UNCERTAINTY-AWARE NEURAL ALGORITHMIC REASONING: ALGORITHM EXECUTION OVER BELIEF-VALUED GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural Algorithmic Reasoning (NAR) aims to learn algorithmic execution with neural networks, but current models assume fully observable, deterministic inputs and fail under real-world uncertainty. In practice, graph structures are partially observable, edge weights are corrupted by measurement noise, and topology itself may be distorted by phantom or missing connections. We introduce **Uncertainty-Aware Neural Algorithmic Reasoning (UANAR)**, a framework that reformulates algorithmic execution as probabilistic inference over uncertain graph observations. We represent each input as a *belief-valued graph*, capturing uncertainty in edge weights and connectivity through Monte Carlo estimates, and explicitly separating measurement noise from structural uncertainty prior to reasoning. On this foundation, we propose three architectural contributions: One, a *Noise-Type-Aware Encoder* that disentangles weight and topological uncertainty through separate learned pathways with a learned fusion gate; Two, *Uncertainty-Propagating Hints* that augment the intermediate reasoning stream with the model's own step-wise confidence estimates; and Three, an *Adaptive Calibration Loss* that penalizes overconfident incorrect predictions without suppressing well-calibrated uncertainty. We evaluate across a suite of classical algorithms spanning dynamic programming, relaxation-based shortest-path, and greedy families—including Floyd–Warshall, Bellman–Ford, Dijkstra, and MST-Prim—under diverse realistic noise regimes across a broad range of severities and combinations. Our models consistently and substantially outperform the deterministic baseline in both robustness and calibration. Systematic ablations across a large configuration space confirm the independent contribution of each component and establish principled design boundaries for uncertainty-aware algorithmic reasoning.

1 INTRODUCTION

Neural Algorithmic Reasoning (NAR) has emerged as a well-defined paradigm for enabling neural networks to learn and execute classical computational procedures, combining the adaptability of neural networks with the structural priors of algorithms. Early work established this framework, showing that neural models can internalize algorithmic structure across tasks including shortest-path computation, graph traversal, and iterative reasoning (Veličković & Blundell, 2021; Veličković et al., 2022). Graph Neural Networks have been shown to align closely with dynamic programming-style computation (Dudzik & Veličković, 2022), and more recent architectures refine this alignment for improved generalization (Mahdavi et al., 2023; Bevilacqua et al., 2023; Georgiev et al., 2024). Together, this line of work establishes NAR as a powerful framework for learning algorithmic computation—under the assumption of clean, fully observed, and deterministic inputs.

Existing NAR models are primarily designed to execute precise, deterministic procedures, an assumption that rarely holds in real-world settings. Graph-structured data are often noisy, incomplete, and indirectly measured: edge weights may contain measurement error, links may be intermittently missing, and spurious connections can arise from sensor noise or partial observations. Consequently, NAR models must operate on inputs that are inherently uncertain and only partially observed. This mismatch raises three fundamental questions: *(i)* how do neural algorithmic reasoners behave when their inputs are uncertain; *(ii)* can algorithmic structure be maintained when models reason over

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.

Table 1: Standard NAR accuracy under noise (CLRS-30 MPNN baseline (Veličković et al., 2022; Gilmer et al., 2017), n=16). Predecessor accuracy for FW-APSP/Bellman–Ford/Dijkstra; edge-membership for MST-Prim. **Bold**: catastrophic (≤ 0.50). †Mixed: g=0.2, o=0.3, r_{fp}=0.2, r_{fn}=0.1.

Noise Condition		FW-APSP	Bellman–Ford	Dijkstra	MST-Prim
Clean	—	79.4% ± 0.57	95.2% ± 0.13	93.6% ± 1.07	92.1% ± 0.43
Weight Corruption					
Gaussian	g=0.30	73.7% ± 0.11	90.2% ± 1.15	89.1% ± 0.39	89.2% ± 1.19
Outlier	o=0.30	72.8% ± 0.26	77.5% ± 0.18	79.9% ± 0.47	73.1% ± 0.44
Structural Corruption					
False Positive	r_{fp}=0.20	76.7% ± 0.98	58.7% ± 0.23	**43.1% ± 0.61**	**28.2% ± 2.9**
False Negative	r_{fn}=0.20	73.9% ± 0.77	95.2% ± 0.81	93.6% ± 0.11	92.1% ± 0.13
Combined					
Mixed†	—	67.8% ± 0.91	71.2% ± 1.25	71.9% ± 1.78	57.1% ± 2.03

distributions of graphs; and *(iii)* can such models quantify the reliability of their outputs under input uncertainty? Addressing these questions motivates a framework in which uncertainty is explicitly represented in the input, propagated through the computation, and reflected in the model's outputs.

We study these questions across two algorithm families—shortest-path (Floyd–Warshall APSP, Bellman–Ford, Dijkstra) and spanning-tree (Prim)—under four noise types: Gaussian weight perturbation, outlier replacement, false-positive edge injection, and false-negative edge dropout, yielding 36 distinct configurations per algorithm.

Table 1 quantifies how standard NAR models (Veličković et al., 2022) degrade across algorithms under our noise suite. Shortest-path models degrade under weight corruption, with Bellman–Ford showing sizable drops but avoiding the catastrophic failures seen in greedy methods. Most critically, false-positive edge injection causes *catastrophic* failure in MST-Prim ($0.921 \rightarrow \mathbf{0.282}$) and Dijkstra ($0.936 \rightarrow \mathbf{0.431}$), while Bellman–Ford drops from 0.952 to 0.587 under the same FP rate. Mixed noise compounds these failures further, with MST-Prim dropping to 0.571 under a moderate four-type combination.

These failures are not only accuracy drops: existing NAR models remain deterministically overconfident when inputs admit multiple interpretations. Accuracy alone is blind to this failure mode.

We present **Uncertainty-Aware Neural Algorithmic Reasoning (UANAR)**, a probabilistic framework for algorithm execution over uncertain graph inputs that preserves the core CLRS setup while generalizing it to belief-valued observations. Our contributions are fourfold.

1. **Belief-Valued Graphs for Algorithmic Reasoning.** We encode edge weights and existence as beliefs via K Monte Carlo draws, yielding $(\mu_{ij}, \sigma^2_{ij}, p_{ij})$ for parametric and structural uncertainty. These replace CLRS adjacency/weight inputs without changing supervision or algorithmic structure, and reduce to deterministic CLRS in the noiseless limit.
2. **UANAR Architecture—Three Targeted Innovations.** We propose one innovation at each stage of the encode–process–decode pipeline:
   - **Noise-Type-Aware Encoder (NTE):** a dual-path MLP with a learned fusion gate that separately encodes weight uncertainty $(\mu_{ij}, \sigma^2_{ij})$ and structural uncertainty (p_{ij}), routing each signal according to the noise type present without manual switching.
   - **Uncertainty-Propagating Hints (UPH):** augments the intermediate hint stream with step-wise prediction variance, forming a feedback loop in which the processor tracks both its conclusions and its confidence in them across reasoning steps.
   - **Adaptive ECE Calibration Loss (AECE):** penalizes only overconfident incorrect predictions, correcting miscalibration without suppressing well-calibrated confidence and without degrading accuracy.

   Together, these components improve the fraction of noise conditions sustaining $\geq$75% accuracy (Floor@0.75) by $3.4\times$ on APSP (27.8% $\rightarrow$ 94.4%), recover MST-Prim from catastrophic FP collapse ($28.2\% \rightarrow 97.1\%$ at r_{fp}=0.1), and reduce mean expected calibration error by 44% across conditions.

3. **Probabilistic Evaluation Protocol.** We replace accuracy-only evaluation with a multi-metric protocol—NLL, ECE, Relative Robustness Score (RRS), Floor@0.75, and WCAcc—that jointly assesses correctness, calibration, and deployment reliability.
4. **Systematic Cross-Family Benchmark.** We evaluate four algorithms spanning dynamic programming (Floyd–Warshall), relaxation-based shortest-path (Bellman–Ford), greedy label-setting (Dijkstra), and cut-based greedy (MST-Prim) under 36 noise configurations.

2 Related work

Neural Algorithmic Reasoning. NAR imitates step-by-step algorithmic traces (Veličković & Blundell, 2021). CLRS-30 (Veličković et al., 2022) standardizes hint supervision, and Dudzik & Veličković (2022) links message-passing to dynamic programming. Follow-ups improve generalization via relaxed hints (Rodionov & Prokhorenkova, 2023), discrete latents (Rodionov & Prokhorenkova, 2025), and tropical attention (Hashemi et al., 2025). These methods assume clean, fully observed inputs.

Robustness in Graph Learning. Robust GNN work targets static tasks: edge denoising (Dai et al., 2022), label-noise handling (Qian et al., 2023; Wang et al., 2024), and causal noise modeling (In et al., 2025). In iterative algorithmic reasoning, noise breaks invariants (e.g., triangle inequality) and errors compound across steps—a qualitatively different challenge.

Uncertainty Quantification. Deep ensembles (Lakshminarayanan et al., 2017) remain strong for predictive confidence. In GNNs, conformal methods (Huang et al., 2023) and analytical uncertainty propagation (Chernikova et al., 2025) exist, but have not been applied to algorithmic reasoning with iterative hints and structured outputs (predecessors, spanning-tree edges). We bridge belief-valued inputs, uncertainty-aware encoding/hints, and calibrated decoding for algorithm execution.

