FM 2026 TOKYO
The 27th International Symposium on Formal Methods

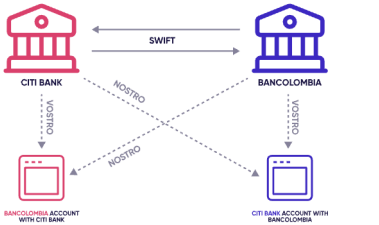
Exact Verification of Graph Neural Networks with Incremental Constraint Solving

Minghao Liu, Chia-Hsuan Lu, Marta Kwiatkowska
University of Oxford
AIPV 2026, May 19, 2026

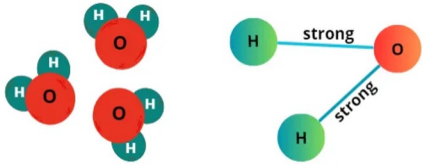
UK Research and Innovation
elsa European Lighthouse on Secure and Safe AI

Graph Neural Networks (GNNs)

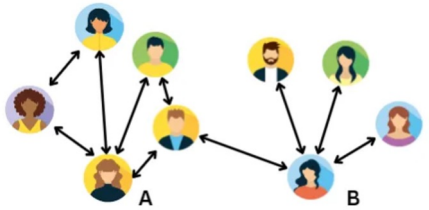
- GNNs are a family of neural network architectures over **graph data**.
- GNNs have been deployed in real-world high-stakes applications:



Financial networks



Chemical compounds



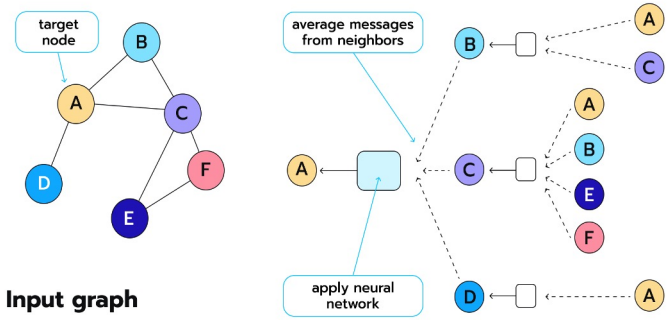
Social networks

- GNNs have been shown to be vulnerable to adversarial attacks.

We use formal verification to guarantee the robustness of GNNs.

Graph Neural Networks (GNNs)

- **Input:** attributed directed graph $G = \langle V, E, X \rangle$
  - V: a node set;
  - $E \subseteq V \times V$: an edge set;
  - $X \in \mathbb{R}^{|V| \times D}$: real attribute vectors for the nodes.



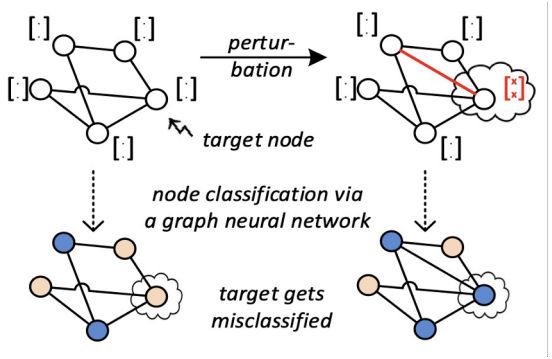
- **Message passing mechanism**
  - GNNs learn the embedding of nodes through message passing by layers;
  - $\mathbf{h}_v^{(k)}$: real-valued embedding vector of node $v \in V$ for the k-th layer ($\mathbf{h}_v^{(0)} = X_v$);
  - We consider *GraphSAGE* [1] (i.e., Message-Passing Neural Networks) in this paper:

$$\mathbf{h}_v^{(k)} = \sigma\left(\mathbf{W}_1^{(k)} \cdot \mathbf{h}_v^{(k-1)} + \mathbf{W}_2^{(k)} \cdot \mathbf{aggr}\left(\left\{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}(v)\right\}\right) + \mathbf{b}_1^{(k)}\right)$$

  - There are three common aggregators: $\mathbf{aggr} \in \{\mathbf{sum}, \mathbf{max}, \mathbf{mean}\}$;
  - We use ReLU activation $\sigma(x) = \max(x, 0)$.

Adversarial Robustness of GNNs

- Admissible graph perturbation space
  - **Attribute perturbation:** change the value of node attributes within a range.
    - For every $v \in V$ and $1 \leq i \leq d_0$, $\epsilon_{v,i}^{l} \leq \tilde{\mathbf{x}}_v[i] \leq \epsilon_{v,i}^{u}$;
  - **Structural perturbation:** insert/remove edges in a fragile edge set $F \subseteq V \times V$.
    - Only perturb fragile edges: $F \backslash E \subseteq \tilde{E} \subseteq E \cup F$;
    - Global budget: $E \backslash \tilde{E} + \tilde{E} \backslash E \leq \Delta$;
    - Local budget: For every $v \in V$,
      $$\left|\mathcal{N}(v) \backslash \widetilde{\mathcal{N}}(v)\right| + \left|\widetilde{\mathcal{N}}(v) \backslash \mathcal{N}(v)\right| \leq \delta_v;$$
- Structural perturbations make GNNs different:
  - Lead to hard discrete optimisation problems in verification.



