ directly. For node and graph tasks, a distinguished SuperNode s may provide readouts from $\boldsymbol{R}_{is}^{(L)}$ and $\boldsymbol{R}_{ss}^{(L)}$. Shared tuple maps and symmetric pivot aggregation make tuple updates permutation equivariant and the fixed-SuperNode graph readout invariant (Proposition 1). Pairwise PA costs $\mathcal{O}(N^3 d_r + N^2 d_r^2)$ time per block; the fused kernel avoids materializing cubic pivot messages and uses $\mathcal{O}(N^2 d_r)$ forward intermediate storage. Full architectural definitions, implementation details, and the general-k cost are given in Appendix B.

Theoretical placement. The common-pivot neighborhood construction above closely mirrors folklore k-FWL. Under attributed atomic tuple initialization, shared tuple maps, permutation-invariant pivot aggregation, and invariant graph readout, equal k-FWL colors imply equal k-FloydNet states layer by layer; hence $k\text{-}FloydNet \preceq$ k-FWL. In particular, $FloydNet \preceq$ 2-FWL, equivalently ordinary 3-WL under this convention. On BREC, the evaluated variants match the corresponding WL reference success sets. The proof and its scope appear in Appendix D.

3 Experiments

We evaluate structural discrimination, algorithmic generalization, TSP optimization, and transfer to graph benchmarks.

Our code, data-generation scripts, and exact hyperparameters are available at `https://anonymous.4open.science/r/floydnet-anonymization/`. Where applicable (e.g., CLRS), we use the official evaluation harness without hints at test time unless explicitly stated. For TSP, train and test sets have disjoint size ranges and seeds; we report the full benchmark and a filtered subset that reduces multiple-optimum cases.

3.1 Relation-State Discrimination and Pair-Level Concordance

We first test whether explicit pair and tuple states support structural counting and graph-pair discrimination.

Quantitative Expressiveness on Homomorphism Counting Tasks. We use the synthetic counting tasks of Zhang et al. (2024a). These tasks provide an empirical quantitative probe beyond standard 1-WL-limited message passing. As shown in Tables 1 and 3, FloydNet obtains substantially

Table 1: MAE on the evaluated homomorphism-counting tasks. FloydNet obtains near-zero MAE on the reported tasks.

Model \ Task	Graph-level			Node-level		Edge-level		
MPNN (GIN)	.300	.233	.254	.505	.478	-	-	-
Subgraph GNN	.011	.015	.012	.004	.058	.003	.058	.048
Local 2-GNN	.008	.008	.010	.003	.004	.005	.006	.008
Local 2-FGNN	.003	.005	.004	.005	.005	.007	.007	.008
FloydNet	.000	.001	.001	.000	.000	.000	.000	.000

Table 2: Accuracy on BREC. Pairwise agreement with each variant's corresponding WL reference is reported in the supplementary per-pair results.

Model / Heuristic	Accuracy (%)
MPNN (1-WL equiv.)	<20%
Graphormer	19.8%
PPGT	58.5%
KP-GNN	68.8%
3-WL(2-FWL) Test	**67.5%**
FloydNet	**67.5%**
3-FloydNet	**95.0%**
4-FloydNet	**99.8%**

Table 3: MAE on the evaluated (chordal) cycle-counting tasks. FloydNet obtains near-zero MAE on the reported tasks.

Model \ Task	Graph-level						Node-level						Edge-level					
MPNN	.358	.208	.188	.146	.261	.205	.600	.413	.300	.207	.318	.237	-	-	-	-	-	-
Subgraph GNN	.010	.020	.024	.046	.007	.027	.003	.005	.092	.082	.050	.073	.001	.003	.090	.096	.038	.065
Local 2-GNN	.008	.011	.017	.034	.007	.016	.002	.005	.010	.023	.004	.015	.001	.005	.010	.019	.005	.014
Local 2-FGNN	.003	.004	.010	.020	.003	.010	.004	.006	.012	.021	.004	.014	.003	.006	.012	.022	.005	.012
FloydNet	.000	.000	.001	.001	.000	.001	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

lower Mean Absolute Error than the displayed MPNN baseline across the reported graph-, node-, and edge-level tasks.