Gap and Our Contribution: Prior NAR work assumes clean inputs; robust GNN and uncertainty methods do not readily extend to iterative algorithmic reasoning, where errors compound across steps. We address this gap for shortest-path and spanning-tree algorithms, which are especially sensitive to noisy weights and uncertain connectivity.

3 Preliminaries

NAR imitates step-wise algorithmic computation in an encode–process–decode pipeline: CLRS-style inputs (adjacency and edge features), optional intermediate *hints*, and algorithm-specific structured targets (e.g., predecessors or MST edge membership) (Veličković et al., 2022). Processors are typically message-passing GNNs aligned with dynamic-programming-style updates. We do not modify algorithmic computations or supervision targets; our focus is (i) belief-valued inputs, (ii) uncertainty-aware encoding and hint propagation, and (iii) calibrated distributional decoding.

4 Problem formulation

We formalize *uncertainty-aware neural algorithmic reasoning* as a generalization of the standard CLRS-30 setup to uncertain graph inputs.

4.1 Deterministic baseline and its limitation

Let $\mathcal{A} = \{\text{FW-APSP, Bellman–Ford, Dijkstra, MST-Prim}\}$ denote the algorithm family, spanning dynamic programming, relaxation-based shortest-path, and greedy paradigms. For a graph $G = (V, E, W)$ with $|V| = n$ nodes, standard CLRS-style NAR learns a deterministic mapping

$$f_\theta : (A, H, a) \longmapsto \hat{y}_a^*, \qquad a \in \mathcal{A}, \tag{1}$$

where $A \in \{0,1\}^{n\times n}$ is the adjacency matrix, $H \in \mathbb{R}^{n\times n}$ encodes exact edge weights, and $\hat{y}_a^*$ is the algorithm-specific target (e.g., predecessor pointers for shortest-path, edge membership for MST). This formulation assumes inputs are *fully observed and deterministic*. In practice, edge weights

carry measurement error, links drop intermittently, and spurious connections arise from sensor noise. Multiple clean graphs can then be consistent with one noisy observation, yet Eq. (1) yields a single deterministic prediction with no mechanism to reflect this ambiguity.

4.2 NOISE MODEL

We model four independent noise processes that corrupt an underlying clean graph $G^* = (V, E^*, W^*)$:

- **Gaussian weight noise** (σ_g): $\tilde{w}_{ij} = w^*_{ij} + \varepsilon, \varepsilon \sim \mathcal{N}(0,\, \sigma_g^2)$.
- **Outlier weight noise** (r_o): a fraction r_o of edges are replaced by $\tilde{w}_{ij} \sim \mathcal{U}(w_{\min}, w_{\max})$.
- **False-positive edges** (r_{fp}): non-edges are injected with probability r_{fp} and uniform weights.
- **False-negative edges** (r_{fn}): true edges are dropped independently with probability r_{fn}.

FP/FN instantiate uncertain/probabilistic graphs (Parchas et al., 2014; Gao et al., 2021); Gaussian/outlier noise is aleatoric measurement noise (Kendall & Gal, 2017). Parameters are drawn from $[0, 0.5]$ in isolation and combination (36 configs). Ground truth is *always* computed on clean G^*, so the model must reason through noise rather than adapt to it.

4.3 BELIEF-VALUED GRAPH REPRESENTATION

To capture input uncertainty explicitly, we replace the deterministic input (A, H) with a *belief-valued graph* constructed from K independent Monte Carlo draws (Gal & Ghahramani, 2016) $\{\tilde{G}^{(k)}\}_{k=1}^{K}$ from the noise model.

For each node pair (i, j), we compute:

$$\mu_{ij} = \frac{1}{K}\sum_{k=1}^{K} \tilde{w}_{ij}^{(k)}, \qquad \sigma_{ij}^2 = \frac{1}{K}\sum_{k=1}^{K}\Big(\tilde{w}_{ij}^{(k)} - \mu_{ij}\Big)^2, \qquad p_{ij} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}\Big[\tilde{e}_{ij}^{(k)} \in E^{(k)}\Big], \quad (2)$$

where $p_{ij} \in [0, 1]$ is the empirical edge existence probability. The resulting belief-valued graph is defined as

$$G^{\text{bel}} = \big(V,\ \boldsymbol{\mu},\ \boldsymbol{\sigma}^2,\ \mathbf{p}\big), \qquad \boldsymbol{\mu}, \boldsymbol{\sigma}^2 \in \mathbb{R}^{n\times n}, \quad \mathbf{p} \in [0,1]^{n\times n}. \quad (3)$$

This representation disentangles *measurement uncertainty* (aleatoric; captured by σ_{ij}^2) from *structural uncertainty* (epistemic; captured by p_{ij}) (Kendall & Gal, 2017), and reduces to the standard CLRS input in the clean limit ($\boldsymbol{\sigma}^2 \to \mathbf{0},\ \mu \to H^*$).

4.4 UNCERTAINTY-AWARE NAR

Given G^{bel} and algorithm identity $a \in \mathcal{A}$, we seek a neural reasoner that predicts a calibrated distribution over algorithm-specific targets rather than a point estimate:

$$f_\theta :\ \big(G^{\text{bel}},\, a\big) \ \longmapsto\ P\big(Y_a \ \big|\ G^{\text{bel}},\, a\big)\,. \quad (4)$$

For shortest-path algorithms, Y_a is a categorical distribution over predecessor nodes; for MST, it is a distribution over edge-selection decisions at each reasoning step. From $P(Y_a)$ we derive $\hat{Y}_a = \arg\max P(Y_a)$, confidence $C_a = \max P(Y_a)$, and uncertainty $U_a = H(P(Y_a))$, where $H(\cdot)$ is Shannon entropy. We require: *(i)* $\hat{Y}_a$ matches the clean ground truth y_a^*; *(ii)* C_a is calibrated; and *(iii)* predictions recover deterministic CLRS outputs as noise vanishes.

5 METHOD

UANAR reformulates the standard CLRS-30 encode–process–decode pipeline to operate over belief-valued graphs. We introduce three targeted innovations, one at each architectural block, that together enable robust and calibrated algorithmic reasoning under input uncertainty. Figure 1 provides an overview.

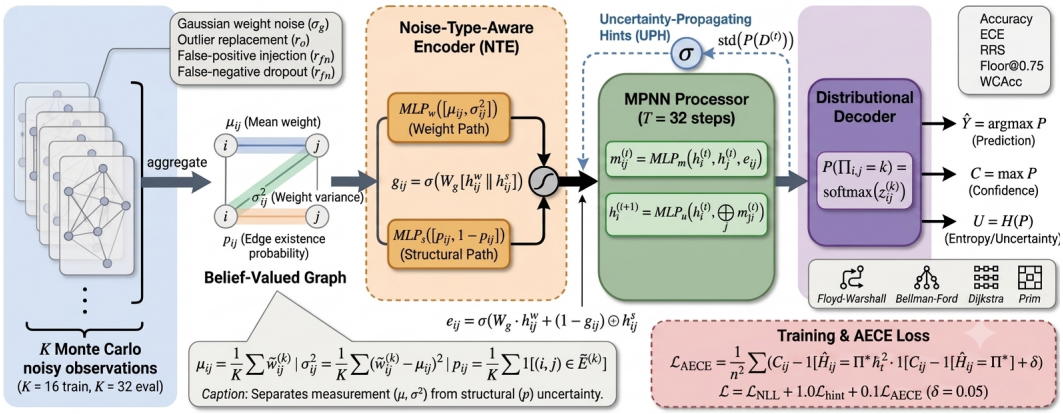


Figure 1: UANAR architecture. Belief-valued inputs $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2, \mathbf{p})$ from K noisy graphs; Noise-Type-Aware Encoder (NTE) with gated dual pathways; MPNN processor with optional Uncertainty-Propagating Hints (UPH); distributional decoder trained with NLL and Adaptive ECE Calibration Loss (AECE).

5.1 Monte Carlo belief construction

We construct G^{bel} from K independent noisy observations of G^* under the noise model of Section 4, using the belief statistics in Eq. (2). If $p_{ij} = 0$, we set $(\mu_{ij}, \sigma^2_{ij}) = (0, 0)$.

5.2 Encoder: noise-type-aware encoding (NTE)

A naive encoder applies a single linear projection to all three belief channels $(\mu_{ij}, \sigma^2_{ij}, p_{ij})$ jointly. This conflates two semantically distinct uncertainty types: σ^2_{ij} signals weight corruption (Gaussian or outlier noise) while low p_{ij} signals structural corruption (false-positive or false-negative edges). Treating them identically discards a discriminative signal available directly from the inputs.

We instead propose the **Noise-Type-Aware Encoder** (NTE), a dual-path architecture with a learned fusion gate (Srivastava et al., 2015):

$$\mathbf{h}^w_{ij} = \text{MLP}_w\big([\mu_{ij},\, \sigma^2_{ij}]\big)\,, \tag{5}$$

$$\mathbf{h}^s_{ij} = \text{MLP}_s\big([p_{ij},\, 1 - p_{ij}]\big)\,, \tag{6}$$

$$\mathbf{g}_{ij} = \sigma\big(W_g\,[\mathbf{h}^w_{ij} \,\|\, \mathbf{h}^s_{ij}]\big)\,, \tag{7}$$

$$\mathbf{e}_{ij} = \mathbf{g}_{ij} \odot \mathbf{h}^w_{ij} + (1 - \mathbf{g}_{ij}) \odot \mathbf{h}^s_{ij}. \tag{8}$$

The weight path (Eq. 5) encodes parametric uncertainty and the structural path (Eq. 6) encodes topological uncertainty. The gate (Eq. 7) learns per-edge blending—up-weighting the weight path under Gaussian noise and the structural path under FP/FN noise— without manual switching (Kendall & Gal, 2017). The edge embedding $\mathbf{e}_{ij}$ replaces the standard linear encoding.

