Rethinking Test Generalisation In Neural Algorithmic Reasoning

Manasvi Aggarwal[*]
Independent Researcher
manasvi.agg95@gmail.com

Dulhan Jayalath[*]
University of Oxford
dulhan@robots.ox.ac.uk

Jonas Jürß[*]
University of Cambridge
jj570@cl.cam.ac.uk

Abstract

We discover distribution bias in the evaluation of neural algorithmic reasoning (NAR) that misrepresents out-of-distribution (OOD) generalisation of neural networks. With a Triplet-GMPNN baseline, we find that the GNN outperforms NAR in 46% of algorithmic reasoning tasks in the CLRS benchmark and is within 1 standard deviation for 67%. We show that this result is biased by test sets with specific problem instance sizes rather than a distribution of problem sizes. To address this, we introduce the **G**eneralisation **O**ut-**o**f-**D**istribution (GOOD) score, a simple way to measure NAR generalisation using the area under a test score vs problem distribution curve. Through analysis with GOOD score and empirical curves, we identify that NAR generalisation is better than reported but is *still* often outperformed by simple GNN baselines asymptotically, highlighting new opportunities to improve NAR.

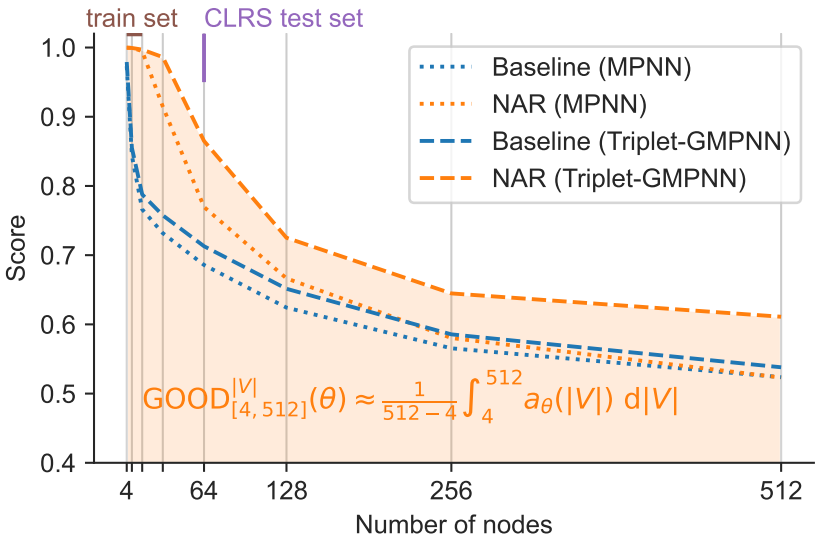

Figure 1: Average performance at different test set sizes aggregated over algorithms. GOOD computes an approximation of the area under the curve. Vertical lines show sampling points $n_1, \ldots, n_K$ (see Eq. 2). The distribution parameter $|V|$ denotes measurement over the number of nodes.

[*]Equal contribution

Aggarwal et al., Rethinking Test Generalisation In Neural Algorithmic Reasoning (Extended Abstract). Presented at the Fourth Learning on Graphs Conference (LoG 2025), Hybrid Event, December 10–12, 2025.

1 Introduction

Neural Algorithmic Reasoning (NAR) [1] teaches neural networks how to solve classical algorithms to instill neural networks with algorithmic capabilities such as step-by-step reasoning and the ability to generalise out-of-distribution. In this paradigm, algorithm inputs are represented as graphs[2] and the algorithm is executed recurrently with a graph neural network (GNN) that evolves the node embeddings over these recurrent steps to reach a solution. During training, algorithmic control flow supervision improves the alignment of the neural network with the steps of the algorithm. In turn, by training to follow the steps of the algorithm, the network generalises to out-of-distribution problem instances during training, e.g., to larger input graph sizes or to graphs constructed from different generation methods.

In this paper, we critically analyse the primary benchmark in NAR, CLRS-30 [1], and find that its standard NAR baselines often do not outperform simple non-recurrent GNNs on its out-of-distribution test set. By analysing the generalisation patterns of NAR and simple GNNs on our own test set of larger graphs, we find that conclusions about generalisation made from these test sets are biased by the specific test instance sizes. For example, a method may generalise better across a smaller, or larger, set of graph sizes than those in the test set. However, its overall generalisation pattern is represented by its performance across a range of graph sizes. Thus, this generalisation is not reflected in the test set scores as they do not account for how these methods generalise across the graph size distribution, but rather for a specific set of test graph sizes. Our analysis highlights the need for a better empirical measure of OOD generalisation that reflects the generalisation patterns of NAR methods on different test graph sizes.