Qualitative Expressiveness on Isomorphism Testing. The BREC benchmark (Wang & Zhang, 2024) provides 400 pairs of non-isomorphic graphs specifically designed to be indistinguishable by the 1-WL test; a model's accuracy reflects how many pairs it can correctly tell apart. As shown in Table 2, under its Reliable Paired Comparisons (RPC) protocol, FloydNet distinguishes exactly the same graph pairs as the 3-WL reference. This pair-level concordance is stronger than aggregate-score agreement and matches the 3-WL reference success set on BREC, without implying equivalence on arbitrary graphs (Section 2.4).

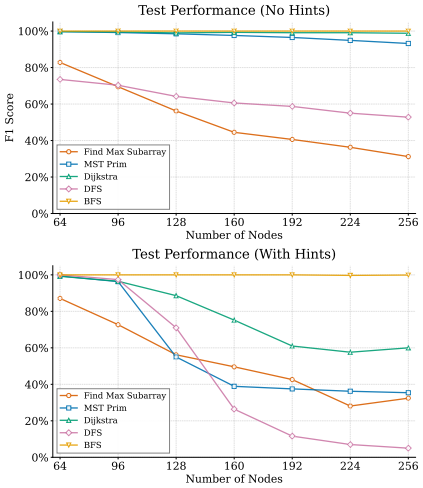
Expressiveness of k-FloydNet. We further evaluate k-FloydNet on BREC. As summarized in Table 2 and detailed in Appendix E.1, 3-FloydNet and 4-FloydNet achieve accuracies of 95.0% and 99.8%, respectively. Each variant distinguishes exactly the same BREC pairs as its corresponding WL reference, extending this close structural and realized-expressivity correspondence across tuple orders (Section 2.4).

3.2 Learned Relation Composition for Algorithmic Reasoning

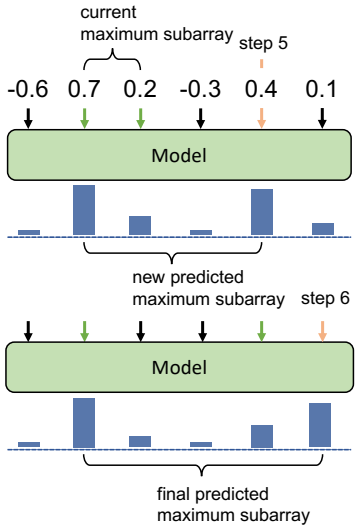
We next test whether pivot-mediated relation composition supports algorithmic computation and OOD size generalization.

The CLRS Benchmark. The CLRS-30 benchmark (Veličković et al., 2022) is a standard suite for assessing algorithmic reasoning. We train one FloydNet model per CLRS-30 task. Evaluation is performed on out-of-distribution instances with larger problem sizes, without providing any ground-truth hints to the model at test time. We evaluate on the full benchmark and report aggregated performance across eight categories in Table 4. Under this protocol, FloydNet has the highest reported aggregate performance and exceeds 95% on most algorithms, including Strings, graph algorithms such as Floyd–Warshall, and Sorting. In these configurations, removing hints improves most tasks. To test length scalability, we extend our OOD evaluation to problem sizes up to $n = 256$, within the broader extrapolation range studied by SALSA-CLRS (Minder et al., 2023) but using our own evaluation sizes and protocol. As visualized in Figure 3(a), the no-hint model performs better at

these sizes. Full per-algorithm results and detailed extrapolation scores are available in Appendix E.2.