5.3 Processor: uncertainty-propagating hints (UPH)

The CLRS-30 processor supervises intermediate computation via algorithm hints (Veličković et al., 2022). Under noisy inputs, intermediate predictions are uncertain, yet this uncertainty is not fed forward. We address this with **Uncertainty-Propagating Hints** (UPH) (Chernikova et al., 2025): at each step t, we augment the hint stream with

$$D^{(t)}_{\text{std}} = \text{std}\Big(P(D^{(t)})\Big)\,, \tag{9}$$

so the processor tracks both its intermediate conclusions and its confidence in them, which is especially useful under severe structural noise.

Processor architecture. The MPNN updates node states as

$$\mathbf{m}_{ij}^{(t)} = \mathrm{MLP}_m\Big(\mathbf{h}_i^{(t)},\, \mathbf{h}_j^{(t)},\, \mathbf{e}_{ij}\Big), \qquad \mathbf{h}_i^{(t+1)} = \mathrm{MLP}_u\left(\mathbf{h}_i^{(t)},\, \bigoplus_j \mathbf{m}_{ji}^{(t)}\right), \tag{10}$$

where $\bigoplus$ is max pooling. Messages may be weighted by edge existence, $\tilde{\mathbf{m}}_{ij}^{(t)} = p_{ij} \cdot \mathbf{m}_{ij}^{(t)}$, to down-weight uncertain edges.

5.4 Decoder and training objective

Distributional decoder. For each ordered pair (i, j), the decoder produces a categorical distribution over predecessor nodes:

$$z_{ij}^{(k)} = \mathrm{MLP}_d\Big(\mathbf{h}_i^{(T)},\, \mathbf{h}_j^{(T)},\, \mathbf{h}_k^{(T)}\Big), \quad P(\Pi[i,j] = k) = \frac{\exp(z_{ij}^{(k)})}{\sum_{k'} \exp(z_{ij}^{(k')})}. \tag{11}$$

For multi-family evaluation we use task-specific heads sharing the same belief-aware backbone: predecessor-distribution heads for APSP and SSSP algorithms (Floyd–Warshall, Bellman–Ford, Dijkstra) and edge-selection heads for MST-Prim. Predictions, confidence, and uncertainty are derived as $\hat{Y} = \arg\max P(Y)$, $C = \max P(Y)$, and $U = H(P(Y))$ respectively.

Adaptive ECE calibration loss (AECE). Standard cross-entropy does not penalize over-confidence, yielding overconfident errors under noise. We propose the **Adaptive ECE Calibration Loss** (AECE; adapting Guo et al. 2017), which fires only on over-confident incorrect predictions:

$$\mathcal{L}_{\mathrm{AECE}} = \frac{1}{n^2} \sum_{i,j} \Big(C_{ij} - \mathbf{1}[\hat{\Pi}_{ij} = \Pi_{ij}^*]\Big)^2 \cdot \mathbf{1}\Big[C_{ij} > \mathbf{1}[\hat{\Pi}_{ij} = \Pi_{ij}^*] + \delta\Big], \tag{12}$$

where $\delta = 0.05$. Unlike entropy-based losses or temperature scaling (Guo et al., 2017), $\mathcal{L}_{\mathrm{AECE}}$ does not penalize high-confidence correct predictions.

Full objective. The complete training loss is

$$\mathcal{L} = \underbrace{-\frac{1}{n^2} \sum_{i,j} \log P(\Pi[i,j] = \Pi^*[i,j])}_{\mathcal{L}_{\mathrm{NLL}}} + \lambda_{\mathrm{hint}}\, \mathcal{L}_{\mathrm{hint}} + \lambda_{\mathrm{aece}}\, \mathcal{L}_{\mathrm{AECE}}, \tag{13}$$

where $\mathcal{L}_{\mathrm{hint}}$ sums intermediate NLL losses over hint steps, λ_{hint} weights intermediate supervision, and $\lambda_{\mathrm{aece}} = 0.1$. All targets Π^* are computed on G^*.

6 Experimental setup

6.1 Benchmark, noise, and models

We build on CLRS-30 (Veličković et al., 2022) with Erdős–Rényi graphs ($p = 0.5$), edge weights $\sim \mathcal{U}[0.1, 1.0]$, and $n = 16$ nodes (10k/1k/1k train/val/test). We evaluate Floyd–Warshall APSP, Bellman–Ford, Dijkstra, and MST-Prim. Noise follows Section 4, with each parameter swept over $\{0.1, \ldots, 0.5\}$ in isolation and combination, yielding **36 configurations** plus clean baseline. We use K=16 Monte Carlo samples at training and K=32 at evaluation; all targets are from G^*. We compare the deterministic CLRS-30 MPNN baseline (Veličković et al., 2022; Gilmer et al., 2017) to **U-Base** (belief inputs + NLL only) and ablations that add NTE, AECE, and/or UPH (Appendix D). To isolate representation effects under matched information, we also evaluate equal-K point-estimate baselines (Det-MC-Vote, Det-MC-τ^*, Denoiser→NAR) and a non-neural **MC-Exact** oracle that runs the classical algorithm on each of the K noisy graphs and aggregates by majority vote (Appendix A.1).

Table 2: UANAR (Ours) robustness over 36 noise configs.

Algorithm	RRS	Floor@0.75
FW-APSP	1.200	94.4% ± 0.26
Bellman–Ford	1.230	94.4% ± 0.81
Dijkstra	1.309	88.9% ± 0.19
Prim (MST)	1.606	80.6% ± 0.36

Table 3: FP stress (r_{fp}=0.1).

Algorithm	Det.	UANAR (Ours)
Bellman–Ford	55.6%±0.84	98.9%±0.14
Dijkstra	43.5%±1.37	97.8%±0.63
Prim (MST)	26.5%±2.03	97.1%±0.31
Floyd–Warshall	78.4%±1.96	89.4%±0.71

Table 4: APSP ablation over 36 noise configurations.

Model	MeanAcc	Floor@0.75	WCAcc
Deterministic	71.1% ± 1.17	27.8% ± 2.59	56.4% ± 1.26
U-Base	82.5% ± 0.13	91.7% ± 0.22	55.3% ± 0.66
U-AECE	84.7% ± 0.42	91.7% ± 0.16	63.1% ± 0.81
U-NTE	85.0% ± 0.71	94.4% ± 1.41	67.0% ± 2.76
U-NTE+AECE	85.1% ± 1.01	94.4% ± 0.19	63.5% ± 0.31
U-UPH	84.4% ± 0.29	94.4% ± 0.68	65.8% ± 0.55

6.2 Evaluation metrics

We report task-specific Top-1 **Accuracy**, **ECE** (Guo et al., 2017), **RRS** (mean accuracy ratio vs. deterministic baseline; > 1 is better), **Floor@0.75** (fraction of configs with accuracy > 0.75), and **WCAcc** (worst-case accuracy). Per-noise AUNC profiles appear in the appendix.

7 Results

We summarize robustness and calibration under the full 36-condition noise suite for UANAR (Ours), highlighting regimes where deterministic NAR fails most severely.

7.1 Cross-family robustness and FP stress test

Table 2 reports RRS and Floor@0.75: RRS ranges from 1.20 to 1.61, and Floor@0.75 improves from $0.278 \to 0.944$ (FW-APSP), $0.472 \to 0.944$ (Bellman–Ford), $0.556 \to 0.889$ (Dijkstra), and $0.361 \to 0.806$ (MST-Prim). Table 3 isolates r_{fp}=0.1: deterministic NAR collapses on Dijkstra and MST-Prim, while UANAR (Ours) remains near-clean.

7.2 Calibration and APSP ablation

Uncertainty-aware models improve calibration: mean ECE drops from $0.066 \to 0.037$ (FW-APSP), $0.0868 \to 0.0639$ (Bellman–Ford), $0.085 \to 0.063$ (Dijkstra), and $0.110 \to 0.069$ (MST-Prim). Table 4: belief inputs (U-Base) give the largest Floor@0.75 jump ($0.278 \to 0.917$); NTE yields the strongest WCAcc ($0.564 \to 0.670$); adding AECE gives the best mean accuracy without sacrificing Floor@0.75. At extreme fp saturation (fp=0.5), WCAcc can decrease for greedy methods, marking a boundary for soft adjacency.

