Primal-Dual Neural Algorithmic Reasoning

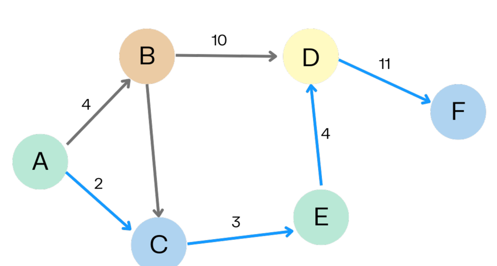
Yu He[1], Ellen Vitercik[1]   [1]Stanford University

TL;DR: We propose a general **neural algorithmic reasoning (NAR)** framework for **NP-hard** problems, using the primal-dual approximation algorithm.

Algorithmic reasoning

Neural algorithmic reasoning (NAR) teaches neural networks to simulate algorithmic execution.

Algorithm



+

Neural network

Fixed input formats   Rich domain-specific features

Most works on NAR focus on polynomial-time-solvable problems, but many real-world problems are NP-hard!

Primal-dual approximation

Duality: Each optimization problem can be viewed from two perspectives: the primal and the dual.

Minimum Hitting Set

Given a set of subsets $\mathcal{T}$, each containing some elements $e \in E$, a hitting set A covers at least one element e from each subset T. The goal is to minimize the total (non-negative) weights of elements in A.

Let x_e (primal) represent whether to include element e in A, and y_T (dual) represent the weight assigned to subset T.

Algorithm 1 General primal-dual approximation algorithm

Input: Ground set E with weights $\boldsymbol{w}$, family of subsets $\mathcal{T} \subseteq 2^E$

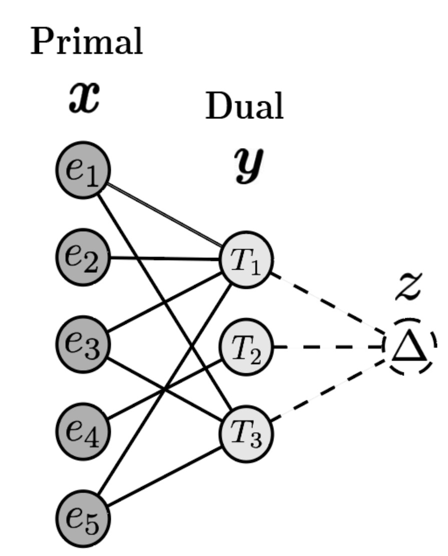
1: $A \leftarrow \emptyset$; for all $e \in E, r_e \leftarrow w_e$
2: **while** $\exists T : A \cap T = \emptyset$ **do**
3: $\quad \mathcal{V} \leftarrow \{T : A \cap T = \emptyset\}$
4: $\quad$ **repeat**
5: $\quad\quad$ **for** $T \in \mathcal{V}$ **do** $\delta_T \leftarrow \min_{e \in T} \left\{ \frac{r_e}{|\{T' : e \in T'\}|} \right\}$ — Increase dual variables
6: $\quad\quad$ **for** $e \in E \setminus A$ **do** $r_e \leftarrow r_e - \sum_{T: e \in T} \delta_T$
7: $\quad$ **until** $\exists e \notin A : r_e = 0$
8: $\quad A \leftarrow A \cup \{e : r_e = 0\}$ — Update primal variables

Output: A

When a constraint is met for a primal variable (L7), include it into the solution (L8). Repeat until a hitting set is found (L2).