Shown in red: Demonstration of attribute and structural perturbations.

Adversarial Robustness of GNNs

- Definition of GNN robustness

> Given a node-classification GNN f, an attributed directed graph G with admissible perturbation space $\mathcal{Q}(G)$, a target node $t \in V$, and $f(G,t) = \hat{c}_t$, we say that f is **adversarially robust** for t with class $\hat{c}_t$ if and only if, for every perturbed graph $\tilde{G} \in \mathcal{Q}(G)$, it holds that $f(\tilde{G},t) = \hat{c}_t$.

- We consider both node- and graph-classification cases.
  - Definition for graph-classification GNNs are similar.
  - For graph classification, the predicted class is obtained via

$$\hat{c}_G = \underset{c \in C}{\arg\max} \left(\mathrm{Softmax} \left(\mathbf{W}_3 \cdot \sum_{v \in V} \mathbf{h}_v^{(K)} + \mathbf{b}_2 \right) [c] \right).$$

Related Work

- Most work is **approximate verification** approach (on GCNs).
  - Attribute perturbation only [2];
  - Structural perturbation also allowed [3,4,5].
- We are most similar to two **exact verification** approaches:
  - *SCIP-MPNN* [6]
    - Verification via Mixed-Integer Programming (MIP) formulation;
    - Propose bound tightening strategies to improve efficiency;
    - Only consider sum-aggregated GNNs and edge deletions.
  - *RobLight* [7]
    - Branch-and-bound approach with bound propagation;
    - Only support structural perturbations and hard to generate certificates.

Contributions

- We introduce the first exact verification method for GNNs with two common aggregators, **max** and **mean**.
- We design specialised **tightened bound propagation** strategies for the two aggregators to reduce computational cost.
- We propose an **iterative algorithm** utilising incremental constraint solving to speed up performance while maintaining exact verification capability.
- We implement *GNNev*, an **open-source exact verifier** for GNNs that supports new aggregators, attribute/structural perturbations, and edge additions/deletions.
- The performance is evaluated through extensive experiments on real-world graph datasets (e.g., fraud detection, biochemistry).

MIP Encoding

- We show the encoding for K-layer node-classification GNNs.
- Constraints on input perturbation
  - Input (attribute) variables: for each $v \in \mathcal{N}_K(t)$, set variables $attr_{v,1}, attr_{v,2}, \ldots, attr_{v,d_0}$;
  - $\mathcal{N}_k(t)$ is the set of k-hop incoming neighbours of target node t;
  - Constraints for attribute perturbation

$$\epsilon^l_{v,i} \leq attr_{v,i} \leq \epsilon^u_{v,i}.$$

  - Constraints for structural perturbation (Boolean variable $pe_{u,v}$: if $(u,v) \in F$ is perturbed)

$$\sum_{(u,v)\in F} pe_{u,v} \leq \Delta,$$

For each $v \in \mathcal{N}_{K-1}(t)$, $\sum_{(u,v)\in F} pe_{u,v} \leq \delta_v.$

MIP Encoding

- Constraints on GNN architecture
  - Embedding variables: for the k-th layer, for each $v \in \mathcal{N}_{K-k}(t)$, set variables $h_{v,1}^{(k)}, h_{v,2}^{(k)}, \ldots, h_{v,d_k}^{(k)}$;
  - Let $h_{v,i}^{(0)} = attr_{v,i}$;
  - Constraints for message generation
    - Variables for the message vector to node v: $msg_{v,1}^{(k)}, msg_{v,2}^{(k)}, \ldots, msg_{v,d_{k-1}}^{(k)}$;
    - Set auxiliary variables $a_{v,i,u}^{(k)}$: the contribution of node u to $msg_{v,i}^{(k)}$ through edge (u,v);
    - For each $(u,v) \in E \backslash F$: $a_{v,i,u}^{(k)} = h_{v,i}^{(k-1)}$
    - For each $(u,v) \in E \cap F$: $pe_{u,v} \rightarrow a_{v,i,u}^{(k)} = 0 \quad \text{and} \quad \neg pe_{u,v} \rightarrow a_{v,i,u}^{(k)} = h_{v,i}^{(k-1)}$
    - For each $(u,v) \in F \backslash E$: $pe_{u,v} \rightarrow a_{v,i,u}^{(k)} = h_{v,i}^{(k-1)} \quad \text{and} \quad \neg pe_{u,v} \rightarrow a_{v,i,u}^{(k)} = 0.$