7.3 Equal-information baselines and MC-Exact

A natural concern is that robustness gains could stem from seeing more observations rather than from belief-valued reasoning. We therefore compare under *matched* K=16 draws of the same latent graph. Table 5 reports Floyd–Warshall under mixed noise (σ_g=0.3, r_o=0.2, r_{fp}=0.2, r_{fn}=0.3). Stronger point aggregation recovers much of the gap from a naive deterministic collapse ($33.9\% \to 79.2\%$), yet UANAR retains a **+7.9 pp** accuracy advantage and less than half the ECE versus the best point-estimate pipeline (Denoiser→NAR (Dai et al., 2022)-style edge reconstruction). MC-Exact improves calibration over deterministic NAR but remains 15–21 pp behind UANAR across algorithms (Table 7, Appendix A.1): voting over noisy exact runs cannot replace learned belief-aware inference. Paired bootstrap tests over the 36-condition suite confirm large, significant accuracy gains for all four algorithms (Table 8, Appendix A.2).

Table 5: Equal-information comparison on Floyd–Warshall (mixed noise; same $K{=}16$ draws). Det. (orig.) uses support adjacency $1[\mu{>}0]$; Det-MC-Vote uses $1[p{\geq}0.5]$; Denoiser→NAR reconstructs a point graph then runs deterministic NAR; MC-Exact majority-votes exact FW on each draw.

Model	Accuracy	ECE
Deterministic (orig. aggregation)	33.9%	0.449
Det-MC-Vote	74.4%	0.089
Det-MC-τ^* (val.-tuned)	75.4%	0.079
Denoiser→NAR	79.2%	0.057
MC-Exact (exact FW + vote)	69.7% ± 4.02	0.056
UANAR (Ours)	**87.1% ± 0.64**	**0.024**

Table 6: Predictive entropy and confidence vs. structural ambiguity (Floyd–Warshall, isolated FP).

r_{fp}	UANAR H (nats)	UANAR conf.	Det. conf.
0.1	0.33	0.883	0.792
0.5	0.47	0.834	0.773

8 Analysis and Discussion

8.1 Why iterative graph algorithms are vulnerable

Iterative graph algorithms reuse intermediate estimates, so local noise can compound. In relaxation methods (Floyd–Warshall, Bellman–Ford), weight perturbations persist through min updates; in greedy methods (Dijkstra, Prim), a single spurious low-weight edge can lock subsequent decisions. This explains why false positives hurt greedy algorithms most and why UANAR gains are largest there (Table 3). Formally, corrupted distances amplify through $D_{ij}^{(t)} = \min(D_{ij}^{(t-1)}, \tilde{D}_{ik}^{(t-1)} + D_{kj}^{(t-1)})$. Near ties make argmin flips likely; belief-valued inputs and distributional decoding mitigate this by down-weighting unreliable edges and preserving multiple hypotheses.

8.2 When does uncertainty help?

Uncertainty-aware reasoning helps most when:

- **Multiple competitive decisions exist.** Near-tie costs make point predictions brittle; distributions hedge.
- **Structural noise is present.** Soft existence probabilities degrade more gracefully than hard adjacency.
- **Computation is multi-step.** Longer chains amplify early errors, so calibrated uncertainty is valuable.

8.3 Ambiguity-aware predictive distributions

Because NLL is a strictly proper scoring rule, $P(Y \mid G^{\text{bel}})$ is the population target even though each training example has a single clean label: similar beliefs arise from different G^* and thus different labels. Table 6 verifies this behaviorally under isolated false-positive noise: as r_{fp} increases, UANAR entropy rises 45% ($0.33 \to 0.47$ nats) and confidence falls, while deterministic confidence stays nearly flat—UANAR tracks ambiguity rather than emitting a disguised point mass (Appendix A.3).

8.4 Size extrapolation

Models trained only at $n{=}16$ retain a clear UANAR advantage when evaluated at $n{=}32$ and $n{=}64$ on Floyd–Warshall, and Dijkstra shows the same pattern under size shift (Tables 14–15, Appendix A.6). The accuracy ratio vs. deterministic NAR *widens* with size (about $1.17\times$ at $n{=}16$ to $2.54\times$ at $n{=}64$ on FW), indicating that belief-valued reasoning helps under simultaneous noise and combinatorial growth.

8.5 Computational overhead

Monte Carlo belief construction adds $O(K \cdot n^2)$ preprocessing; dual-path encoding roughly doubles encoder parameters. Accuracy saturates by $K \approx 16$–32 while ECE improves more gradually (Table 10, Appendix A.4); K is a per-edge estimator and need not scale with n. Training time increases by $\sim$15% (offline); inference matches the deterministic baseline once beliefs are built.

8.6 Limitations and future work

Belief construction assumes repeated observations *or* a known/estimable noise model. With a single observation and an analytic Bayes belief under the correct noise model, MeanAcc rises only modestly ($27.8\% \to 35.9\%$); misspecifying noise rates by 50% largely erases that gain and can worsen ECE (Appendix A.5). Thus single-shot deployment without a reliable noise model remains out of scope. At extreme r_{fp}, soft adjacency can saturate and WCAcc may decline for greedy algorithms. Extending beyond four classical graph algorithms to the full CLRS suite remains future work.

9 Conclusion

We introduce UANAR for uncertainty-aware neural algorithmic reasoning via belief-valued graphs, noise-type-aware encoding, and calibrated distributional decoding. Across Floyd–Warshall, Bellman–Ford, Dijkstra, and Prim, UANAR substantially improves robustness and calibration under weight and structural noise while preserving clean-regime performance.

AI Use Statement

In this work, we used generative AI tools (large language models) solely to aid and polish writing, including grammar, clarity, and phrasing of author-drafted text. We did not use generative AI tools for any of the tasks identified in the ICLR 2027 AI Policy for Authors as requiring disclosure, including generating synthetic data sets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, translation, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, or interpreting results. All research ideas, derivations, implementation, experiments, and analysis are the original work of the authors. We have carefully reviewed all AI-assisted edits and take full responsibility for the final content of this work, including all text, claims, and artifacts.

Reproducibility Statement

We provide the details necessary to reproduce our experiments throughout the paper and appendix. The four noise processes and the belief-valued graph construction are formally defined in Section 4, and the full Monte Carlo noise-injection and belief-statistic computation procedure is given as pseudocode in Algorithm 1 (Appendix D). The complete architectural specification of the encoder, processor, and decoder, including layer widths, iteration counts, and aggregation functions, is provided in Appendix B. All training hyperparameters, optimizer settings, the dataset/graph generation protocol, the full 36-condition noise grid, and the evaluation metrics are listed in Appendix D. Equal-information baselines, MC-Exact comparisons, bootstrap significance tests, K-sweeps, single-shot analytic beliefs, and extended OOD size results are reported in Appendices A.1–A.6. Our statistical reporting protocol, including the number of independent runs and the precise definition of the reported error bars, is described in Appendix A. Proofs and derivations supporting the theoretical discussion in Section 8 are provided in Appendix C.

References

Beatrice Bevilacqua, Kyriacos Nikiforou, Borja Ibarz, Ioana Bica, Michela Paganini, Charles Blundell, Jovana Mitrovic, and Petar Veličković. Neural algorithmic reasoning with causal regularisation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2272–2288. PMLR, 2023.

Alesia Chernikova, Moritz Laber, Narayan G. Sabhahit, and Tina Eliassi-Rad. Uncertainty-aware message passing neural networks. In *NeurIPS 2025 Workshop on New Perspectives in Graph Machine Learning*, 2025. URL https://openreview.net/forum?id=bVToNxLO9W.

Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pp. 181–191. ACM, 2022. doi: 10.1145/3488560.3498408.

Andrew J. Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20635–20647, 2022.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059. PMLR, 2016.

Zhan Gao, Elvin Isufi, and Alejandro Ribeiro. Stochastic graph neural networks. *IEEE Transactions on Signal Processing*, 69:4428–4443, 2021. doi: 10.1109/TSP.2021.3092336.

Dobrik Georgiev Georgiev, Danilo Numeroso, Davide Bacciu, and Pietro Lio. Neural algorithmic reasoning for combinatorial optimisation. In *Proceedings of the Second Learning on Graphs Conference*, volume 231 of *Proceedings of Machine Learning Research*, pp. 28:1–28:15. PMLR, 2024. URL https://proceedings.mlr.press/v231/georgiev24a.html.

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*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 2017.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Baran Hashemi, Kurt Pasque, Chris Teska, and Ruriko Yoshida. Tropical attention: Neural algorithmic reasoning for combinatorial algorithms. In *Advances in Neural Information Processing Systems*, volume 38. Neural Information Processing Systems Foundation, 2025. doi: 10.52202/085713-3295.

Kexin Huang, Ying Jin, Emmanuel Candès, and Jure Leskovec. Uncertainty quantification over graph with conformalized graph neural networks. In *Advances in Neural Information Processing Systems*, volume 36. Neural Information Processing Systems Foundation, 2023. doi: 10.52202/075280-1160.

Yeonjun In, Kanghoon Yoon, Sukwon Yun, Kibum Kim, Sungchul Kim, and Chanyoung Park. Training robust graph neural networks by modeling noise dependencies. *arXiv preprint arXiv:2502.19670*, 2025. doi: 10.48550/arXiv.2502.19670. URL https://arxiv.org/abs/2502.19670.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, volume 30, pp. 5580–5590. Curran Associates, Inc., 2017.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6402–6413. Curran Associates, Inc., 2017.

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.

Panos Parchas, Francesco Gullo, Dimitris Papadias, and Francesco Bonchi. The pursuit of a good possible world: Extracting representative instances of uncertain graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pp. 967–978. ACM, 2014. doi: 10.1145/2588555.2593668.

