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

Christopher Hojny^{*1} Shiqiang Zhang^{*2} Juan S. Campos² Ruth Misener²

Abstract

Since graph neural networks (GNNs) are often vulnerable to attack, we need to know when we can trust them. We develop a computationally effective approach towards providing robust certificates for message-passing neural networks (MPNNs) using a Rectified Linear Unit (ReLU) activation function. Because our work builds on mixed-integer optimization, it encodes a wide variety of subproblems, for example it admits (i) both adding and removing edges, (ii) both global and local budgets, and (iii) both topological perturbations and feature modifications. Our key technology, topology-based bounds tightening, uses graph structure to tighten bounds. We also experiment with aggressive bounds tightening to dynamically change the optimization constraints by tightening variable bounds. To demonstrate the effectiveness of these strategies, we implement an extension to the open-source branch-and-cut solver SCIP. We test on both node and graph classification problems and consider topological attacks that both add and remove edges.

1. Introduction

Graph neural networks (GNNs) may have incredible performance in graph-based tasks, but researchers also raise concerns about their vulnerability: small input changes sometimes lead to wrong GNN predictions (Günemann, 2022). To study these GNN vulnerabilities, prior works roughly divide into two classes: adversarial attacks (Dai et al., 2018; Zügner et al., 2018; Takahashi, 2019; Xu et al., 2019; Zügner & Günemann, 2019b; Chen et al., 2020; Ma et al., 2020; Sun et al., 2020; Wang et al., 2020; Geisler et al., 2021) and certifiable robustness (Bojchevski & Günemann, 2019; Zügner & Günemann, 2019a; 2020; Bojchevski et al.,

2020; Jin et al., 2020; Sälzer & Lange, 2023). Beyond the difficulties of developing adversarial robustness for dense neural networks (Lomuscio & Maganti, 2017; Fischetti & Jo, 2018), incorporating graphs brings new challenges to both adversarial attacks and certifiable robustness. The first difficulty is defining graph perturbations because, beyond tuning the features (Takahashi, 2019; Zügner et al., 2018; Zügner & Günemann, 2019a; Bojchevski et al., 2020; Ma et al., 2020), an attacker may inject nodes (Sun et al., 2020; Wang et al., 2020) or add/delete edges (Dai et al., 2018; Zügner et al., 2018; Bojchevski & Günemann, 2019; Zügner & Günemann, 2019b; 2020; Xu et al., 2019; Chen et al., 2020; Jin et al., 2020; Geisler et al., 2021). Second, binary elements in the adjacency matrix create discrete optimization problems. Finally, perturbations to a node may indirectly attack other nodes via message passing or graph convolution.

Adversarial attacks aim to change the predictions of a GNN with admissible perturbations. In graph classification, the attacking goal is the prediction of a target graph (Dai et al., 2018; Chen et al., 2020). In node classification, attacks may be *local* (or targeted) and *global* (or untargeted). Local attacks (Dai et al., 2018; Zügner et al., 2018; Takahashi, 2019; Wang et al., 2020) try to change the prediction of a single node under perturbations, and global attacks (Xu et al., 2019; Zügner & Günemann, 2019b; Ma et al., 2020; Sun et al., 2020; Geisler et al., 2021) allow perturbations to a group of nodes. Except for the Q-learning approach and genetic algorithm of Dai et al. (2018), most aforementioned works are first-order methods which derive or approximate gradients w.r.t. features and edges. Binary variables are flipped when they are chosen to be updated.

Certifiable robustness tries to guarantee that the prediction will not change under any admissible GNN perturbation. The state-of-the-art (Bojchevski & Günemann, 2019; Zügner & Günemann, 2019a; 2020; Jin et al., 2020) typically formulates certifiable robustness as a constrained optimization problem, where the objective is the worst-case margin between the correct class and other class(es), and the constraints represent admissible perturbations (Günemann, 2022). Given a GNN and a target node/graph, a certificate requires proving that the objective is always positive. Any feasible solution with a negative objective is an adversarial attack. Most existing certificates (Zügner & Günemann, 2019a; Jin et al., 2020) focus on graph convolutional net-

^{*}Equal contribution ¹Eindhoven University of Technology, Eindhoven, The Netherlands ²Department of Computing, Imperial College London, UK. Correspondence to: Christopher Hojny <c.hojny@tue.nl>.