MIP Encoding

- Constraints on GNN architecture (cont.)
  - Constraints for message generation
    - For sum aggregation: $$msg_{v,i}^{(k)} = \sum_{(u,v)\in F\cup E} a_{v,i,u}^{(k)},$$
    - For max aggregation: $$msg_{v,i}^{(k)} = \max_{(u,v)\in F\cup E} a_{v,i,u}^{(k)}.$$ (big-M encoding for max)
    - For mean aggregation: $$deg_v = \sum_{(u,v)\in F\cap E} (1 - pe_{u,v}) + \sum_{(u,v)\in F\setminus E} pe_{u,v} + |E\setminus F|,$$
      $$deg_v \cdot msg_{v,i}^{(k)} = \sum_{(u,v)\in F\cup E} a_{v,i,u}^{(k)},$$
      $$-M \cdot deg_v \le msg_{v,i}^{(k)} \le M \cdot deg_v,$$

MIP Encoding

- Constraints on GNN architecture (cont.)
  - Constraints for embedding updating
    - Pre-ReLU: $y_{v,i}^{(k)} = \sum_{j \in [1, d_{k-1}]} \mathbf{W}_1^{(k)}[i,j] \cdot h_{v,j}^{(k-1)} + \mathbf{W}_2^{(k)}[i,j] \cdot msg_{v,j}^{(k)} + \mathbf{b}_1^{(k)}$
    - Post-ReLU: $h_{v,i}^{(k)} = \max\left(y_{v,i}^{(k)}, 0\right)$
- Constraints on verification objective
  - C is the set of classes;
  - Set a constraint to break the robustness: $\left(\max_{c \in C \setminus \{\hat{c}_t\}} y_{t,c}^{(K)}\right) \geq y_{t,\hat{c}_t}^{(K)}$
  - This implies that there exists a class $c \in C$ with $c \neq \hat{c}_t$ has higher score than $\hat{c}_t$.
  - Therefore, robustness is verified iff. the MIP instance is **unsatisfiable**.

Bound Tightening

- The only special part for GNNs is the aggregators (max and mean).

$$\mathbf{h}_v^{(k)} = \sigma\left(\mathbf{W}_1^{(k)} \cdot \mathbf{h}_v^{(k-1)} + \mathbf{W}_2^{(k)} \cdot \mathbf{aggr}\left(\left\{\!\!\left\{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}(v)\right\}\!\!\right\}\right) + \mathbf{b}_1^{(k)}\right)$$

  - We simplify the problem as: $z := \mathbf{aggr}(X_1 \cup X_2' \cup X_3')$
  - X_1: variables from non-fragile edges $(E \backslash F)$;
  - X_2: variables from fragile edges $(E \cap F)$, and $X_2' \subseteq X_2$;
  - X_3: variables from fragile non-edges $(F \backslash E)$, and $X_3' \subseteq X_3$;
  - There is a constant integer s, such that $|X_2 \backslash X_2'| + |X_3'| \leq s$.

- We design efficient methods to **propagate bounds** for max and mean aggregations and prove that the computed bounds are tight (reachable).

Bound Tightening

- We maintain the k-th largest/smallest bounds:
  - $hi(X, k)$: the k-th **largest upper bound** among set X (let $hi(X, k) = -\infty$ if $k > |X|$);
  - $lo(X, k)$: the k-th **smallest lower bound** among set X (let $lo(X, k) = \infty$ if $k \leq 0$);
  - We compute $hi(X, k)$ and $lo(X, k)$ in $O(|X| \log k)$ time by maintaining max- and min-heaps.
- For max aggregation
  - Upper bound: $$\overline{z} = \begin{cases} \max(0, hi(X_2, 1), hi(X_3, 1)), & \text{if } X_1 = \emptyset, s \geq |X_2|, \\ \max(hi(X_1, 1), hi(X_2, 1), hi(X_3, 1)), & \text{otherwise.} \end{cases}$$
  - Lower bound: $$\underline{z} = \begin{cases} \min(0, lo(X_2, 1), lo(X_3, 1)), & \text{if } X_1 = \emptyset, s > |X_2|, \\ \min(0, lo(X_2, 1)), & \text{if } X_1 = \emptyset, s = |X_2|, \\ lo(X_2, |X_2| - s), & \text{if } X_1 = \emptyset, s < |X_2|, \\ lo(X_1, |X_1|), & \text{if } X_1 \neq \emptyset, s \geq |X_2|, \\ \max(lo(X_1, |X_1|), lo(X_2, |X_2| - s)), & \text{if } X_1 \neq \emptyset, s < |X_2|. \end{cases}$$

Bound Tightening

- For mean aggregation
  - Consider subproblem $z_{s_2,s_3} := \mathbf{mean}(X_1 \cup X_2' \cup X_3')$
    - $|X_2 \backslash X_2'| = s_2, |X_3'| = s_3$
  - Upper and lower bounds:
    - $\bar{z}_{s_2,s_3} = \overline{w}_{s_2,s_3}/n_{s_2,s_3}$, $\underline{z}_{s_2,s_3} = \underline{w}_{s_2,s_3}/n_{s_2,s_3}$ $\quad (n_{s_2,s_3} = |X_1| + |X_2| - s_2 + s_3)$