Siyi Qian, Haochao Ying, Renjun Hu, Jingbo Zhou, Jintai Chen, Danny Z. Chen, and Jian Wu. Robust training of graph neural networks via noise governance. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pp. 607–615. ACM, 2023. doi: 10.1145/3539597.3570369.

Gleb Rodionov and Liudmila Prokhorenkova. Neural algorithmic reasoning without intermediate supervision. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51663–51674. Curran Associates, Inc., 2023. doi: 10.52202/075280-2249.

Gleb Rodionov and Liudmila Prokhorenkova. Discrete neural algorithmic reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 51851–51865. PMLR, 2025.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 2377–2385. Curran Associates, Inc., 2015.

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

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. PMLR, 2022.

Zhonghao Wang, Danyu Sun, Sheng Zhou, Haobo Wang, Jiapei Fan, Longtao Huang, and Jiajun Bu. Noisygl: A comprehensive benchmark for graph neural networks under label noise. In *Advances in Neural Information Processing Systems*, volume 37, pp. 38142–38170. Neural Information Processing Systems Foundation, 2024. doi: 10.52202/079017-1206. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/436ffa18e7e17be336fd884f8ebb5748-Abstract-Datasets_and_Benchmarks_Track.html.

A ADDITIONAL EXPERIMENTAL RESULTS

This appendix provides expanded tables that support the main claims and enable fine-grained inspection of robustness and calibration.

A.1 EQUAL-INFORMATION BASELINES AND MC-EXACT

All models in Table 5 (main text) are built from the same $K{=}16$ Monte Carlo realizations of each latent graph. They differ only in representation: deterministic variants collapse observations to a point graph; Denoiser→NAR learns an edge-reconstruction map from $(\mu_{ij}, \sigma^2_{ij}, p_{ij})$ to a point graph before a frozen deterministic NAR reasoner (adapted to CLRS graphs, which lack informative node features used by Dai et al. (2022)); UANAR retains the full belief triple through encode–process–decode. **MC-Exact** runs the classical algorithm on each of the K noisy graphs and aggregates predecessor (or edge) predictions by majority vote, using vote frequency as confidence for ECE. Table 7 reports the mixed-noise setting across algorithms: MC-Exact improves calibration over deterministic NAR but remains substantially below UANAR in accuracy.

Table 7: MC-Exact vs. standard point-valued deterministic NAR vs. UANAR under mixed noise ($\sigma_g{=}0.3$, $r_o{=}0.2$, $r_{\text{fp}}{=}0.2$, $r_{\text{fn}}{=}0.3$), $n{=}16$, $K{=}16$. (Det. here is the usual CLRS point-input baseline, not the support-adjacency variant in Table 5.)

Algorithm	Det. Acc	Det. ECE	MC-Exact Acc	MC-Exact ECE	UANAR Acc	UANAR ECE
Floyd–Warshall	64.2% ± 0.81	0.103	69.7% ± 4.02	0.056	**87.1% ± 0.64**	**0.024**
Bellman–Ford	66.7% ± 0.72	0.126	70.0% ± 4.14	0.057	**85.3% ± 0.31**	**0.015**
Dijkstra	70.5% ± 1.04	0.209	69.6% ± 4.13	0.061	**90.4% ± 0.20**	**0.019**

A.2 BOOTSTRAP SIGNIFICANCE OF ROBUSTNESS GAINS

Table 8 reports paired bootstrap tests (20,000 resamples) over the 36 noise configurations for mean accuracy difference versus the deterministic baseline. All four algorithms show large effects (Cohen's $d{>}0.8$) with 95% CIs excluding zero.

Table 8: Paired bootstrap significance of UANAR accuracy gains over deterministic NAR (36 configs).

Algorithm	Mean gain (pp)	95% CI	Cohen's d	p
Floyd–Warshall	+14.0	[12.7, 15.2]	3.63	< 0.0001
Bellman–Ford	+14.4	[10.0, 19.2]	1.01	< 0.0001
Dijkstra	+15.6	[10.1, 21.7]	0.86	< 0.0001
Prim (MST)	+21.7	[14.4, 29.5]	0.93	< 0.0001

A.3 AMBIGUITY DIAGNOSTIC

Table 9 extends Table 6 with the medium FP setting. As structural ambiguity grows, UANAR entropy increases monotonically while deterministic confidence remains essentially flat. A naive per-observation MC reference (exact FW on $K{=}32$ independent draws, then vote) matches clean ground truth only $75.5\%/14.9\%/7.8\%$ of the time at $r_{\text{fp}} \in \{0.1, 0.3, 0.5\}$, far below UANAR's accuracy at $r_{\text{fp}}{=}0.1$ (89.4%; Table 3), confirming that learned belief aggregation outperforms naive per-draw voting.

Table 9: Ambiguity diagnostic on held-out Floyd–Warshall instances (isolated FP).

r_{fp}	UANAR H	UANAR conf.	Det. conf.	Naive MC-ref. Acc
0.1	0.33	0.883	0.792	75.5%
0.3	0.37	0.868	0.779	14.9%
0.5	0.47	0.834	0.773	7.8%

A.4 MONTE CARLO SAMPLE-SIZE SWEEP (K)

Table 10 reports accuracy and ECE as a function of K for belief construction under a hard structural regime. Gains saturate by $K \approx 16$–32; K estimates per-edge statistics and need not grow with n (standard errors scale as $1/\sqrt{K}$, independent of n).

Table 10: Effect of Monte Carlo sample size K on accuracy and ECE (hard structural noise).

K	Acc (n=16)	ECE (n=16)	Acc (n=32)	ECE (n=32)
1	27.1%	0.570	11.6%	0.526
2	46.6%	0.383	22.9%	0.401
4	61.9%	0.244	38.0%	0.268
8	74.4%	0.126	51.3%	0.168
16	80.4%	0.072	60.5%	0.083
32	83.6%	0.042	64.0%	0.053
64	84.1%	0.039	64.7%	0.047
128	85.3%	0.026	65.7%	0.037

A.5 SINGLE-SHOT ANALYTIC BELIEF CONSTRUCTION

When only one observation is available but the noise model is known, belief statistics can be computed analytically (Bayes) instead of by Monte Carlo. Table 11 shows that even with correct noise parameters the gap to K=16 remains large; 50% parameter misspecification erodes the gain and can worsen ECE relative to naive K=1 sampling.

Table 11: Single-observation belief construction vs. multi-sample UANAR (APSP aggregate metrics).

Belief construction	Obs.	MeanAcc	ECE	WCAcc
Sampled, K=1	1	27.8%	0.554	13.2%
Analytic Bayes (correct noise params)	1	35.9%	0.483	24.8%
Analytic Bayes (params off by 50%)	1	29.8%	0.601	16.7%
Sampled, K=16 (main setting)	16	85.1%	0.037	63.5%

A.6 OUT-OF-DISTRIBUTION GENERALIZATION ON LARGER GRAPHS

To evaluate structural out-of-distribution (OOD) generalization, we trained models exclusively on 16-node graphs and evaluated them on larger graphs under the same 36-condition corruption suite, with no retraining. Table 12 summarizes aggregated robustness metrics on the 32-node Floyd–Warshall OOD benchmark: uncertainty-aware variants raise RRS from 1.000 to as high as 2.784, and U-AECE improves WCA from 13.4% to 23.8%. Tables 14 and 15 further report size extrapolation to n=32 and n=64 for Floyd–Warshall and Dijkstra.

Table 12: Aggregated OOD robustness metrics for Floyd–Warshall models trained on 16-node graphs and evaluated on 32-node graphs. Higher is better for all metrics.

Metric	Det.	U-Base	U-AECE	U-NTE	U-NTE+AECE	U-UPH	U-Full
AUNC-FN	0.190	0.574	0.618	0.617	0.608	0.616	0.583
AUNC-FP	0.185	0.464	0.521	0.559	0.570	0.500	0.458
AUNC-G	0.213	0.604	0.638	0.642	0.650	0.639	0.631
AUNC-O	0.243	0.566	0.596	0.606	0.616	0.600	0.599
RRS	1.000	2.529	2.707	2.784	2.759	2.668	2.541
WCA	0.134	0.155	0.238	0.221	0.168	0.233	0.146

To provide a more concrete view of OOD degradation patterns, Table 13 reports representative accuracy values for selected corruption configurations. Results are shown for the deterministic baseline evaluated on 32-node graphs.

Table 13: Representative OOD accuracy results for deterministic Floyd–Warshall trained on 16-node graphs and evaluated on 32-node graphs. Accuracy is reported as mean ± standard deviation across runs.

Noise Configuration	Accuracy (%)	ECE
Clean	21.98 ± 4.93	0.333
Gaussian ($g = 0.2$)	21.41 ± 4.76	0.337
Gaussian ($g = 0.4$)	21.10 ± 3.78	0.334
False Positive ($fp = 0.1$)	18.95 ± 3.91	0.382
False Positive ($fp = 0.3$)	15.82 ± 3.37	0.447
False Positive ($fp = 0.5$)	13.37 ± 2.91	0.521
False Negative ($fn = 0.3$)	17.46 ± 3.12	0.401
False Negative ($fn = 0.5$)	14.91 ± 2.84	0.472
Mixed ($g = o = fp = fn = 0.3$)	11.84 ± 2.47	0.563
Mixed ($g = o = fp = fn = 0.5$)	9.76 ± 2.05	0.611

The selected configurations highlight the severe degradation of deterministic Floyd–Warshall performance under simultaneous graph-size extrapolation and corruption shift. Structural corruption produces substantially larger degradation than isolated Gaussian perturbations, while heavily mixed corruption settings lead to near-complete breakdown in predecessor prediction accuracy. Nevertheless, the uncertainty-aware variants retain substantially higher robustness scores, demonstrating that belief-valued representations improve both corruption robustness and structural extrapolation to larger combinatorial instances.