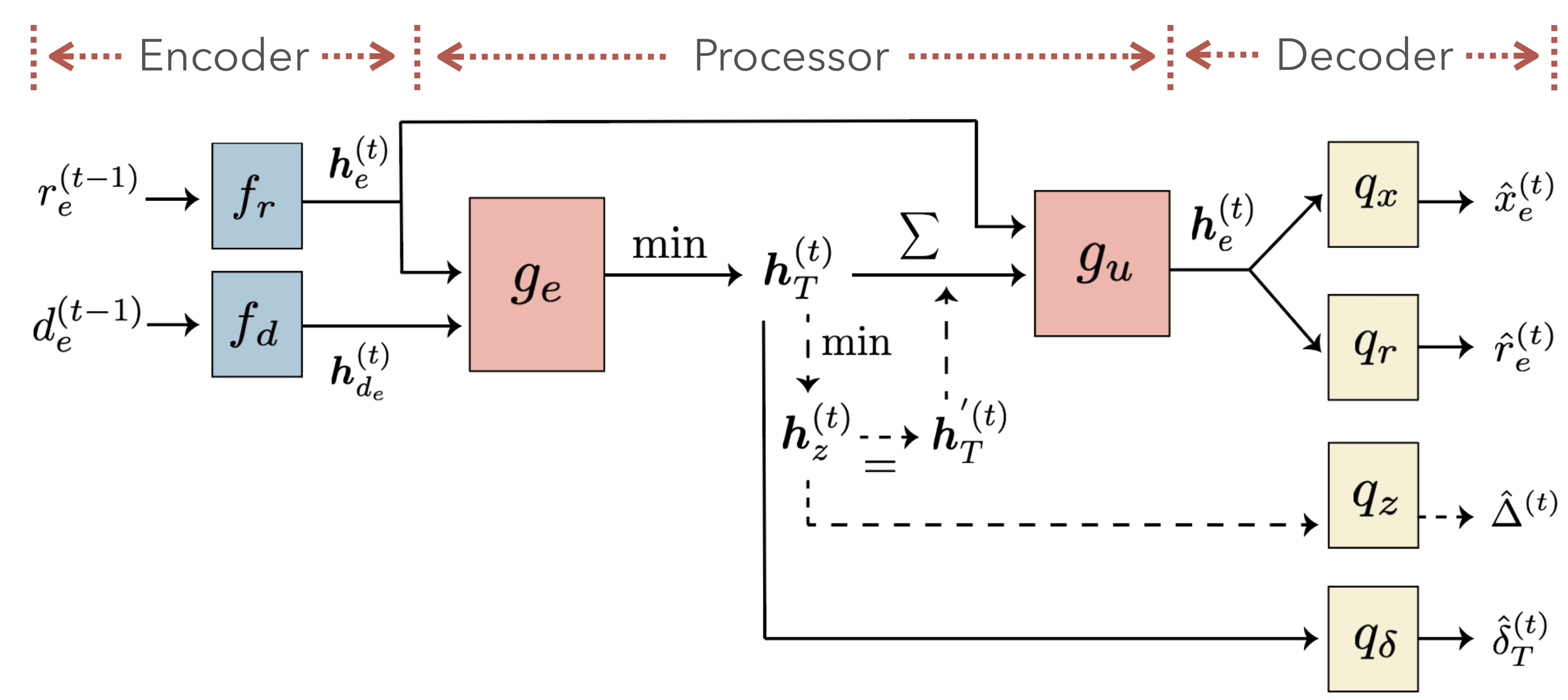
PDNAR: A general NAR framework

Bipartite graph representation



- Construction: Connect e and T if $e \in T$ (the subset contains the element).
- Removal: When e is included in the solution, remove all its connected Ts.
- Uniform increase rule (optional): a virtual node z that connects to all duals.

Architectural design



- Encoder: Maps scalar inputs (e.g. r_e) to feature vectors.
- Processor: Simulate algorithmic steps with message-passing:
  - L5: $h_T^{(t)} = \min_{e \in \mathcal{N}(T)} g_e(h_e^{(t)}, h_{d_e}^{(t)})$ – aggregate
  - L6: $h_e^{(t)} = g_u\left(h_e^{(t)}, \sum_{T \in \mathcal{N}(e)} h_T^{(t)}\right)$ – update
- Decoder: Produce outputs (e.g. x_e) from feature vectors.

The virtual node allows simultaneous updates of all dual variables, extending PDNAR to a broader range of algorithms (e.g. greedy).

Training signals

- Intermediate algorithmic steps synthetically generated by running the primal-dual approximation algorithm.
- Optimal solutions efficiently obtained by solving small problems using integer programming solvers.

We test generalization to larger instances through theoretically justified algorithmic alignment.

Experiments

NP-hard algorithmic problems

- Minimum Vertex Cover (MVC)
- Minimum Set Cover (MSC)
- Minimum Hitting Set (MHS)

A general formulation of a wide range of problems.

Instances are generated using Barabási-Albert (bipartite) graphs.

A general NAR framework for NP-hard problems

Table 1. Model-to-algorithm weight ratio (smaller is better) trained on 16-node graphs and tested on larger graphs.

		16 (1x)	128 (8x)	512 (32x)	1024 (64x)
MVC	GAT	0.962	1.071	1.114	1.125
	NAR	0.998	1.002	1.013	1.018
	No algo	1.142	1.099	1.099	1.095
	No optm	0.995	0.998	0.998	0.994
	PDNAR (max)	0.968	1.005	1.010	1.007
	PDNAR	**0.943**	**0.958**	**0.958**	**0.957**
MSC	No algo	1.028	1.017	1.008	1.006
	No optm	1.008	0.992	0.973	0.975
	PDNAR	**0.979**	**0.915**	**0.915**	**0.913**
MHS	No algo	1.047	1.036	1.122	1.256
	No optm	1.002	0.999	1.015	1.053
	PDNAR	**0.989**	**0.965**	**0.996**	**1.027**

- Algorithmic reasoning enhances generalization.
- Optimal supervision enables the model to surpass the performance of the underlying algorithm.

Robust to size and OOD generalization

Table 2. Model-to-algorithm weight ratio (smaller is better) trained on 16-node Barabási-Albert (bipartite) graphs, and tested on OOD graph families. Note b is the preferential attachment parameter (trained on b=5).

		16 (1x)	128 (8x)	512 (32x)	1024 (64x)
MVC	E-R	0.955	0.950	0.989	0.993
	Star	0.966	0.982	0.992	0.998
	Lobster	0.971	0.960	0.966	0.966
	3-Con	0.974	0.957	0.962	0.961
MSC	b=3	0.943	0.918	0.929	0.922
	b=8	0.969	0.940	0.941	0.943
MHS	b=3	0.988	0.982	1.008	1.005
	b=8	0.979	0.960	1.008	1.014

We also showcase **two applications of PDNAR**:

- Algorithmically-informed embeddings to improve GNN performance in real-world datasets.
- Warm starts to speed up commercial solvers.

More details in the full paper ⟶