To account for generalisation across a distribution of test graphs, we introduce GOOD, a measure of OOD generalisation based on the area under a score vs test graph distribution curve. Our results with GOOD show surprising generalisation patterns. Although models trained with NAR appear superior at small graph sizes, they often do not generalise to large graphs asymptotically as well as a simple GNN. By finding that NAR often does not outperform, or even performs worse than simple GNNs asymptotically, this work highlights a critical roadblock in achieving out-of-distribution generalisation in NAR. In identifying this issue and introducing GOOD, we hope to introduce further rigour in the empirical evaluation of NAR generalisation.

2 Related Work

Several works evaluate NAR without hints, while others extend generalisation to practical domains and different graph distributions. Rodionov and Prokhorenkova [2] provide an approach similar to NAR, recurrently applying the GNN a variable number of times depending on the problem size, forgoing only hint supervision. Bevilacqua et al. [3] and Mahdavi et al. [4] similarly provide ablations with recurrent steps but without hint supervision and see evidence of training without hints being beneficial for test generalisation. Jürß et al. [5] find that enforcing specific hints and other forms of intermediate supervision can hurt generalisation in recursive algorithms. Unlike this prior work, our approach studies the regime where we also remove the recurrent application of GNNs characteristic of NAR to compare NAR generalisation against simple GNNs. Moreover, other analyses of NAR generalisation extend to practical domains, e.g. brain vessel classification [6] and pointcloud registration [7], or test on larger graphs than in the training set [8] and different graph generation distributions [4, 9]. While these measure specific aspects of OOD performance, no work provides a unified generalisation metric. Critically, beyond past work, we also show the bias of standard small sets of specific test graph sizes and instead provide an analysis of generalisation over large *distributions* of test graphs, in turn setting a new standard for evaluation. Lastly, our work is not to be confused with the *GOOD* benchmark [10] which provides graph datasets, rather than a metric, for measuring different covariate and concept shifts between train and test sets.

3 Method

GNN Baseline. We modify CLRS to include a GNN baseline that uses the same processor network as an NAR baseline, but with no recurrence and no intermediate supervision of algorithmic control flow (i.e., hints). By ablating these two features, training with CLRS is equivalent to training a GNN

[2] In the CLRS benchmark, these are fully connected graphs.

on the inputs and outputs of the algorithm, removing all of the additional algorithmic features of NAR. Note that NAR involves applying the same processor a variable number of times depending on the problem size. Therefore, even without hints, it is not equivalent to common GNN architectures which involve applying a fixed number of layers with distinct weights.

Generalisation Out-of-Distribution (GOOD). We also introduce a new metric for measuring empirical OOD generalisation. To represent the generalisation pattern of a method over a distribution, e.g. graph sizes, we need to evaluate the expectation

$$\mathbb{E}_{\psi \sim U(m_{\min}, m_{\max})}[a_\theta(\psi)] = \frac{1}{m_{\max} - m_{\min}} \int_{m_{\min}}^{m_{\max}} a_\theta(\psi) d\psi \tag{1}$$

where $a_\theta(\psi)$ is a function describing the accuracy of a method with a GNN parameterised by a set of weights θ on an evaluation set of graphs of instance size n. As it is often only practically feasible to sample a finite number of distribution parameters ψ. We define an empirical metric based on an approximation of this integral with the trapezoid rule

$$\text{GOOD}^{\psi}_{n_1,\ldots,n_K}(\theta) = \frac{1}{n_K - n_1} \sum_{i=2}^{K} \frac{a_\theta(n_{i-1}) + a_\theta(n_i)}{2} \cdot (n_i - n_{i-1}), \tag{2}$$

where $m_{\min} = n_1 < n_2 < \cdots < n_K = m_{\max}$ denote the distribution parameters we evaluate on. For notational simplicity, we write $\text{GOOD}_{n_1,\ldots,n_K}$ as $\text{GOOD}^{\psi}_{[m_{\min}, m_{\max}]}$, focusing on the approximated interval (e.g. the range of graph sizes) and leave the intermediate distribution parameters as an implementation detail.