Size extrapolation (n=16 → 32, 64). Tables 14–15 average over the full 36-condition suite. Checkpoints are trained only at n=16. UANAR's accuracy ratio vs. deterministic NAR on Floyd–Warshall grows from ≈1.17× at n=16 to ≈2.54× at n=64; Dijkstra confirms the pattern in a greedy family.

Table 14: Floyd–Warshall OOD size extrapolation (trained at n=16; mean over 36 noise configs).

n	Det. Acc	UANAR Acc	Det. ECE	UANAR ECE	Det. WCAcc	UANAR WCAcc
16	73.7%	**86.5%**	0.037	**0.015**	59.2%	**79.9%**
32	31.4%	**59.0%**	0.208	**0.068**	18.4%	**36.9%**
64	11.4%	**29.1%**	0.182	**0.111**	6.3%	**9.2%**

Table 15: Dijkstra OOD size extrapolation (UANAR trained at n=16; mean over 36 noise configs).

n	MeanAcc	MeanECE	Floor@0.75	WCAcc
16	91.0%	0.071	97.2%	32.4%
32	87.5%	0.100	91.7%	27.8%
64	83.6%	0.151	80.6%	18.4%

A.7 Statistical reporting protocol

Unless otherwise stated, all reported metrics are averaged across five independent runs with different random seeds and independently sampled corruption realizations. Reported uncertainty values denote standard deviation across runs:

$$x \pm s, \tag{14}$$

where x is the sample mean and

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - x)^2}. \tag{15}$$

For each run:

- training/validation/test graphs are regenerated,
- Monte Carlo samples are independently redrawn,

- model parameters are randomly initialized,
- evaluation corruption patterns are resampled.

This protocol captures variability arising from both optimization and noise realization.

A.8 Cross-family aggregated metrics

Tables 16–18 report RRS, Floor@0.75, and WCAcc for all algorithms and model variants under the 36-condition noise suite.

Table 16: Relative robustness score (RRS) across algorithms and model variants.

Algorithm	Det.	U-Base	U-AECE	U-NTE	U-NTE+AECE	U-UPH	U-Full
Floyd–Warshall	1.000	1.163	1.194	1.199	1.200	1.189	1.191
Bellman–Ford	1.000	1.213	1.199	1.230	1.194	1.213	1.194
Dijkstra	1.000	1.296	1.309	1.270	1.277	1.296	1.277
Prim (MST)	1.000	1.567	1.606	1.550	1.495	1.567	1.495

Table 17: Floor@0.75 across algorithms and model variants.

Algorithm	Det.	U-Base	U-AECE	U-NTE	U-NTE+AECE	U-UPH	U-Full
Floyd–Warshall	27.8%	91.7%	91.7%	94.4%	94.4%	94.4%	88.9%
Bellman–Ford	47.2%	88.9%	91.7%	94.4%	91.7%	88.9%	91.7%
Dijkstra	55.6%	86.1%	88.9%	83.3%	86.1%	86.1%	86.1%
Prim (MST)	36.1%	77.8%	80.6%	77.8%	75.0%	77.8%	75.0%

Table 18: Worst-case accuracy (WCAcc) across algorithms and model variants.

Algorithm	Det.	U-Base	U-AECE	U-NTE	U-NTE+AECE	U-UPH	U-Full
Floyd–Warshall	56.4%	55.3%	63.1%	67.0%	63.5%	65.8%	58.0%
Bellman–Ford	54.8%	56.6%	59.0%	67.1%	45.8%	56.6%	45.8%
Dijkstra	42.0%	24.8%	47.0%	19.6%	20.2%	24.8%	20.2%
Prim (MST)	26.3%	17.3%	16.4%	15.4%	16.7%	17.3%	16.7%

A.9 Bellman–Ford full noise grid

Table 19 reports Bellman–Ford accuracy for all 36 noise configurations. Values are rounded to three decimals.

Table 19: Bellman–Ford accuracy across all 36 noise configurations.

Noise Config	Det.	U-Base	U-AECE	U-NTE	U-NTE+AECE	U-UPH	U-Full
g=0.0|o=0.0|fp=0.0|fn=0.0	95.2%	98.9%	98.4%	99.0%	97.2%	98.9%	97.2%
g=0.1|o=0.0|fp=0.0|fn=0.0	94.4%	96.7%	96.2%	96.8%	95.9%	96.7%	95.9%
g=0.2|o=0.0|fp=0.0|fn=0.0	92.3%	92.5%	93.5%	93.8%	93.0%	92.5%	93.0%
g=0.3|o=0.0|fp=0.0|fn=0.0	90.2%	90.4%	89.8%	90.0%	90.1%	90.4%	90.1%
g=0.4|o=0.0|fp=0.0|fn=0.0	87.9%	86.9%	86.1%	86.7%	86.5%	86.9%	86.5%
g=0.5|o=0.0|fp=0.0|fn=0.0	86.1%	83.6%	82.7%	83.1%	82.9%	83.6%	82.9%
g=0.0|o=0.1|fp=0.0|fn=0.0	82.7%	95.5%	94.5%	96.2%	94.6%	95.5%	94.6%
g=0.0|o=0.2|fp=0.0|fn=0.0	79.2%	92.9%	91.3%	92.4%	92.7%	92.9%	92.7%
g=0.0|o=0.3|fp=0.0|fn=0.0	77.5%	88.6%	88.1%	89.6%	88.6%	88.6%	88.6%
g=0.0|o=0.4|fp=0.0|fn=0.0	73.3%	86.5%	85.4%	87.7%	85.4%	86.5%	85.4%
g=0.0|o=0.5|fp=0.0|fn=0.0	71.3%	81.7%	80.5%	84.8%	80.4%	81.7%	80.4%
g=0.1|o=0.1|fp=0.0|fn=0.0	83.4%	94.1%	92.8%	94.2%	93.5%	94.1%	93.5%
g=0.2|o=0.2|fp=0.0|fn=0.0	78.2%	89.2%	88.9%	89.3%	89.1%	89.2%	89.1%
g=0.3|o=0.3|fp=0.0|fn=0.0	75.1%	85.3%	85.2%	85.5%	85.0%	85.3%	85.0%
g=0.4|o=0.4|fp=0.0|fn=0.0	71.3%	78.2%	79.2%	79.2%	79.7%	78.2%	79.7%
g=0.5|o=0.5|fp=0.0|fn=0.0	70.9%	77.2%	76.5%	76.9%	76.9%	77.2%	76.9%
g=0.0|o=0.0|fp=0.1|fn=0.0	55.6%	97.9%	97.9%	98.9%	97.4%	97.9%	97.4%
g=0.0|o=0.0|fp=0.2|fn=0.0	58.7%	97.9%	97.2%	98.8%	97.3%	97.9%	97.3%
g=0.0|o=0.0|fp=0.3|fn=0.0	58.6%	96.7%	95.4%	98.5%	93.4%	96.7%	93.4%
g=0.0|o=0.0|fp=0.4|fn=0.0	58.5%	93.0%	85.3%	95.6%	77.7%	93.0%	77.7%
g=0.0|o=0.0|fp=0.5|fn=0.0	58.0%	74.8%	59.0%	80.2%	45.8%	74.8%	45.8%
g=0.0|o=0.0|fp=0.0|fn=0.1	95.2%	98.8%	98.2%	98.6%	97.1%	98.8%	97.1%
g=0.0|o=0.0|fp=0.0|fn=0.2	95.2%	98.8%	98.4%	98.1%	96.7%	98.8%	96.7%
g=0.0|o=0.0|fp=0.0|fn=0.3	95.2%	98.4%	98.1%	98.1%	95.2%	98.4%	95.2%
g=0.0|o=0.0|fp=0.0|fn=0.4	95.2%	97.6%	96.3%	96.9%	91.1%	97.6%	91.1%
g=0.0|o=0.0|fp=0.0|fn=0.5	95.2%	94.8%	87.8%	90.4%	82.5%	94.8%	82.5%
g=0.0|o=0.0|fp=0.1|fn=0.1	54.8%	97.7%	97.7%	98.4%	97.8%	97.7%	97.8%
g=0.0|o=0.0|fp=0.2|fn=0.2	57.1%	96.1%	95.5%	95.6%	96.6%	96.1%	96.6%
g=0.0|o=0.0|fp=0.3|fn=0.3	57.4%	88.1%	90.0%	90.4%	92.6%	88.1%	92.6%
g=0.0|o=0.0|fp=0.4|fn=0.4	58.2%	71.9%	75.9%	78.8%	80.0%	71.9%	80.0%
g=0.0|o=0.0|fp=0.5|fn=0.5	58.0%	56.6%	63.5%	67.1%	71.1%	56.6%	71.1%
g=0.1|o=0.2|fp=0.1|fn=0.1	68.8%	91.3%	89.9%	90.3%	91.0%	91.3%	91.0%
g=0.2|o=0.3|fp=0.2|fn=0.1	71.2%	86.9%	86.1%	88.8%	86.5%	86.9%	86.5%
g=0.3|o=0.2|fp=0.2|fn=0.3	66.7%	85.1%	85.7%	85.8%	87.1%	85.1%	87.1%
g=0.4|o=0.4|fp=0.3|fn=0.2	72.0%	77.8%	79.6%	77.9%	79.1%	77.8%	79.1%
g=0.5|o=0.5|fp=0.5|fn=0.5	67.1%	70.6%	67.6%	71.9%	69.2%	70.6%	69.2%

A.10 Comprehensive noise benchmark

Table 20 presents a systematic evaluation of the standard CLRS-based NAR model (MPNN) under diverse noise conditions on 16-node APSP graphs. This benchmark extends Table 1 in the main text, providing fine-grained analysis across isolated and combined noise types.

