Verifying message-passing neural networks via topology-based bounds tightening

Christopher Hojny[*,1], Shiqiang Zhang[*,2], Juan S. Campos[2], Ruth Misener[2]

[*] These authors contributed equally.
[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Imperial College London, London, UK

Funding: EPSRC EP/W003317/1 & EP/T001577/1

Adversarial attack v.s. Certifiable robustness

Machine learning models are vulnerable: small input changes could lead to wrong predictions.

Denote f as a model, assume $\mathcal{P}(X^*)$ is the admissible perturbations on input X^*.

Adversarial attack 🗡️	Certifiable robustness 🛡️
$\exists X \in \mathcal{P}(X^*)$, s.t., $f(X) \neq f(X^*)$	$f(X) = f(X^*),\ \forall X \in \mathcal{P}(X^*)$

Besides input features, the graph structure involved in graph neural networks (GNNs) provides more options to attack (🔨 🏹 🪚), while makes it harder to be verified (certified robustness).

Problem definition

Given a trained GNN f for graph/node classification task, where the predicted label corresponds to the maximal logit. Given an input (X^*, A^*) consisting of features X^* and adjacency matrix A^*, denote its predictive label as c^*. The worst case margin between predictive label c^* and attack label c under perturbations $\mathcal{P}(\cdot)$ is:

$$\begin{aligned} m(c^*, c) := \min_{(X,A)} \ & f_{c^*}(X, A) - f_c(X, A) \\ s.t.\ & X \in \mathcal{P}(X^*),\ A \in \mathcal{P}(A^*). \end{aligned} \tag{1}$$

A positive $m(c^*, c)$ means that the logit of class c^* is always larger than class c.

Let $\mathcal{C}$ be the set of all classes. If $m(c^*, c) > 0, \forall c \in \mathcal{C} \backslash \{c^*\}$, then any admissible perturbation can not change the predictive label, i.e., this GNN is robust at (X^*, A^*).

Admissible perturbations

Perturbations on features, i.e., $\mathcal{P}(X^*)$, are usually defined as a l_p norm ball around X^*. The choice of norm is quite flexible for attack since one feasible attack is sufficient. For verification, l_∞ norm is most commonly used since it defines bounds for each feature separately.

Remark: If only feature perturbations are allowed, then verifying a GNN is equivalent to verifying a NN since the connections between layers are fixed.

New challenges for GNN verification:

- Perturbations on graph structure, e.g., add edges/remove edges/inject nodes, directly change the connections between layers.
- Perturbations on one node indirectly attack other nodes via message passing or graph convolution.

Verification of message passing neural networks (MPNNs)

Motivation: classic and general GNN framework, but few certificates.

Tool: a recently developed mixed-integer programming (MIP) formulation for MPNNs.

Definition: consider a MPNN with l-th layer defined as:

$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u \in V} A_{u,v} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right), \ \forall v \in V \quad (2)$$

where $V = \{0, 1, \ldots, N-1\}$ is the node set, N is the number of nodes, $A_{u,v} \in \{0, 1\}$ denotes the existence of edge $u \to v$.

Perturbations:

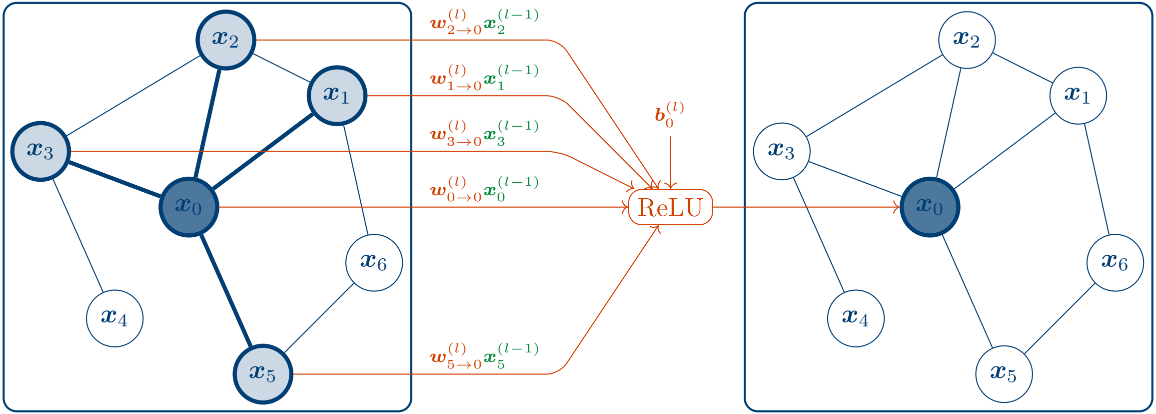
- Graph classification: remove/add edges with global/local budgets.
- Node classification: remove edges with global/local budgets.

Message passing with fixed graph structure

$(l-1)^{th}$ layer

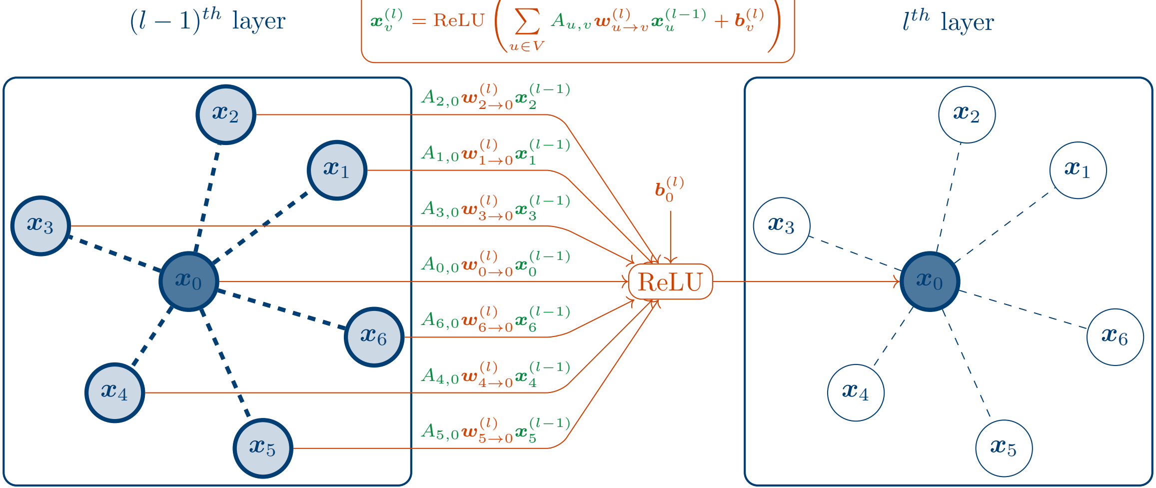
$$\boldsymbol{x}_v^{(l)} = \text{ReLU}\left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$

l^{th} layer



Message passing with unknown graph structure

$$\boldsymbol{x}_v^{(l)} = \mathrm{ReLU}\left(\sum_{u\in V} A_{u,v}\boldsymbol{w}_{u\to v}^{(l)}\boldsymbol{x}_u^{(l-1)} + \boldsymbol{b}_v^{(l)}\right)$$



MIP encoding of MPNNs

$$\boldsymbol{x}_v^{(l)} = \max\{\underbrace{\bar{\boldsymbol{x}}_v^{(l)}}, \mathbf{0}\} \longleftrightarrow \begin{cases} x_{v,f}^{(l)} \geq 0 \\ x_{v,f}^{(l)} \geq \bar{x}_{v,f}^{(l)} \\ x_{v,f}^{(l)} \leq \bar{x}_{v,f}^{(l)} - lb(\bar{x}_{v,f}^{(l)}) \cdot (1 - \sigma_{v,f}^{(l)}) \\ x_{v,f}^{(l)} \leq ub(\bar{x}_{v,f}^{(l)}) \cdot \sigma_{v,f}^{(l)} \end{cases}$$