GOOD Score in Python code

```
import numpy as np
def good_score(test_scores, distribution_params):
    return np.trapz(test_scores, distribution_params)
```

We treat the integration interval as continuous in our formulation. If ψ takes discrete values (e.g. specific graph sizes $n_1, \ldots, n_K$), a strict expectation over the discrete uniform distribution would divide by $n_K - n_1 + 1$ rather than $n_K - n_1$. However, since all GOOD scores are multiplied by the same constant factor $(n_K - n_1)/(n_K - n_1 + 1) \approx 1$, they remain directly comparable. We adopt the continuous interpretation for cleaner notation.

Our metric addresses settings where performance varies non-uniformly along a generalisation axis, and the sampled graph distribution can overlap with the training set (e.g. graph sizes 1-16 are included in the GOOD calculation throughout the paper). For instance, when evaluating robustness to graph size, performance typically decays rapidly, making a logarithmic sampling more informative than uniform intervals. However, if our goal is to estimate performance over the entire range of the generalisation axis—such as all Erdős-Rényi graphs with edge probabilities $p \in [0, 1]$, or all graphs up to some maximum size—we need $\mathbb{E}_\psi[a_\theta(\psi)]$ where ψ is uniformly distributed. Directly sampling ψ uniformly would forgo the ability to strategically place evaluation points where they are most informative. Our metric resolves this by computing a weighted expectation: we evaluate at deliberately chosen (potentially non-uniform) points along ψ, then re-weight to recover the uniform expectation. In practice, the sampling method should remain the same across evaluations so that approximation errors are consistent and values are comparable.

4 Experiments

We focus on the CLRS-30 benchmark, evaluating the GNN baseline (Section 3) alongside standard NAR methods to test whether a simple GNN trained only on final outputs can achieve competitive performance. We experiment with two GNN processors: (a) **MPNN** [11], a standard message-passing neural network , and (b) **Triplet-GMPNN** [12], a higher-order GNN specifically introduced for NAR with triplet-based edge processors.

Following the standard CLRS-30 protocol [1], models are trained on synthetic graphs within the benchmark's input size (from 4 up to 16 nodes). To evaluate the generalisation capabilities, we

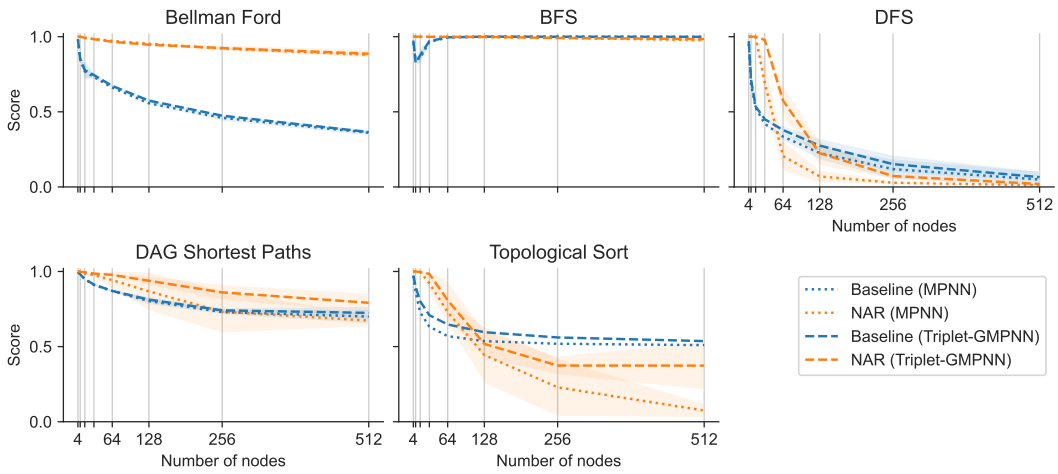

Figure 2: Generalisation patterns of selected graph algorithms across test graph sizes. In DFS and Topological Sort, the simple GNN baseline consistently outperforms NAR across larger graph sizes. Mean and standard deviation over 5 seeds.

perform evaluations on both in-distribution (ID) and out-of-distribution (OOD) settings. ID evaluation measures performance on test graphs of the same size as the training distribution, while OOD experiments evaluate models on graphs of varying sizes larger than those seen in the train set. We report averages over multiple random seeds and apply early stopping based on validation accuracy. We use the original CLRS-30 preprocessing, augmentation, evaluation protocol, and official hyperparameters [1].