(a) **OOD Scalability Analysis** ($n \to 256$)**.**



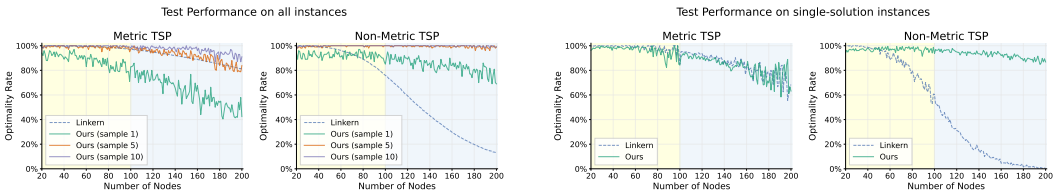
(b) **Algorithmic Alignment.**

Figure 3: **Probing the Scalability and Reasoning Logic of FloydNet on CLRS. (a)** We extend our OOD evaluation to problem sizes up to $n = 256$, within the broader extrapolation range studied by Minder et al. (2023) but using our own evaluation sizes and protocol. The results indicate that models trained *without hints* (upper) generally exhibit superior length extrapolation capabilities compared to those trained *with hints* (lower). **(b)** Visualization of the internal reasoning process for the Maximum Subarray problem (Kadane's algorithm). The model iteratively predicts the new maximum subarray based on the current maximum subarray and the next number corresponding to the current step. The observed alignment provides evidence consistent with learning a state-updating computation, but does not by itself establish faithful execution of the reference algorithm.

Probing Algorithmic Reasoning. To examine whether FloydNet represents intermediate algorithmic structure, we probed its layer-wise states on dynamic programming tasks. We use *Find Max. Subarray (Kadane's algorithm)* as a case study, where the reference solution maintains and updates an optimal subarray during a traversal. As visualized in Figure 3b, alignment with the reference intermediate trajectory provides evidence consistent with learning a state-updating computation, rather than relying solely on input–output correlations; it does not by itself establish faithful execution of the reference algorithm.

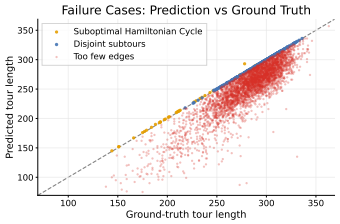
Table 4: Average test accuracy (%) on the CLRS-30 benchmark, aggregated by algorithm category. N denotes the number of algorithms in each group. All models are trained on problem sizes of $n = 16$ and tested on larger instances of $n = 64$.

Class	Model	Sort (N=4)	Search (N=3)	D&C (N=1)	Greedy (N=2)	DP (N=3)	Graph (N=12)	String (N=2)	Geometry (N=3)	Total (N=30)
GNN	**Triplet-GMPNN** (Ibarz et al., 2022)	75.64	58.83	76.36	91.22	81.99	86.42	49.09	94.09	80.04
GNN	**RANR** (Xu & Veličković, 2024)	94.15	82.90	83.53	91.66	42.69	74.15	49.12	88.44	75.78
GNN	**G-ForgetNet** (Bohde et al., 2024)	78.09	63.84	78.97	91.79	86.70	88.80	54.73	95.09	82.89
GT	**RT** (Diao & Loynd, 2022)	50.01	65.31	66.52	85.33	83.20	65.33	32.52	84.55	66.18
GT	**ET** (Müller et al., 2024)	82.26	63.00	64.44	81.67	83.49	86.08	54.84	88.22	80.13
Ours	**FloydNet**	**100.00**	**91.60**	**86.20**	**93.17**	**90.02**	**98.60**	**99.72**	**99.68**	**96.64**

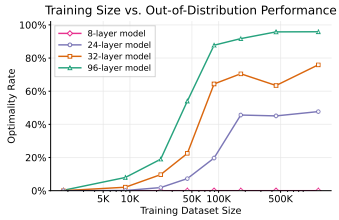


(a) **Full Benchmark (Multi-Solution)**

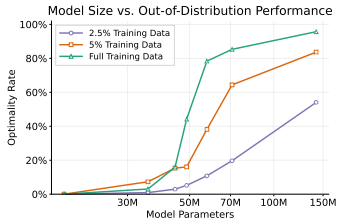
(b) **Filtered Subset**



(c) Failure Case Analysis



(d) Data Scaling Analysis



(e) Model Scaling Analysis

Figure 4: **Analysis of FloydNet on the TSP.** (a) Optimality rate on the benchmark (including multi-solution instances). The yellow region denotes validation on in-distribution sizes ($N \leq 100$), while the cyan region demonstrates OOD generalization to larger problem sizes ($100 < N \leq 200$). **(b)** Performance on the filtered subset, where perturb-and-resolve is used to reduce ambiguity from multiple optimal tours. **(c)** Distribution of error types. **(d-e)** Scaling laws showing that FloydNet effectively utilizes increased data and model depth, avoiding the over-smoothing issues typical of deep MPNNs.