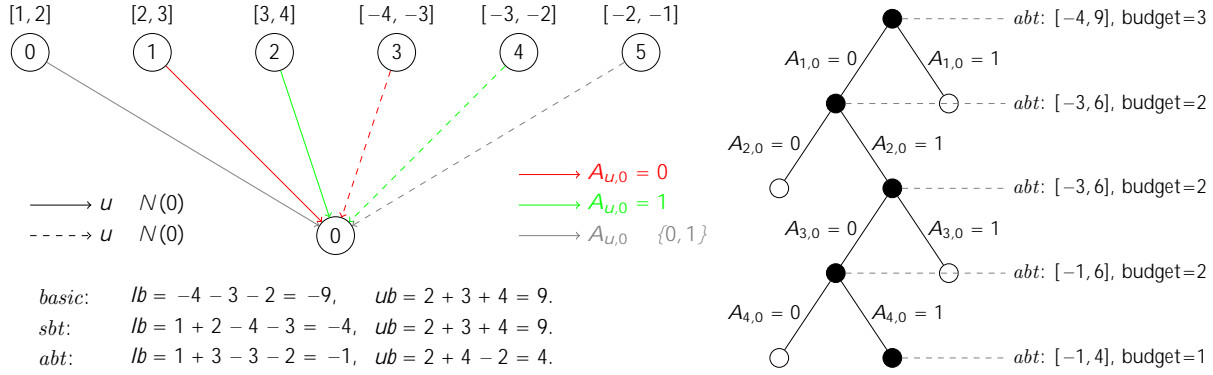


Figure 1: **(left)** Consider a graph with 6 nodes $u = 0; \dots; 5$ and one feature. The neighbor set of node 0 is $N(0) = \{0; 1; 2; 3; 4; 5\}$. The input bounds are given above each node. Assume the budget, i.e., maximal number of modifications, for node 0 is 3. The modifications could be removing neighbors from $N(0)$ or adding new neighbors from $\{3; 4; 5\}$. Four decisions have been made in the branch-and-bound tree, i.e., binary variables representing edges are set as $A_{1,0} = 0; A_{2,0} = 1; A_{3,0} = 0; A_{4,0} = 1$. Since node 2 is a neighbor of node 0 while node 3 is not, fixing $A_{2,0} = 1$ and $A_{3,0} = 0$ spends no budget. For each method, we compute the bounds for node 0 in the next layer. To compute a lower bound, the plain strategy (*basic*) chooses all negative lower bounds without considering either budgets or previous decisions in the branch-and-bound tree. Static bounds tightening (*sbt*), the first topology-based bounds tightening routine, removes node 2 and adds node 3; 4 as neighbors within 3 budgets, but ignores decisions in the branch-and-bound tree. Aggressive bounds tightening (*abt*) yields tighter bounds by saving node 0 and adding node 5 as neighbors. **(right)** The branch-and-bound tree corresponding to the left. We provide the bounds yielded from *abt* and budget left after each decision.

works (GCNs) (Kipf & Welling, 2017). The certificate on personalized propagation of neural predictions (Gasteiger et al., 2019) relies on local budget and yields looser guarantees in the presence of a global budget (Bojchevski & Günnemann, 2019). Also, each certificate has specific requirements on the perturbation type, e.g., changing node features only (Zügner & Günnemann, 2019a), modifying graph structure only (Bojchevski & Günnemann, 2019; Jin et al., 2020), removing edges only (Zügner & Günnemann, 2020), and allowing only orthogonal Gromov-Wasserstein threats (Jin et al., 2022). Instead of verifying GNNs directly, several works (Bojchevski et al., 2020; Wang et al., 2021; Xia et al., 2024) provide certified defenses based on the randomized smoothing framework (Cohen et al., 2019).