4.1 Results

We first analyse results across a set of 24 algorithms (Figure 3). On the CLRS test set with Triplet-GMPNN, a simple GNN outperforms NAR in 11/24 (45.8%) of the tasks we tried and is within 1 standard deviation for 16/24 (66.7%). With MPNN, similar results hold, with 11/24 wins (45.8%), and 14/24 on-par 14/24 (58.3%) for the simple GNN. This surprising result indicates a serious failure of NAR to generalise. In Figure 2, we observe that while NAR tends to be more robust to graph sizes that are slightly out of distribution, a GNN baseline will eventually perform on par for sufficiently large graphs, e.g. on Topological Sort and DFS. From this point, we observe a phase change where the GNN baseline retains relatively stable performance whereas NAR continues worsening dramatically. Counter-intuitively, NAR exhibits more overfitting than a simpler GNN baseline despite having been developed explicitly with OOD generalisation in mind. In Table 1, we show the GOOD scores reflect this asymptotic difference in generalisation.

5 Discussion

What explains NAR failure cases? The performance gap varies significantly depending on the type of algorithm. Thus, one explanation could be algorithm-dependent issues with generalisation. For example, Binary Search can theoretically be solved in a single decision step as finding a value is equivalent to matching a node embedding. In contrast, Bellman-Ford, Floyd-Warshall, and Dijkstra require several iterations to update distances across the graph. Here, NAR performs better than the GNN because recurrence helps handle long chains of updates. Further work is required to understand why NAR may generalise worse than a GNN beyond certain graph sizes and on certain algorithms. Recent work [3–5] has found that enforcing specific hints and other forms of intermediate supervision can hurt generalisation.

Conclusion. We identify that test evaluations in NAR focus on specific graph instance sizes and fail to account for generalisation across a range of graph sizes to reflect true algorithmic generalisation. By plotting test score vs graph size curves, we show that measuring test generalisation on a single set of graph sizes leads to a misrepresentation of the generalisation of NAR methods. Thus, we introduce

Algorithm	Processor	Method	$\text{GOOD}^{|V|}_{[4,512]}$
Bellman Ford	MPNN	Baseline	0.4989
	MPNN	NAR	**0.9274**
	Triplet-GMPNN	Baseline	0.5107
	Triplet-GMPNN	NAR	**0.9286**
BFS	MPNN	Baseline	**0.9923**
	MPNN	NAR	0.9903
	Triplet-GMPNN	Baseline	**0.9926**
	Triplet-GMPNN	NAR	0.9924
DFS	MPNN	Baseline	**0.1756**
	MPNN	NAR	0.1181
	Triplet-GMPNN	Baseline	0.2078
	Triplet-GMPNN	NAR	**0.2146**
DAG Shortest Paths	MPNN	Baseline	0.7670
	MPNN	NAR	**0.7846**
	Triplet-GMPNN	Baseline	0.7796
	Triplet-GMPNN	NAR	**0.8803**
Topological Sort	MPNN	Baseline	**0.5408**
	MPNN	NAR	0.3411
	Triplet-GMPNN	Baseline	**0.5878**
	Triplet-GMPNN	NAR	0.4956

Table 1: GOOD performance over the set of selected graph algorithms in Figure 2. The distribution parameter $|V|$ represents the number of nodes in the graph.

the Generalisation Out-of-Distribution (GOOD) score, a new metric based on the area under a test score vs test graph distribution curve that accounts for the generalisation of NAR methods across a distribution of graphs. With GOOD, and a new perspective on test generalisation we identify failures of NAR on large test graph sizes where it surprisingly fails to beat simple GNN baselines.

Author Contributions

Following the Contributor Roles Taxonomy (CRediT) methodology: **MA**: Software, Writing – review & editing **DJ**: Conceptualisation, Methodology, Supervision, Writing – original draft, review & editing, Resources, Investigation **JJ**: Conceptualisation, Methodology, Supervision, Writing – original draft, review & editing, Visualisation, Formal analysis

Acknowledgements

We thank Petar Veličković and Dobrik Georgiev for valuable discussions and we thank the reviewers at the Learning on Graphs (LoG 2025) conference for insightful comments which improved the paper.