3.3 Pairwise Relational Optimization: TSP

TSP tests pair-state computation by mapping dense edge weights to a Hamiltonian-cycle edge set.

Settings. We use two benchmarks: a *non-metric* TSP dataset of random complete graphs (up to 200 nodes, integer weights in $[1, 100]$) and a *metric* TSP dataset from random 2D integer coordinates. To handle solution multiplicity, we train within a DDPM framework (Ho et al., 2020). Reference optimal tours are obtained with Concorde, an exact branch-and-cut solver (Applegate et al., 1998). The model is trained on $N \leq 100$ and tested on $100 < N \leq 200$ for OOD evaluation. We benchmark against Linkern, an implementation of the Chained Lin–Kernighan heuristic (Applegate et al., 2003). We also evaluate a filtered subset to reduce ambiguity from multiple optimal tours. Full details are in Appendix E.3.

Results. On held-out non-metric TSP instances ($100 < N \leq 200$), FloydNet achieves a 99.8% optimality rate with 10 samples (Figure 4a), versus Linkern's 38.8%. The gap widens with size: at $180 \leq N \leq 200$, Linkern drops to 16.1% while FloydNet maintains 99.4%. On the filtered non-metric subset (Figure 4b), FloydNet finds the exact optimum in 92.6% of cases vs. Linkern's 15.7%; at $180 \leq N \leq 200$ this becomes 88.3% vs. 1.3%. On metric TSP, where Linkern benefits from geometric structure, FloydNet achieves comparable performance. Most failure cases produce incomplete tours rather than suboptimal ones, and this issue diminishes consistently with model scale (see Appendix E.3).

Scaling Properties. As shown in Figures 4d and 4e, OOD performance improves with larger training sets and greater model depth (from $L = 1$ to $L = 96$) in the evaluated TSP configurations. In these configurations, deeper models perform better; every FloydNet layer retains access to all candidate pivots.

3.4 Transfer to Applied Relational Prediction

Finally, we test whether the pair-relation inductive bias transfers beyond constructed algorithmic settings to molecular, contact-prediction, node-classification, and graph-level prediction benchmarks.

The LRGB results in Table 5 show that, among the reported baselines, FloydNet performs best on PCQM-Contact and COCO-SP, while remaining competitive on PascalVOC-SP and trailing Graph-Mamba on the two Peptides tasks. Results on ZINC are reported in Appendix E.4.

Training and validation curves for the remaining tasks are discussed in Appendix E.4.

Table 5: Mean performance on Long-Range Graph Benchmarks over five runs. Baseline values are shown with their original model citations; dataset definitions and metrics follow Dwivedi et al. (2022).

Class	Task Dataset Metric	Graph Class. Peptides-Func AP ↑	Graph Reg. Peptides-Struct MAE ↓	Node Class. PascalVOC-SP F1 ↑	Node Class. COCO-SP F1 ↑	Link Pred. PCQM-Contact MRR ↑
GNN	GCN(Kipf & Welling, 2017; Dwivedi et al., 2022)	0.5930	0.3496	0.1268	0.0841	0.3234
GT	Exphormer(Shirzad et al., 2023)	0.6258	0.2512	0.3446	0.3430	0.3587
	GPS(Rampášek et al., 2022)	0.6535	0.2500	0.3724	0.3774	0.3437
SSM	Graph-Mamba(Wang et al., 2024)	0.6739	0.2478	0.4191	0.3960	0.3395
Ours	**FloydNet**	**0.6224**	**0.2612**	**0.4046**	**0.4901**	**0.6143**