$$\downarrow$$

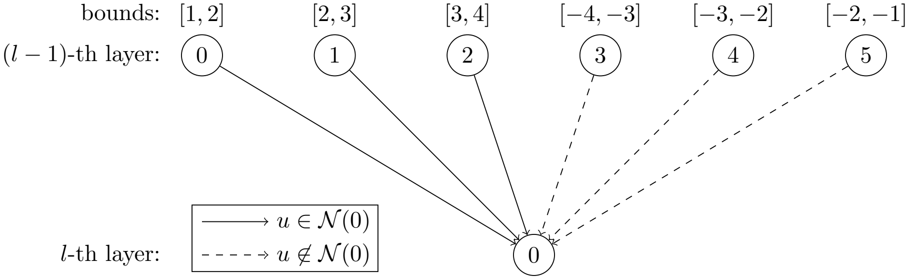
$$\bar{\boldsymbol{x}}_v^{(l)} = \sum_{u \in V} \boldsymbol{w}_{u \to v}^{(l)} \boldsymbol{x}_{u \to v}^{(l-1)} + \boldsymbol{b}_v^{(l)}$$

$$\uparrow$$

$$\overbrace{\boldsymbol{x}_{u \to v}^{(l-1)}} = A_{u,v} \boldsymbol{x}_u^{(l-1)} \leftrightarrow \begin{cases} x_{u \to v,f}^{(l-1)} \geq lb(x_{u,f}^{(l-1)}) \cdot A_{u,v} \\ x_{u \to v,f}^{(l-1)} \leq ub(x_{u,f}^{(l-1)}) \cdot A_{u,v} \\ x_{u \to v,f}^{(l-1)} \leq x_{u,f}^{(l-1)} - lb(x_{u,f}^{(l-1)}) \cdot (1 - A_{u,v}) \\ x_{u \to v,f}^{(l-1)} \geq x_{u,f}^{(l-1)} - ub(x_{u,f}^{(l-1)}) \cdot (1 - A_{u,v}) \end{cases}$$

Basic bounds tightening (*basic*)

Assume that there are $N = 6$ nodes with only one input and output feature. For simplicity, assume all weights equal to 1 and all biases equal to 0.

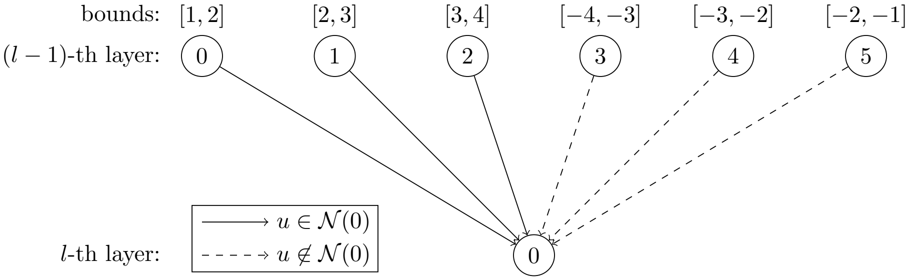


To get the bounds for node 0 in l-th layer, *basic* considers all possibilities of input nodes:

- $lb = \min(0, 1) + \min(0, 2) + \min(0, 3) + \min(0, -4) + \min(0, -3) + \min(0, -2) = -9.$
- $ub = \max(0, 2) + \max(0, 3) + \max(0, 4) + \max(0, -3) + \max(0, -2) + \max(0, -1) = 9.$

Static bounds tightening (*sbt*)

Given that the budget, i.e., the maximal number of modified edges of node 0, is 3. Denote the set of input nodes as $\mathcal{N}'(0)$, then we need to make sure that $|\mathcal{N}'(0)\Delta\mathcal{N}(0)| \leq 3$.