The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work. http://dx.doi.org/10.5281/zenodo.22558.

DJ is supported by an AWS Studentship from the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems (AIMS) (EP/S024050/1).

References

[1] Petar Velickovic, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The CLRS algorithmic reasoning benchmark. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 22084–22102. PMLR, 2022. URL https://proceedings.mlr.press/v162/velickovic22a.html. 2, 3, 4

[2] Gleb Rodionov and Liudmila Prokhorenkova. Neural algorithmic reasoning without intermediate supervision. *CoRR*, abs/2306.13411, 2023. doi: 10.48550/ARXIV.2306.13411. URL https://doi.org/10.48550/arXiv.2306.13411. 2

[3] Beatrice Bevilacqua, Kyriacos Nikiforou, Borja Ibarz, Ioana Bica, Michela Paganini, Charles Blundell, Jovana Mitrovic, and Petar Veličković. Neural algorithmic reasoning with causal regularisation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2272–2288. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/bevilacqua23a.html. 2, 4

[4] Sadegh Mahdavi, Kevin Swersky, Thomas Kipf, Milad Hashemi, Christos Thrampoulidis, and Renjie Liao. Towards better out-of-distribution generalization of neural algorithmic reasoning tasks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=xkrtvHlp3P. 2

[5] Jonas Jürß, Dulhan Hansaja Jayalath, and Petar Velickovic. Recursive algorithmic reasoning. In *Learning on Graphs Conference, 27-30 November 2023, Virtual Event*, volume 231 of *Proceedings of Machine Learning Research*, page 5. PMLR, 2023. URL https://proceedings.mlr.press/v231/jurss24a.html. 2, 4, 9

[6] Danilo Numeroso, Davide Bacciu, and Petar Velickovic. Dual algorithmic reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=hhvkdRdWt1F. 2

[7] Efimia Panagiotaki, Daniele De Martini, Lars Kunze, and Petar Velickovic. Nar-*icp: Neural execution of classical icp-based pointcloud registration algorithms. *CoRR*, abs/2410.11031, 2024. doi: 10.48550/ARXIV.2410.11031. URL https://doi.org/10.48550/arXiv.2410.11031. 2

[8] Julian Minder, Florian Grötschla, Joël Mathys, and Roger Wattenhofer. Salsa-clrs: A sparse and scalable benchmark for algorithmic reasoning, 2023. URL https://arxiv.org/abs/2309.12253. 2

[9] Dobrik Georgiev, Pietro Lio, Jakub Bachurski, Junhua Chen, Tunan Shi, and Lorenzo Giusti. Beyond erdos-renyi: Generalization in algorithmic reasoning on graphs. In *The Second Learning on Graphs Conference*, 2023. 2

[10] Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. GOOD: A graph out-of-distribution benchmark. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/0dc91de822b71c66a7f54fa121d8cbb9-Abstract-Datasets_and_Benchmarks.html. 2

[11] 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, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017. URL http://proceedings.mlr.press/v70/gilmer17a.html. 3

[12] Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Joseph Dudzik, Matko Bosnjak, Alex Vitvitskyi, Yulia Rubanova, Andreea Deac, Beatrice Bevilacqua, Yaroslav Ganin, Charles Blundell, and Petar Velickovic. A generalist

neural algorithmic learner. In Bastian Rieck and Razvan Pascanu, editors, *Learning on Graphs Conference, LoG 2022, 9-12 December 2022, Virtual Event*, volume 198 of *Proceedings of Machine Learning Research*, page 2. PMLR, 2022. URL https://proceedings.mlr.press/v198/ibarz22a.html. 3, 10

[13] Petar Velickovic and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. doi: 10.1016/J.PATTER.2021.100273. URL https://doi.org/10.1016/j.patter.2021.100273. 9