This work develops certificates on the classic message passing framework, especially GraphSAGE (Hamilton et al., 2017). Using a recently-proposed mixed-integer formulation for GNNs (McDonald et al., 2024; Zhang et al., 2024; 2023), we directly encode a GNN into an optimization problem using linear constraints. Many perturbations are compatible with our formulation: (i) both adding and removing edges, (ii) both global and local budgets, and (iii) both topological perturbations and feature modifications.

When verifying fully-dense, feed-forward neural networks with ReLU activation, prior work shows that tightening variable bounds in a big-M formulation (Anderson et al., 2020) may lead to better computational performance (Tjeng et al., 2019; Botoeva et al., 2020; Tsay et al., 2021; Badilla et al.,

2023; Zhao et al., 2024). Since tighter variable bounds may improve the objective value of relaxations of the big-M formulation, they may be useful when providing a certificate of robustness. Because the optimization problems associated with verifying MPNNs are so large, this work cannot use the tighter, convex-hull based optimization formulations (Singh et al., 2019a; Tjandraatmadja et al., 2020; Müller et al., 2022). Instead, we use what we call *topology-based bounds tightening* to enable a much stronger version of feasibility-based bounds tightening: we extend SCIP (Bestuzheva et al., 2023) to explicitly use the graph structures. We also develop an *aggressive bounds tightening* (Belotti et al., 2016) routine to dynamically change the optimization constraints by tightening variable bounds within SCIP. Key outcomes include: (i) solving literature node classification instances in a fraction of a second, (ii) solving an extra 266 graph classification instances after implementing topology-based bounds tightening in SCIP, and (iii) making the open-source solver SCIP nearly as performant as the commercial solver Gurobi, e.g., improving the time penalty of the open-source solver from a factor of 10 to a factor of 3 for robust instances.

The paper begins in Section 2 by defining a mixed-integer encoding for MPNNs. Section 3 presents the verification problem and develops our two topology-based bounds tightening routines, static bounds tightening *sbt* and aggressive bounds tightening *abt*. Section 4 presents the numerical experiments and Section 5 concludes. Figure 1 represents a toy example showing the basic approach in comparison to our two topology-based approaches *sbt* and *abt*.

2. Definition & Encoding of MPNNs

We inherit the mixed-integer formulation of MPNNs from Zhang et al. (2024) and formulate ReLU activation using big-M (Anderson et al., 2020). Consider a trained GNN:

$$f: \mathbb{R}^N \rightarrow \mathbb{R}^N; \quad f_0; 1g^N \quad \mathbb{R}^N \rightarrow \mathbb{R}^{d_L} \quad (1)$$

$$(X; A) \mapsto f(X; A)$$

whose l -th layer with weights $w_{u!}^{(l)}$ and biases $b_v^{(l)}$ is:

$$x_v^{(l)} = \bigoplus_{u \in N(v)} w_{u!}^{(l)} x_u^{(l-1)} + b_v^{(l)} \quad (2)$$

where $v \in V$, $V = \{0; 1; \dots; N-1\}g$ is the node set, $N(v)$ is the neighbor set of node v , and \cdot is activation. Given input features $f x_v^{(0)} g_{v \in V}$ and the graph structure, we can derive the hidden features $f x_v^{(l)} g_{v \in V}$; $x_v^{(l)} \in \mathbb{R}^{d_l}$. When $l = L$, we obtain the node representation $x_v^{(L)}$ of each node.

2.1. Big-M formulation for MPNNs

When the graph structure is not fixed, we need to include all possible contributions from all nodes, i.e., the l -th layer is:

$$x_v^{(l)} = \bigoplus_{u \in V} A_{u,v} w_{u!}^{(l)} x_u^{(l-1)} + b_v^{(l)} \quad (3)$$

where $A_{u,v} \in \{0; 1\}g$ controls the existence of edge $u \rightarrow v$.