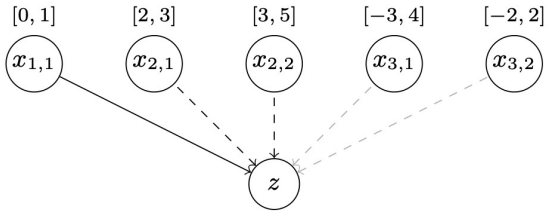
$$\overline{w}_{s_2,s_3} = \sum_{x \in X_1} \overline{x} + \sum_{1 \le i \le |X_2| - s_2} hi(X_2, i) + \sum_{1 \le i \le s_3} hi(X_3, i),$$

$$\underline{w}_{s_2,s_3} = \sum_{x \in X_1} \underline{x} + \sum_{1 \le i \le |X_2| - s_2} lo(X_2, i) + \sum_{1 \le i \le s_3} lo(X_3, i).$$

  - To obtain $\overline{z}$ and $\underline{z}$, at most $(s+1)^2$ subproblems to solve
    - We reduce the time complexity to $O((s+N)\log s)$ using the unimodality.

Bound Tightening

- An example of tightened bounds
  - $X_1 = \{x_{1,1}\},\ X_2 = \{x_{2,1}, x_{2,2}\},\ X_3 = \{x_{3,1}, x_{3,2}\}.$



	Plain	Tightened
max (ub)	$\max(1,3,5,4,2) = \mathbf{5}$	$\max(1,5,4) = \mathbf{5}$
max (lb)	$\min(0,2,3,-3,-2) = -3$	$\max(0,2) = \mathbf{2}$
mean (ub)	$\frac{1+5+4}{3} = 3.33$	$\frac{1+5+3+4}{4} = \mathbf{3.25}$
mean (lb)	$\frac{0-3-2}{3} = -1.67$	$\frac{0+2+3-3}{4} = \mathbf{0.5}$

- Theorem on the tightness of bounds

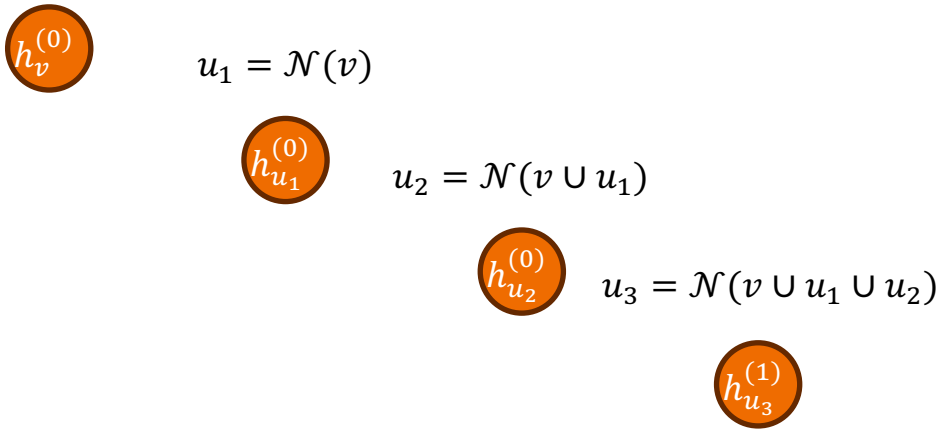
The upper and lower bounds provided in this section for max and mean aggregations are **tight**. That is, there exist sets $X_2' \subseteq X_2$ and $X_3' \subseteq X_3$ with $|X_2 \backslash X_2'| + |X_3'| \leq s$ that achieve these bounds exactly.

Incremental Constraint Solving

- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.

Incremental Constraint Solving

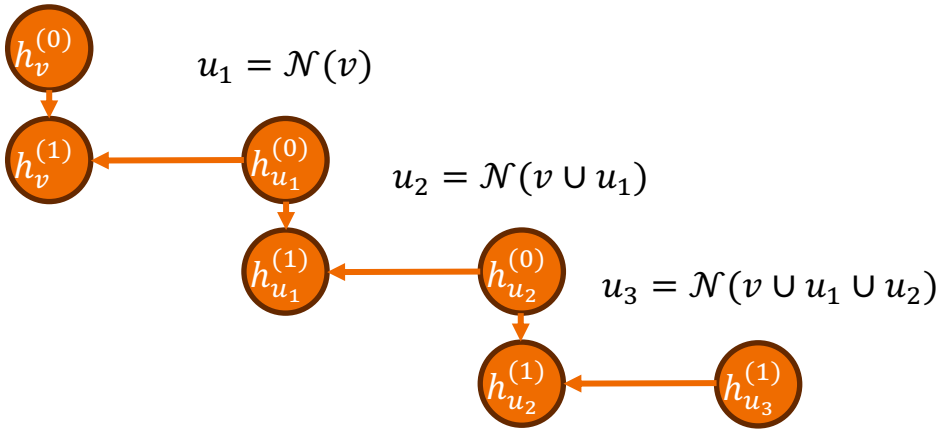
- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.
- Process of our algorithm
  - Forward bound propagation;



An example of a 3-layer GNN verification task.