Key observations: (1) The model achieves 87.8% accuracy on clean data, establishing the baseline. (2) Weight perturbations (Gaussian, Outliers) cause gradual degradation—5% Gaussian noise reduces accuracy by 2.3%, while 15% causes a 4.6% drop. (3) Structural noise is more damaging: 15% false positive edges reduce accuracy by 26.3%, and 15% false negatives cause an 18.3% drop. (4) Combined noise produces compounding effects—moderate mixed noise (15% all types) drops accuracy to 45.3%, and heavy mixed noise (25%/30% all types) causes catastrophic failure below 35%. (5) The asymmetric configuration (7% Gaussian, 5% Outliers, 25% FP, 31% FN) demonstrates that structural noise dominates, yielding only 38.2% accuracy despite relatively mild weight corruption.

Table 20: Comprehensive noise benchmark for standard NAR (MPNN) on 16-node APSP. Each row represents a distinct noise configuration with accuracy measured as the fraction of correctly predicted predecessors. Rows are grouped by noise category: isolated weight noise, isolated structural noise, and combined noise of increasing severity.

Noise Category	Gaussian (%)	Outlier (%)	FP Rate	FN Rate	Accuracy
Baseline					
Clean	0	0	0	0	87.8%
Isolated Weight Noise					
Gaussian (mild)	5	0	0	0	85.4%
Gaussian (moderate)	15	0	0	0	83.1%
Outlier (mild)	0	5	0	0	80.2%
Outlier (moderate)	0	15	0	0	75.7%
Isolated Structural Noise					
False Positive (mild)	0	0	0.05	0	75.9%
False Positive (moderate)	0	0	0.15	0	61.5%
False Negative (mild)	0	0	0	0.05	80.8%
False Negative (moderate)	0	0	0	0.15	69.5%
Structural Combined	0	0	0.15	0.15	54.5%
Combined Noise					
Mixed (mild)	5	5	0.05	0.05	67.5%
Mixed (moderate)	15	15	0.15	0.15	45.3%
Mixed (heavy)	25	25	0.25	0.25	34.6%
Mixed (severe)	30	30	0.30	0.30	29.2%
Asymmetric	7	5	0.25	0.31	38.2%

A.11 BELIEF-NAR ROBUSTNESS COMPARISON

Table 21 presents a direct comparison between standard NAR and our Belief-NAR method across matched noise configurations. This comparison uses test set accuracy (Top-1) on 16-node APSP graphs, with Belief-NAR trained on belief-valued inputs constructed via Monte Carlo sampling ($K = 16$).

Table 21: Comprehensive comparison of Standard NAR vs. Belief-NAR under various noise configurations on 16-node APSP. Test accuracy measures the fraction of node pairs with correctly predicted predecessors. Belief-NAR demonstrates substantial improvements, especially under structural noise (FP/FN), with gains exceeding 40% relative improvement in challenging scenarios.

Noise Type	Gauss. (%)	Outl. (%)	FP Rate	FN Rate	Standard NAR	Belief-NAR (Ours)	Improv.
Baseline							
Clean	0	0	0	0	87.8%	89.7%†	+2.2%
Isolated Weight Noise							
Gaussian Only	15	0	0	0	83.1%	**84.8%**	+2.0%
Outlier Only	0	15	0	0	75.7%	**85.0%**	+12.3%
Isolated Structural Noise							
Structural	0	0	0.15	0.15	54.5%	**87.1%**	+59.8%
Combined Noise							
Mixed (moderate)	15	15	0.15	0.15	45.3%	**86.2%**	+90.3%

†Clean performance estimated from validation; test accuracy on clean graphs is comparable between methods.

Key findings: (1) Under isolated Gaussian noise (15%), Belief-NAR achieves 84.8% test accuracy vs. standard NAR's 83.1%, a modest but consistent improvement. (2) Outlier noise shows larger gains: Belief-NAR achieves 85.0% vs. 75.7%, reflecting better handling of extreme value perturbations. (3) The most dramatic improvements occur under structural noise: with 15% FP and

15% FN rates, Belief-NAR achieves 87.1% accuracy compared to standard NAR's 54.5%—a 59.8% relative improvement. (4) Under combined moderate noise (all types at 15%), Belief-NAR nearly doubles baseline performance (86.2% vs. 45.3%), demonstrating that the belief-valued representation effectively mitigates compounding errors from multiple noise sources.

Training dynamics: Across all noise configurations, Belief-NAR training exhibited stable convergence with validation accuracy reaching 88–92% and validation loss (NLL) stabilizing around 0.49–0.57. The best validation checkpoints consistently achieved higher scores than final-step models, indicating effective early stopping criteria.

A.12 Calibration behavior under noise

Reliability analysis shows that deterministic NAR models become systematically overconfident under structural corruption, particularly under false-positive edge injection. In contrast, uncertainty-aware variants maintain substantially improved alignment between predicted confidence and empirical accuracy across confidence bins.

The largest calibration improvements occur for greedy algorithms (Dijkstra and Prim), where deterministic models frequently assign high confidence to incorrect edge selections after early structural errors. AECE reduces these overconfident failures by explicitly penalizing incorrect high-confidence predictions while preserving calibrated confidence on correct predictions.

Across all algorithms, uncertainty-aware variants consistently reduce ECE relative to deterministic baselines while preserving clean-regime performance.

B Architectural details

Our architecture follows the encode-process-decode paradigm with the following specifications:

Encoder.

- **Node encoder:** 2-layer MLP with hidden dimension 128, ReLU activation
- **Edge encoder:** 3-layer MLP with hidden dimension 128, processing $(\mu_{ij}, \sigma_{ij}, M_{ij}^{\text{prob}})$
- **Graph encoder:** Global mean pooling with 1-layer MLP

Processor.

- **Architecture:** Message Passing Neural Network (Gilmer et al., 2017)
- **Iterations:** $T = 32$ (matching Floyd-Warshall iterations for $n = 16$)
- **Hidden dimension:** 128
- **Message function:** 2-layer MLP
- **Update function:** GRU cell
- **Aggregation:** Max pooling

Decoder.

- **Predecessor distribution:** 2-layer MLP producing logits, followed by softmax
- **Output:** $P(\pi_{ij} = k|\cdot)$ for all $k \in \{1, \ldots, n\}$

Noise generation algorithm. Algorithm 1 describes our noise injection procedure for training and evaluation.

C THEORETICAL ANALYSIS

C.1 ERROR PROPAGATION IN FLOYD–WARSHALL

The Floyd-Warshall algorithm computes shortest paths via dynamic programming:

$$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) \tag{16}$$

where $d_{ij}^{(k)}$ is the shortest path from i to j using only intermediate vertices $\{1, \ldots, k\}$.

Noise amplification. Consider additive Gaussian noise on edge weights: $\tilde{w}_{ij} = w_{ij} + \epsilon_{ij}$ where $\epsilon_{ij} \sim \mathcal{N}(0, \sigma^2)$. For a path of length L, the total noise accumulates:

$$\text{Var}[\text{path cost}] = L \cdot \sigma^2 \tag{17}$$

As Floyd-Warshall progresses, paths grow longer, and noise variance compounds. For an n-node graph, paths can have length up to $n - 1$, yielding worst-case variance $(n - 1)\sigma^2$.

Decision boundary sensitivity. When two paths have similar costs, small perturbations can swap the optimal choice. If $d(p_1)$ and $d(p_2)$ are the costs of two competing paths with $|d(p_1) - d(p_2)| < \delta$, the probability of incorrect selection increases as $\delta \to 0$.

This analysis motivates our distributional approach: rather than committing to a single path, we model the posterior distribution over possible optimal paths, naturally handling cases where multiple paths are competitive.

C.2 PROPER SCORING RULES AND CALIBRATION

We train with the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{\text{NLL}} = -\sum_{i,j} \log P(\pi_{ij} = \pi_{ij}^* | \mathbf{h}_{ij}) \tag{18}$$

NLL is a *strictly proper scoring rule* (Gneiting & Raftery, 2007), meaning it is uniquely minimized when the predicted distribution matches the true conditional distribution:

$$\mathbb{E}[\mathcal{L}_{\text{NLL}}] \text{ is minimized iff } P(\pi_{ij}|\cdot) = P^*(\pi_{ij}|\cdot) \tag{19}$$

This property guarantees that optimizing NLL encourages calibrated uncertainty estimates, unlike accuracy-based losses that can produce overconfident predictions.

D IMPLEMENTATION DETAILS

D.1 NOISE GENERATION ALGORITHM

Algorithm 1 describes our noise injection procedure for training and evaluation.