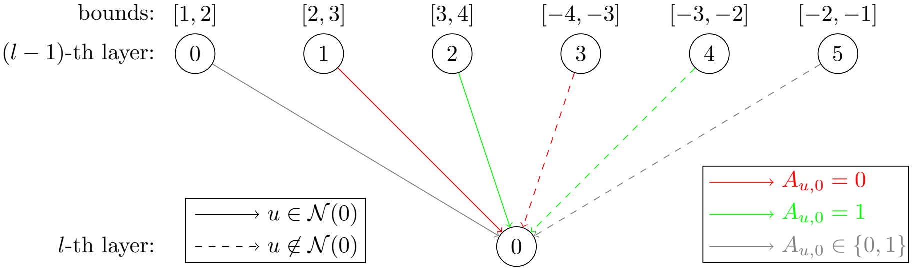


Comparing all possible options gives the *sbt* bounds:

- $lb = 1 + 2 - 4 - 3 = -4$: $\mathcal{N}'(0) = \{0, 1, 3, 4\}$, i.e., remove node 2 + add node 3 and 4.
- $ub = 2 + 3 + 4 = 9$: $\mathcal{N}'(0) = \mathcal{N}(0)$.

Aggressive bounds tightening (*abt*)

Assume that 4 decisions have been made in current branch-and-bound (B&B) tree node, which are $A_{1,0} = 0$, $A_{2,0} = 1$, $A_{3,0} = 0$, $A_{4,0} = 1$. Then we only have 1 budget left.

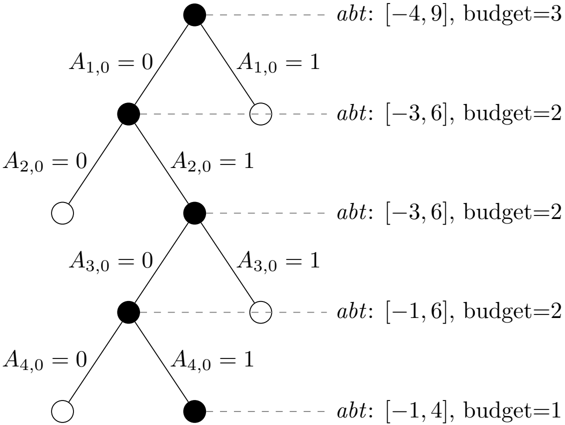


We can (i) change nothing, or (ii) remove node 0, or (iii) add node 5. The *abt* bounds are:

- $lb = 1 + 3 - 3 - 2 = -1$: add node 5.
- $ub = 2 + 4 - 2 = 4$: change nothing.

abt extends *sbt* to each B&B tree node

abt can be interpreted as applying *sbt* to a modified graph with reduced budgets at each B&B tree node. At root node, *abt* = *sbt*.



Numerical results

benchmark	method	all instances			robust instances		
		#	avg-time(s)	# solved	#	avg-time(s)	# solved
ENZYMES	SCIPbasic	5915	605.97	5579	3549	278.58	3444
	SCIPsbt	5915	**230.59**	**5831**	3549	**82.89**	**3528**
	SCIPabt	5915	246.02	5817	3549	88.95	3522
MUTAG	SCIPbasic	1589	679.86	1575	44	798.47	40
	SCIPsbt	1589	**196.07**	**1589**	44	336.41	**44**
	SCIPabt	1589	207.50	**1589**	44	**238.10**	**44**

Conclusion

Based on the results of our SCIP implementation, we have the following observations:

- For moderate robust instances, *basic* $<$ *sbt* $\approx$ *abt*.
- For hard robust instances, *basic* $<$ *sbt* $<$ *abt*.
- For non-robust instances, *basic* $<$ *abt* $<$ *sbt*.

For a non-robust instance, the target is not verification but finding an attack. In such cases, tighter bounds derived from more cutting planes could result in slower solving times.

arXiv

GitHub

c.hojny@tue.nl

s.zhang21@imperial.ac.uk