Incremental Constraint Solving

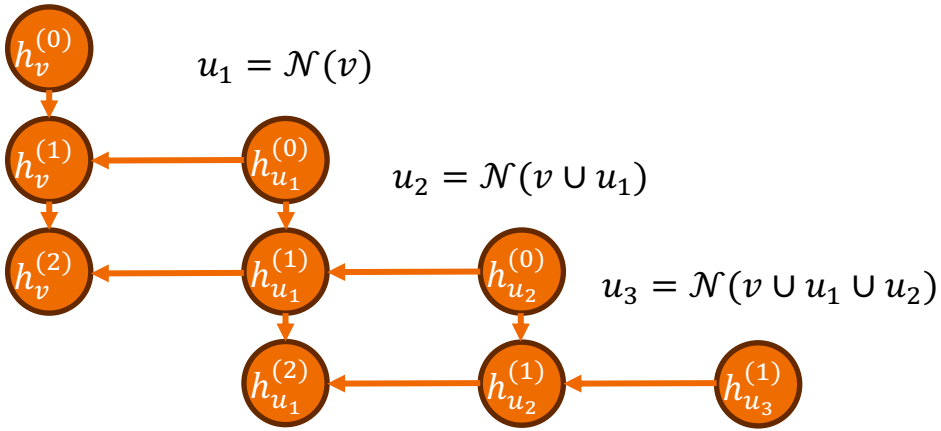
- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.
- Process of our algorithm
  - Forward bound propagation;



An example of a 3-layer GNN verification task.

Incremental Constraint Solving

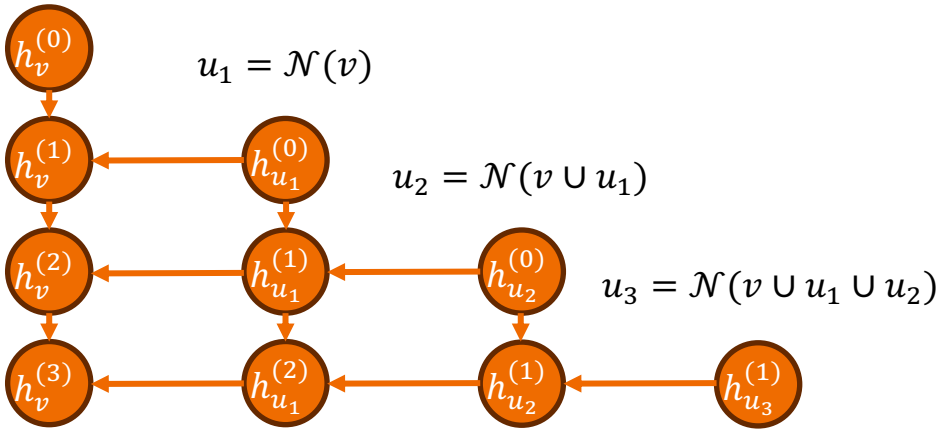
- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.
- Process of our algorithm
  - Forward bound propagation;



An example of a 3-layer GNN verification task.

Incremental Constraint Solving

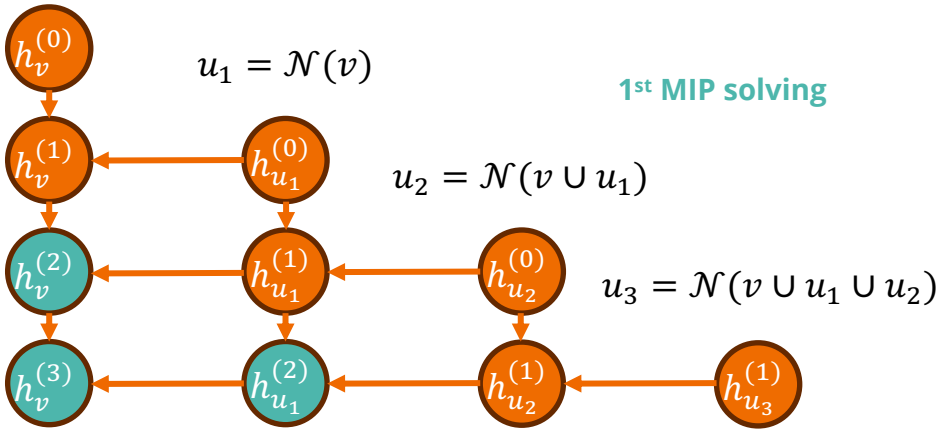
- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.
- Process of our algorithm
  - Forward bound propagation;



An example of a 3-layer GNN verification task.