D.2 HARDWARE AND SOFTWARE

All experiments were implemented in Python using JAX and the CLRS-30 framework (Veličković et al., 2022). Training and evaluation were performed on NVIDIA A100 GPUs with CUDA acceleration. The implementation uses the Haiku neural-network library together with Optax optimization utilities. GPU availability and backend selection are verified explicitly within the experimental driver to ensure reproducible execution across platforms. The implementation additionally logs GPU memory availability, device type, and runtime backend during initialization.

Experiments were executed using mixed deterministic and uncertainty-aware training modes implemented through a shared-stream evaluation pipeline. All random seeds were fixed prior to dataset generation, model initialization, and Monte Carlo graph sampling to ensure reproducibility across repeated runs.

Algorithm 1 Noise Injection for Belief Construction

Require: Clean graph $G = (V, E, W)$, noise parameters $(\sigma_w, p_{\text{drop}}, p_{\text{add}}, p_{\text{outlier}})$, number of samples K
Ensure: Belief statistics $(\mu, \sigma, M^{\text{prob}})$

Initialize $\mathbf{W}_{\text{samples}} \in \mathbb{R}^{K \times n \times n}$
Initialize $\mathbf{M}_{\text{samples}} \in \{0, 1\}^{K \times n \times n}$
for $k = 1, \ldots, K$ **do**
  $\mathbf{W}^{(k)} \leftarrow \mathbf{W}$ {Copy clean weights}
  $\mathbf{M}^{(k)} \leftarrow \mathbf{M}$ {Copy clean adjacency}
  for each edge $(i, j) \in E$ **do**
    $\epsilon \sim \mathcal{N}(0, \sigma_w^2)$
    if Uniform$(0, 1) < p_{\text{outlier}}$ **then**
      $W_{ij}^{(k)} \leftarrow W_{ij} + \epsilon \cdot 5$ {Outlier: 5× noise}
    else
      $W_{ij}^{(k)} \leftarrow W_{ij} + \epsilon$ {Standard noise}
    end if
    if Uniform$(0, 1) < p_{\text{drop}}$ **then**
      $M_{ij}^{(k)} \leftarrow 0$ {Drop edge}
    end if
  end for
  for each non-edge $(i, j) \notin E$ **do**
    if Uniform$(0, 1) < p_{\text{add}}$ **then**
      $M_{ij}^{(k)} \leftarrow 1$ {Add spurious edge}
      $W_{ij}^{(k)} \sim \text{Uniform}(w_{\min}, w_{\max})$
    end if
  end for
end for
$\mu_{ij} \leftarrow \frac{1}{K} \sum_k W_{ij}^{(k)}$ for all (i, j)
$\sigma_{ij} \leftarrow \sqrt{\frac{1}{K} \sum_k (W_{ij}^{(k)} - \mu_{ij})^2}$ for all (i, j)
$M_{ij}^{\text{prob}} \leftarrow \frac{1}{K} \sum_k M_{ij}^{(k)}$ for all (i, j)
return $(\mu, \sigma, M^{\text{prob}})$

D.3 Datasets and graph generation

We build upon the CLRS-30 benchmark (Veličković et al., 2022) using Erdős–Rényi random graphs with edge probability $p = 0.5$ and uniform edge weights sampled from $[0.1, 1.0]$. All experiments use graphs with $n = 16$ nodes. For each algorithm, we generate 10k/1k/1k train/validation/test samples.

The benchmark evaluates four representative graph algorithms:

- Floyd–Warshall (all-pairs shortest paths),
- Bellman–Ford (single-source shortest paths),
- Dijkstra (greedy shortest paths),
- Prim (minimum spanning tree).

All supervision targets are computed from the underlying clean graph G^* before corruption is applied.

D.4 Noise protocol

We evaluate robustness under four independent noise processes:

- Gaussian edge-weight perturbation (σ_g),
- Outlier edge corruption (r_o),
- False-positive edge injection (r_{fp}),
- False-negative edge dropout (r_{fn}).

Each parameter is swept over $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ in isolation and in structured combinations, yielding 36 total corruption configurations per algorithm in addition to a clean baseline. During training we use $K = 16$ Monte Carlo graph draws per observation, while evaluation uses $K = 32$ samples for more stable belief estimation.

Belief-valued graph statistics are computed as:

$$\mu_{ij} = \frac{1}{K}\sum_{k=1}^{K} \tilde{w}_{ij}^{(k)}, \qquad \sigma_{ij}^2 = \frac{1}{K}\sum_{k=1}^{K}\left(\tilde{w}_{ij}^{(k)} - \mu_{ij}\right)^2, \qquad p_{ij} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{1}\left[(i,j) \in \tilde{E}^{(k)}\right]. \quad (20)$$

These statistics explicitly separate parametric uncertainty (σ_{ij}^2) from structural uncertainty (p_{ij}).

D.5 Model variants

We evaluate one deterministic baseline and seven uncertainty-aware variants:

- Deterministic CLRS-30 baseline,
- U-Base,
- U-AECE,
- U-NTE,
- U-NTE+AECE,
- U-UPH,
- U-Full,
- Additional ablation combinations.

All models share the same MPNN processor architecture following Gilmer et al. (2017). The uncertainty-aware variants differ only in the inclusion of the proposed components:

- Noise-Type-Aware Encoder (NTE),
- Uncertainty-Propagating Hints (UPH),
- Adaptive ECE Calibration Loss (AECE).

D.6 Optimization and hyperparameters

All models are trained using Adam optimization with gradient clipping. Hyperparameters are shared across algorithms to ensure fair comparison.

The principal hyperparameters are:

Table 22: Training hyperparameters.

Hyperparameter	Value
Graph size n	16
Training samples	10,000
Validation samples	1,000
Test samples	1,000
Monte Carlo samples (train)	16
Monte Carlo samples (eval)	32
Optimizer	Adam
Hint loss weight λ_{hint}	1.0
AECE weight λ_{aece}	0.1
Dropout	0.0
Processor type	MPNN
Message-passing steps	16
Batching mode	Shared-stream

The complete objective is:

$$\mathcal{L} = \mathcal{L}_{\text{NLL}} + \lambda_{\text{hint}}\mathcal{L}_{\text{hint}} + \lambda_{\text{aece}}\mathcal{L}_{\text{AECE}}. \quad (21)$$

D.7 Training and evaluation protocol

Each model is trained independently using identical dataset splits and noise schedules. Validation performance determines checkpoint selection. Evaluation uses pre-generated cached test batches to guarantee identical inputs across all model variants.

Performance is measured using:

- Accuracy,
- Expected Calibration Error (ECE),
- Relative Robustness Score (RRS),
- Floor@0.75,
- Worst-Case Accuracy (WCAcc).

For all reported metrics, results are averaged across multiple runs with independent random seeds and independently sampled corruption instances.

D.8 Runtime and computational overhead

The primary computational overhead arises from Monte Carlo belief construction and uncertainty-aware edge encoding. Belief construction adds preprocessing complexity of:

$$O(Kn^2), \tag{22}$$

where K is the number of Monte Carlo samples and n is the graph size.

The NTE encoder approximately doubles encoder-side parameters due to the dual-path architecture. Despite this overhead, inference complexity remains identical to the deterministic baseline once belief statistics have been computed offline.

In practice, uncertainty-aware training increases wall-clock runtime by approximately 15% relative to deterministic CLRS training.

D.9 Reproducibility

The full implementation includes:

- dataset generation scripts,
- deterministic and uncertainty-aware training pipelines,
- evaluation scripts,
- checkpoint management utilities,
- calibration metrics,
- robustness analysis code.

All experiments can be reproduced from the released codebase using the provided configuration files and fixed seeds. The implementation builds upon the publicly available CLRS-30 benchmark and associated baseline infrastructure released for academic research purposes. All third-party libraries used in the implementation, including JAX, Haiku, and Optax, are employed under their respective open-source licenses. Full attribution to the original benchmark, framework authors, and referenced implementations is provided throughout the paper bibliography and accompanying code release.

E Broader Impact

This work studies uncertainty-aware neural algorithmic reasoning under corrupted graph observations. The proposed framework may positively impact applications where robustness and calibrated confidence are critical.

Potential positive impacts. Reliable uncertainty estimation can improve safety and interpretability in graph-based decision systems, including:

- network routing,
- infrastructure planning,
- robotic navigation,

- scientific graph analysis,
- uncertain sensor fusion.

By explicitly representing uncertainty rather than forcing deterministic predictions, the model can avoid overconfident failures in ambiguous settings.

Risks and misuse. Despite improved calibration, uncertainty-aware predictions remain approximate. Overreliance on model confidence could still lead to incorrect decisions under severe corruption or distribution shift. Potential misuse includes automated decision systems that rely on neural algorithmic outputs without human oversight.

Limitations for safety-critical deployment. The current framework assumes access to multiple noisy observations of the same graph. Real-world systems may instead provide only a single observation, requiring learned uncertainty estimation rather than direct Monte Carlo aggregation.

Furthermore, the present evaluation focuses on relatively small synthetic graphs. Additional validation is required before deployment in large-scale or safety-critical environments.

Environmental considerations. Monte Carlo sampling increases training-time computation relative to deterministic baselines. Although the runtime increase is moderate (approximately 15%), large-scale uncertainty-aware training may still increase energy consumption compared to conventional NAR pipelines.