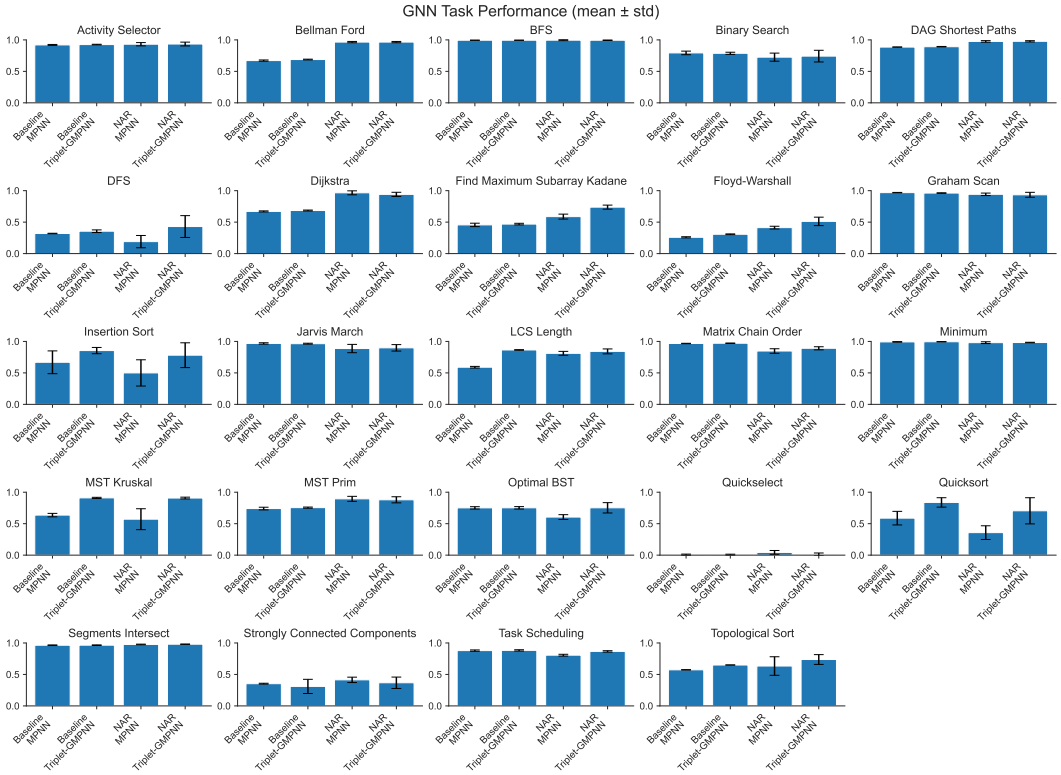

Figure 3: Average score of most algorithms on the CLRS-30 OOD test set. Mean and standard deviation over 5 seeds. Surprisingly, the Simple GNN (no hints, no recurrence) matches or exceeds specialized NAR architectures on about half the tasks for both GNN architectures tested. See Figure 4 for a zoomed-in version of quickselect.

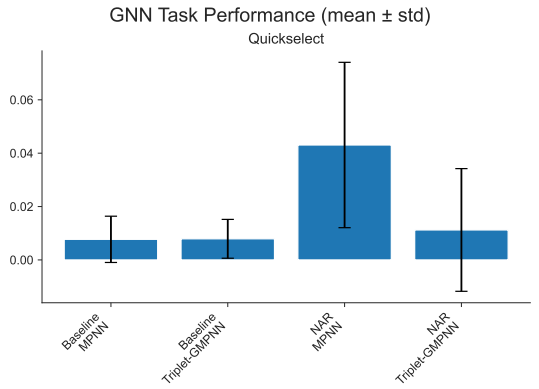

Figure 4: Average score of quickselect on the CLRS-30 OOD test set. Zoomed in from Figure 3.

A Appendix

A.1 Additional results

See Figures 3 and 5.

A.2 Implementation details

NAR formulates the execution of classical algorithms as a sequence of graph transformations, where inputs, outputs, and algorithmic intermediate states (i.e., hints) are represented as a graph structure

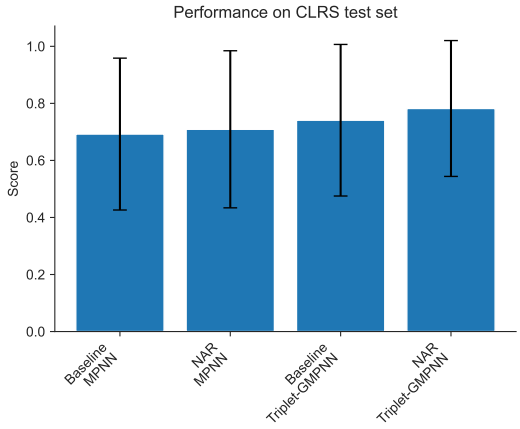

Figure 5: Average performance of all algorithms in Figure 2. Standard deviation over different algorithms.