Incremental Constraint Solving

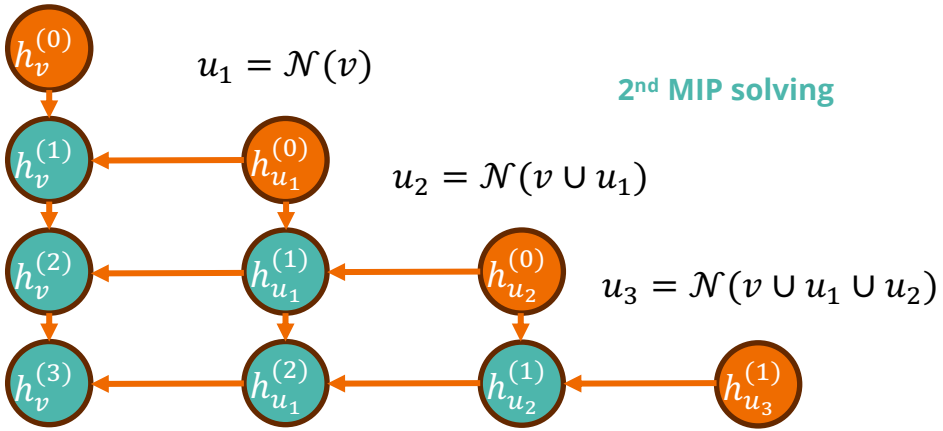
- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.
- Process of our algorithm
  - Forward bound propagation;
  - From layer K to 1:
    - Encode relaxation problem;
    - Incremental MIP solving;
    - If `unsat`, return `robust`;
  - Return `non-robust`.



An example of a 3-layer GNN verification task.

Incremental Constraint Solving

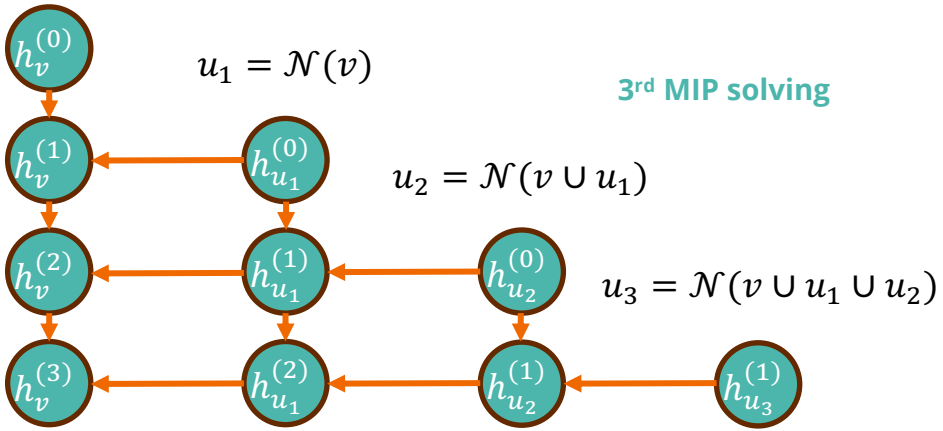
- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.
- Process of our algorithm
  - Forward bound propagation;
  - From layer K to 1:
    - Encode relaxation problem;
    - Incremental MIP solving;
    - If `unsat`, return `robust`;
  - Return `non-robust`.



An example of a 3-layer GNN verification task.

Incremental Constraint Solving

- Size of (sub)graphs affects the encoding complexity and the verification efficiency.
  - For a node t, it is common that $|\mathcal{N}_{k+1}(t)| \gg |\mathcal{N}_k(t)|$;
  - We use this structure to iteratively solve a sequence of relaxation problems;
  - Exactness is still maintained at the end, and performance is improved.
- Process of our algorithm
  - Forward bound propagation;
  - From layer K to 1:
    - Encode relaxation problem;
    - Incremental MIP solving;
    - If `unsat`, return `robust`;
  - Return `non-robust`.



An example of a 3-layer GNN verification task.

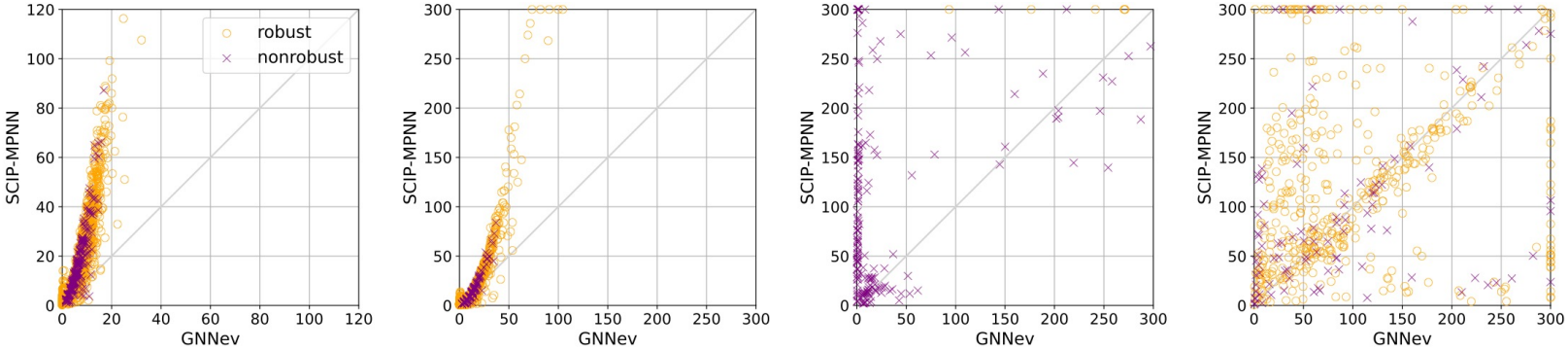
Experiments

- We implement our method as *GNNev*, an open-source exact verifier.
  - The underlying MIP solver is *Gurobi* 11.0.3;
  - Broad usability: Support 3 aggregators sum, max, and mean;
  - Wide attack scenarios: Support attribute perturbation, edge addition/deletion;
  - Easy to deploy: Accept models built by SAGEConv module in *PyG* [8].
- Experimental datasets