4 Related Work

4.1 Beyond 1-WL and Higher-Order GNNs

Standard message-passing GNNs are bounded by the 1-WL test (Xu et al., 2019; Morris et al., 2019). Approaches beyond this limit include higher-order invariant networks (Maron et al., 2019b;a), subgraph models (Bevilacqua et al., 2022; Zhang & Li, 2021), and spectral methods (Zhang et al., 2024b; Lim et al., 2023). The k-Folklore Weisfeiler–Lehman hierarchy (Cai et al., 1992; Maron et al., 2019a) is a useful yardstick for ordered-tuple models: it refines each tuple from the common-pivot multiset of its coordinate replacements, and FGNN explicitly mimics this refinement (Zhang et al., 2024a). FloydNet is motivated independently by relation composition. Under atomic tuple initialization and invariant readout, the implemented k-FloydNet is upper-bounded by k-FWL (see Appendix D); its pivot update is inspired by, but need not be injective like, the discrete k-FWL refinement.

4.2 Entity, Pair, and Higher-Order Attention

Entity-state attention. Graph Transformers apply Self-Attention to node states (Dwivedi & Bresson, 2021; Ying et al., 2021; Kim et al., 2022; Rampasek et al., 2022; Lai et al., 2023), commonly using positional or structural encodings to expose graph topology.

Pair-state attention. Explicit pair representations and triangle-indexed updates precede our work. Evoformer and Pairformer combine pair states with triangle multiplication and attention (Jumper et al., 2021; Abramson et al., 2024); Edge Transformer applies triangular attention to node-pair states (Bergen et al., 2021), with subsequent analysis relating it to the WL hierarchy (Müller et al., 2024); EGT maintains an edge channel alongside node attention (Hussain et al., 2021). These works use explicit pair states. PA differs in its attention candidate: for each pivot, both composable relations jointly construct the candidate key and value, which the target relation then scores.

Higher-order attention. k-FloydNet extends PA to ordered tuple states through common-pivot coordinate replacement. Its attention-operation abstraction yields Self-Attention at $k = 1$ and PA at $k = 2$; the main differences from prior pair-state models are the jointly constructed pivot candidate and the ordered-tuple extension (Appendix B.6.1).

Appendix Appendix C discusses NAR, combinatorial optimization, and message-passing limitations.

5 Conclusion and Future Work

FloydNet updates explicit ordered relation states through pivot-specific relation composition. Its k-tuple formulation places this operation alongside entity-state attention and supports a k-FWL upper

bound under the stated assumptions. Across BREC, CLRS, TSP, and graph prediction benchmarks, explicit relation states provide a useful inductive bias. The principal limitation is dense $\mathcal{O}(N^3)$ pair-pivot computation; future work will study sparse pivots, higher-order architectures, multimodal relational inputs, and connections to Abstract DP (Bertsekas, 2022).

REFERENCES

Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630:493–500, 2024. doi: 10.1038/s41586-024-07487-w.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2.

David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. On the solution of traveling salesman problems. In *Proceedings of the International Congress of Mathematicians*, volume 3, pp. 645–656, 1998. doi: 10.4171/DMV/1-3/62.

David Applegate, William Cook, and André Rohe. Chained lin–kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, 15(1):82–92, 2003. doi: 10.1287/ijoc.15.1.82.15157.

Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, September 1998. ISSN 0004-5411. doi: 10.1145/290179.290180. URL https://doi.org/10.1145/290179.290180.

László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pp. 684–697, 2016.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966. doi: 10.1126/science.153.3731.34.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. doi: 10.1016/j.ejor.2020.07.063.

Leon Bergen, Timothy J. O'Donnell, and Dzmitry Bahdanau. Systematic generalization with edge transformers. *arXiv preprint arXiv:2112.00578*, 2021. URL https://arxiv.org/abs/2112.00578.

Dimitri Bertsekas. *Abstract dynamic programming*. Athena Scientific, 2022.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *International Conference on Learning Representations*, 2022.

Montgomery Bohde, Meng Liu, Alexandra Saxton, and Shuiwang Ji. On the markov property of neural algorithmic reasoning: Analyses and methods. *arXiv preprint arXiv:2403.04929*, 2024.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. URL https://github.com/jax-ml/jax.

Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992. doi: 10.1007/BF01202324.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 10383–10395, 2020.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3 edition, 2009. ISBN 978-0-262-03384-8.