$G = (V, E)$ and used to guide the learning process. The standard CLRS-30 framework [13], which evaluates NAR across 30 fundamental algorithms, is built on two critical components: (1) temporal recurrence to simulate step-by-step execution, and (2) intermediate supervision through algorithmic hints to guide learning. Formally, at each time step t, he model processes encoded inputs through a recurrent GNN processor:

$$\mathbf{h}_i^t = f_n(\mathbf{x}_i^t)\ ; \quad \mathbf{h}_{ij}^t = f_e(\mathbf{e}_{ij}^t)\ ; \quad \mathbf{h}_g^t = f_g(\mathbf{g}^t) \tag{3}$$

$$\mathbf{p}_i^t, \mathbf{p}_{ij}^t = \psi(\mathbf{h}_i^t, \mathbf{p}_i^{t-1}, \mathbf{h}_{ij}^t, \mathbf{h}_g^t) \tag{4}$$

where f_n, f_e, and f_g are linear encoding layers. ψ is the GNN processor [5]. $\mathbf{p}_i^{t-1}$ represents the recurrent state from the previous time step. The model is trained to minimize both hint prediction and final output losses:

$$\mathcal{L}_{\text{CLRS}} = \mathcal{L}_{\text{output}}(\hat{\mathcal{O}}, \mathcal{O}) + \lambda \sum_{t=1}^{T} \mathcal{L}_{\text{hint}}(\hat{\mathcal{H}}^t, \mathcal{H}^t) \tag{5}$$

GNN Baseline: Non-Recurrent, Hint-Free Baseline: We modify the CLRS formulation to critically evaluate its core components (recurrence and hint supervision). This allows us to assess whether simpler approaches can achieve comparable performance on algorithmic reasoning tasks. Here, along with removing the hint loss, we replace the multi-step recurrent GNN by a single-step processor, and the model directly predicts the final output. The equations 4 and 5 are modified to:

$$\mathbf{p}_i, \mathbf{p}_{ij} = \psi(\mathbf{h}_i^0, \mathbf{h}_{ij}^0, \mathbf{h}_g^0) \tag{6}$$

$$\mathcal{L}_{\text{baseline}} = \mathcal{L}_{\text{output}}(\hat{\mathcal{O}}, \mathcal{O}) \tag{7}$$

Here, in Eq. 6, the recurrent state $\mathbf{p}_i^{t-1}$ is removed, and the model is not supervised on hint loss, as shown in Eq. 7. This baseline relies solely on graph structure and final output supervision, learning a direct input-output mapping rather than simulating intermediate algorithmic steps, and allows us to assess the specific contributions of recurrence and hints to algorithmic reasoning.

A.3 Additional Details

We follow the original CLRS-30 experimental protocol with standard hyperparameters to ensure a fair comparison. Tables 2 and 3 summarize the key settings used across all experiments, for both our NAR model and the GNN baselines. Note that CLRS runs all GNNs on fully connected graphs, with edge features indicating whether an edge is present in the ground-truth graph.

Parameter	Value	Description
Embedding Size (h)	128	Node/edge embedding dimension
Batch Size	32	Samples per batch
Optimizer	Adam	$\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$
Learning Rate	0.001	Adam optimizer learning rate
Gradient Clipping	1.0	Norm clipping constant
Training Steps	10,000	Number of training epochs
Random Seeds	5	Independent runs
Graph Sizes (Train, Multi-size)	{4, 7, 11, 13, 16}	Cyclic sequence of training sizes
Graph Sizes (Test)	64	Evaluation size (CLRS Benchmark)

Table 2: Training hyperparameters (from Ibarz et al. [12]).

Parameter	Value	Description
Hidden Size	128	Dimension of latent node/edge states
Message Passing Steps	1	Number of propagation rounds
Attention Heads	1	For GAT-based processors
Dropout	0.0	Dropout probability
Layer Normalization	True	Applied in processor layers
LSTM After Processor	False	If LSTM is used after message passing
Triplet Features	8	Number of triplet features (for Triplet-GMPNN)
Processor Type	MPNN / Triplet-GMPNN	GNN baseline architectures
Encoder Initialization	Xavier on scalars	Weight initialization scheme

Table 3: Architecture hyperparameters of NAR and GNN baselines (from Ibarz et al. [12]).