	Dataset	#Graphs	#Nodes	#Edges	#Attributes	#Classes
Node classification	Cora	1	2,708	5,429	1,433	7
	CiteSeer	1	3,312	4,715	3,703	6
	Amazon	1	11,944	351,216	25	2
	Yelp	1	45,954	98,630	32	2
Graph classification	MUTAG	188	17.9	39.6	7	2
	ENZYMES	600	32.6	124.3	3	6

Experiments

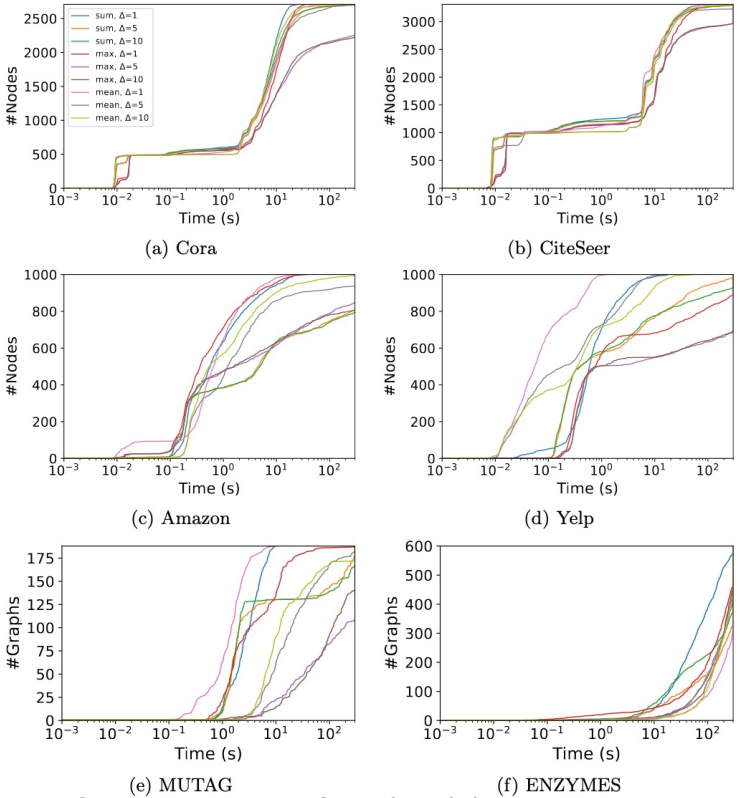
- Comparison with baselines on sum-aggregation
  - Exact Verification tasks on 3-layer GNNs with global budget $\Delta = 2$;
  - We take the best results from the two variants of *SCIP-MPNN* calling *Gurobi*;
  - Performance of *GNNev* is highly competitive, especially on node classification GNNs.



(a) Cora (b) CiteSeer (c) MUTAG (d) ENZYMES

Experiments

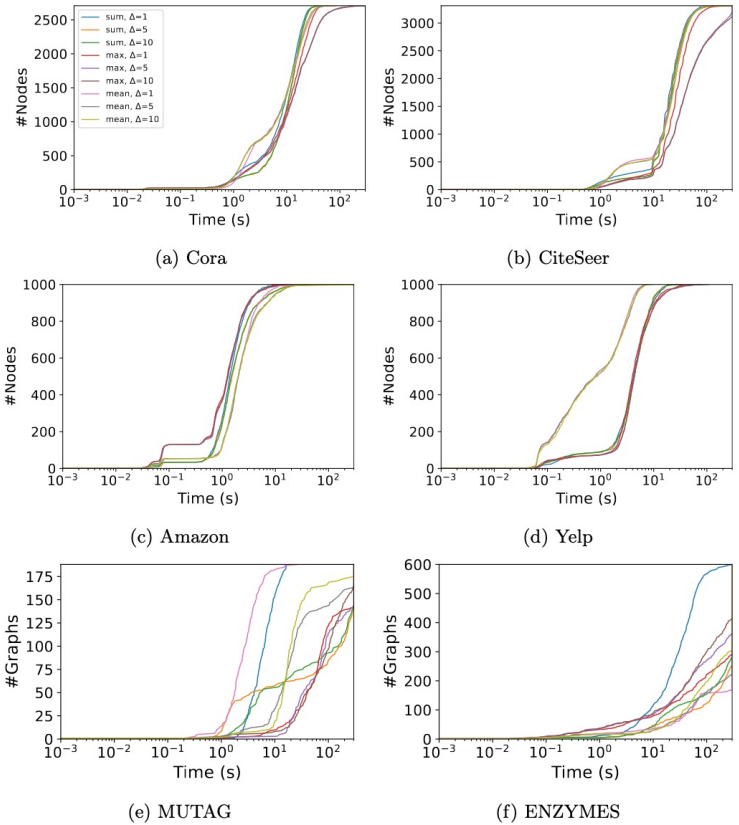
- Performance on all aggregators & perturbations
  - Exact Verification tasks on 3-layer GNNs with different global budgets;
  - For edge deletions:
    - Node classification tasks perform better than graph classification task;
    - Max aggregation is a challenge as big-M encoding is not efficient enough.