Francesco Di Giovanni, T. Konstantin Rusch, Michael M. Bronstein, Andreea Deac, Marc Lackenby, Siddhartha Mishra, and Petar Veličković. How does over-squashing affect the power of GNNs? *Transactions on Machine Learning Research*, 2023.

Cameron Diao and Ricky Loynd. Relational attention: Generalizing transformers for graph-structured tasks. *arXiv preprint arXiv:2210.05062*, 2022.

Andrew Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. *arXiv preprint arXiv:2203.15544*, 2022.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023. URL https://jmlr.org/papers/v24/22-0563.html.

Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962. doi: 10.1145/367766.368168.

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

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Martin Grohe. *Descriptive complexity, canonisation, and definable graph structure theory*, volume 47. Cambridge University Press, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Md. Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Edge-augmented graph transformers: Global self-attention is enough for graphs. *arXiv preprint arXiv:2108.03348*, 2021. URL https://arxiv.org/abs/2108.03348.

Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Joseph Dudzik, Matko Bošnjak, Alex Vitvitskyi, Yulia Rubanova, et al. A generalist neural algorithmic learner. In *Learning on graphs conference*, pp. 2–1. PMLR, 2022.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596:583–589, 2021. doi: 10.1038/s41586-021-03819-2.

Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 14582–14595, 2022.

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

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21188–21198, 2020.

Xin Lai, Yang Liu, Rui Qian, Yong Lin, and Qiwei Ye. Deeper exploiting graph structure information by discrete ricci curvature in a graph transformer. *Entropy*, 25(6):885, 2023.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973. doi: 10.1287/opre.21.2.498.

Sadegh Mahdavi, Kevin Swersky, Thomas Kipf, Milad Hashemi, Christos Thrampoulidis, and Renjie Liao. Towards better out-of-distribution generalization of neural algorithmic reasoning tasks. *arXiv preprint arXiv:2211.00692*, 2022.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in neural information processing systems*, volume 32, 2019a.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019b.

Sean McLeish, Avi Schwarzschild, and Tom Goldstein. Benchmarking chatgpt on algorithmic reasoning. *arXiv preprint arXiv:2404.03441*, 2024.

Julian Minder, Florian Grötschla, Joël Mathys, and Roger Wattenhofer. Salsa-clrs: A sparse and scalable benchmark for algorithmic reasoning. *arXiv preprint arXiv:2309.12253*, 2023.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019. doi: 10.1609/aaai.v33i01.33014602.

Luis Müller, Daniel Kusuma, Blai Bonet, and Christopher Morris. Towards principled graph transformers. *Advances in Neural Information Processing Systems*, 37:126767–126801, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Ladislav Rampasek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, 2022.

Bernard Roy. Transitivité et connexité. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 249:216–218, 1959.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 31613–31632, 2023.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 3706–3731, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d0df4cf12be188d1e282d9f34a8f4f3b-Abstract-Conference.html.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Petar Veličković and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. doi: 10.1016/j.patter.2021.100273.

Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. *arXiv preprint arXiv:1910.10593*, 2019.

Petar Veličković, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The CLRS algorithmic reasoning benchmark. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22084–22102, 2022. URL https://proceedings.mlr.press/v162/velickovic22a.html.

Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024.

Yanbo Wang and Muhan Zhang. An empirical study of realized GNN expressiveness. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52134–52155, 2024. URL https://proceedings.mlr.press/v235/wang24p.html.

Stephen Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962. doi: 10.1145/321105.321107.

Kaijia Xu and Petar Veličković. Recurrent aggregators in neural algorithmic reasoning. *arXiv preprint arXiv:2409.07154*, 2024.

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

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Bohang Zhang, Jingchu Gai, Yiheng Du, Qiwei Ye, Di He, and Liwei Wang. Beyond weisfeiler-lehman: A quantitative framework for GNN expressiveness. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=HSKaGOi7Ar.

Bohang Zhang, Lingxiao Zhao, and Haggai Maron. On the expressive power of spectral invariant graph neural networks. *arXiv preprint arXiv:2406.04336*, 2024b.

Muhan Zhang and Pan Li. Nested graph neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15734–15747, 2021.