Verification time cost for edge deletions.

Experiments

- Performance on all aggregators & perturbations
  - Exact Verification tasks on 3-layer GNNs with different global budgets;
  - For edge deletions:
    - Node classification tasks perform better than graph classification task;
    - Max aggregation is a challenge as big-M encoding is not efficient enough.
  - For edge additions:
    - For performance reason, we only select 1-3 edges pointing to each node to build F;
    - *GNNev* shows similar trends with edge deletions.



Verification time cost for edge additions.

Experiments

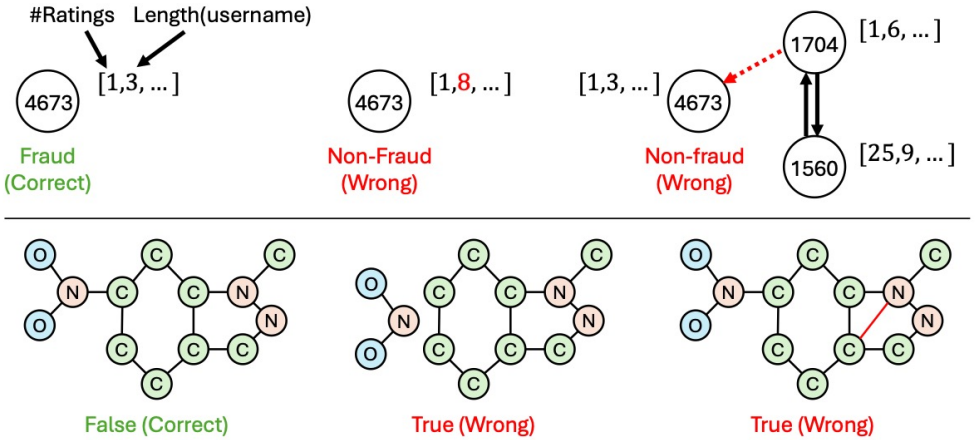
- Case study about adversarial robustness
  - *GNNev* finds that mean-aggregated GNNs are more vulnerable in general;
  - Two non-robust cases found by *GNNev* on real-world datasets
    - *GNNev* can gain insight into the susceptibility of GNNs to adversarial attacks.

Fraud detection (Amazon)

- Modifying a single attribute caused wrong prediction.

Biochemistry (MUTAG)

- Edge deletion/addition leads to invalid predictions.



Conclusion

- We propose a MIP-based exact robustness verification method for message-passing GNNs.
  - We support new common aggregators (max, mean) and diverse perturbation types.
  - Bound tightening and incremental constraint solving are applied.
- We implement an open-source, versatile, and efficient verifier *GNNev*.
  - *GNNev* outperforms MIP-based baselines in both efficiency and functionality.
  - Strong performance on real-world fraud and biochemical datasets.

Thank you!

Extended version on arXiv

Source code on GitHub

Reference

[1] William L. Hamilton, Zhitao Ying, Jure Leskovec: Inductive Representation Learning on Large Graphs. NIPS 2017.

[2] Daniel Zügner, Stephan Günnemann: Certifiable Robustness and Robust Training for Graph Convolutional Networks. KDD 2019.

[3] Aleksandar Bojchevski, Stephan Günnemann: Certifiable Robustness to Graph Perturbations. NeurIPS 2019.

[4] Hongwei Jin, Zhan Shi, Venkata Jaya Shankar Ashish Peruri, Xinhua Zhang: Certified Robustness of Graph Convolution Networks for Graph Classification under Topological Attacks. NeurIPS 2020.

[5] Daniel Zügner, Stephan Günnemann: Certifiable Robustness of Graph Convolutional Networks under Structure Perturbations. KDD 2020.

[6] Christopher Hojny, Shiqiang Zhang, Juan S. Campos, Ruth Misener: Verifying message-passing neural networks via topology-based bounds tightening. ICML 2024.

[7] Chia-Hsuan Lu, Tony Tan, Michael Benedikt: Robustness Verification of Graph Neural Networks Via Lightweight Satisfiability Testing. TACAS 2026.

[8] Matthias Fey, Jan Eric Lenssen: Fast Graph Representation Learning with PyTorch Geometric. ICLR Workshop on Representation Learning on Graphs and Manifolds 2019.