With fixed weights and biases, we still need to handle the nonlinearities caused by (i) bilinear terms $A_{u,v} x_u^{(l-1)}$, and (ii) activation \cdot . Let $x_v^{(l)} = \bigoplus_{u \in V} A_{u,v} w_{u!}^{(l)} x_u^{(l-1)} + b_v^{(l)}$, the Zhang et al. (2023) big-M formulation introduces auxiliary variables $x_{u!}^{(l)}$ to replace the bilinear terms $A_{u,v} x_u^{(l-1)}$ and linearly encodes $x_v^{(l)}$:

$$x_v^{(l)} = \bigoplus_{u \in V} w_{u!}^{(l)} x_{u!}^{(l)} + b_v^{(l)} \quad (4)$$

Let $F_l := \{0; 1; \dots; d_l-1\}g$ and denote the f -th element of $x^{(l)}$ by $x_{v,f}^{(l)}$; $f \in F_l$. Use $lb(\cdot)$ and $ub(\cdot)$ to represent the lower and upper bound of a variable, respectively. Then $x_{u!}^{(l)}$; $f \in F_l$ is equivalently formulated in the following big-M constraints:

$$x_{u!}^{(l)} \in [lb(x_{u,f}^{(l-1)}), A_{u,v} x_{u,f}^{(l-1)}] \quad (5a)$$

$$x_{u!}^{(l)} \in [A_{u,v} x_{u,f}^{(l-1)}, ub(x_{u,f}^{(l-1)})] \quad (5b)$$

$$x_{u!}^{(l)} \in [x_{u,f}^{(l-1)}, lb(x_{u,f}^{(l-1)})] \quad (1 - A_{u,v}) \quad (5c)$$

$$x_{u!}^{(l)} \in [x_{u,f}^{(l-1)}, ub(x_{u,f}^{(l-1)})] \quad (1 - A_{u,v}) \quad (5d)$$

2.2. Big-M formulation for ReLU

When using ReLU as the activation, i.e.,

$$x_{v,f}^{(l)} = \max\{0; x_{v,f}^{(l)} g\} \quad (6)$$

Anderson et al. (2020) proposed a big-M formulation:

$$x_{v,f}^{(l)} \in [0, x_{v,f}^{(l)}] \quad (7a)$$

$$x_{v,f}^{(l)} \in [x_{v,f}^{(l)}, x_{v,f}^{(l)}] \quad (7b)$$

$$x_{v,f}^{(l)} \in [x_{v,f}^{(l)}, lb(x_{v,f}^{(l)})] \quad (1 - v_{v,f}^{(l)}) \quad (7c)$$

$$x_{v,f}^{(l)} \in [ub(x_{v,f}^{(l)}), v_{v,f}^{(l)}] \quad (7d)$$

where $v_{v,f}^{(l)} \in \{0; 1\}g$ controls the on/off of the activation:

$$x_{v,f}^{(l)} = \begin{cases} 0; & v_{v,f}^{(l)} = 0 \\ x_{v,f}^{(l)}; & v_{v,f}^{(l)} = 1; \end{cases} \quad (8)$$

2.3. Bounds propagation

Eqs. (5) and (7) show the importance of variable bounds $lb(\cdot)$ and $ub(\cdot)$ or *big-M parameters*. Given the input feature bounds, we define bounds for auxiliary variables $x_{u!}^{(l)}$ and post-activation variables $x_{v,f}^{(l)}$. Using $x_{u!}^{(l)}$; $f \in F_l = A_{u,v} x_{u,f}^{(l-1)}$, the bounds of $x_{u!}^{(l)}$ are:

$$lb(x_{u!}^{(l)}) = \min\{0; lb(x_{u,f}^{(l-1)})g\} \quad (9a)$$

$$ub(x_{u!}^{(l)}) = \max\{0; ub(x_{u,f}^{(l-1)})g\} \quad (9b)$$

with which we can use arithmetic propagation or feasibility-based bounds tightening to obtain bounds of $x_{v,f}^{(l)}$ based on Eq. (4). Then, we use Eq. (6) to bound $x_{v,f}^{(l)}$:

$$lb(x_{v,f}^{(l)}) = \max\{0; lb(x_{v,f}^{(l)})g\} \quad (10a)$$

$$ub(x_{v,f}^{(l)}) = \max\{0; ub(x_{v,f}^{(l)})g\} \quad (10b)$$

Without extra information, bounds defined in Eqs. (9) and (10) are the tightest possible which derive from interval arithmetic. However, in a branch-and-bound tree, more and more variables will be fixed, which provides the opportunity to tighten the bounds. Additionally, in specific applications such as verification, the graph domain is restricted, allowing us to derive tighter bounds.

Remark 2.1. A MPNN with L message passing steps is suitable for node-level tasks. For graph-level tasks, there is usually a pooling layer after message passing to obtain a global representation and several dense layers thereafter as a final regressor/classifier. We omit these formulations since (i) linear pooling, e.g., mean and sum, is easily incorporated into our formulation, and (ii) dense layers are a special case of Eq. (2) with a single node.

3. Verification of MPNNs

3.1. Problem definition

First, consider node classification. Given a trained MPNN defined as Eq. (2), the number of classes is the number of output features, i.e., $C = d_L$, and the predicted label of node t corresponds to the maximal logit, i.e., $c = \max_{c \in C} f_{t,c}(X; A)$. Given an input $(X; A)$ consisting of features X and adjacency matrix A , denote its predictive label for a target node t as c . The worst case margin between predictive label c and attack label \hat{c} under perturbations $P(\cdot)$ is:

$$m^t(c; \hat{c}) := \min_{(X; A)} f_{t,c}(X; A) - f_{t,\hat{c}}(X; A) \quad (11)$$

$$s.t: X \geq P(X); A \geq P(A);$$

A positive $m^t(c; \hat{c})$ means that the logit of class c is always larger than class \hat{c} . If $m^t(c; \hat{c}) > 0$, then any admissible perturbation can not change the label assigned to node t , that is, this MPNN is robust to node t .

For graph classification, instead of considering a single node, we want to know the worst case margin between two classes for a target graph, i.e.,

$$m(c; \hat{c}) := \min_{(X; A)} f_c(X; A) - f_{\hat{c}}(X; A) \quad (12)$$

$$s.t: X \geq P(X); A \geq P(A);$$

In both problems, the target graph $(X; A)$ and predictive label c are fixed. For node classification, the target node t is also given. Therefore, we omit t in Eq. (11) and reduce both problems to one as shown in Eq. (12).

3.2. Admissible perturbations

The perturbations on features and edges can be described similarly. Locally, we may only change features/edges for each node with a given local budget. Also, there is typically a global budget for the number of changes. The feature perturbations are typically easier to implement since they will not hurt the message passing scheme, i.e., the graph structure is fixed. In such settings, there is no need to use the mixed-integer formulations for MPNNs in Section 2.1 since a message passing step is actually simplified as a dense layer. Since feature perturbations are well-studied (Zügner & Günnemann, 2019a) and our proposed bounds tightening techniques mainly focus on changeable graph structures, our computational results only consider the perturbations on the adjacency matrix.

We first define the admissible perturbations for undirected graphs, which admits both adding and removing edges. Denote the global budget by Q and local budget to node v by

Algorithm 1 Static bounds tightening (*sbt*)

Input: Input features $\mathbf{x}_v^{(0)}$, weights $w_{uv}^{(l)}$, biases $b_v^{(l)}$.
Initialize $lb(\mathbf{x}^{(0)}) = ub(\mathbf{x}^{(0)}) = \mathbf{x}^{(0)}$.

for $l = 1$ **to** L **do**

 Get $lb(\mathbf{x}^{(l)})$ using Eq. (17). *fbasic* uses Eq. (20).*g*

 Get $lb(\mathbf{x}^{(l)})$ using Eq. (10).

 Get $ub(\mathbf{x}^{(l)})$ and $ub(\mathbf{x}^{(l)})$ in a similar way.

end for

q_v , then the perturbations $P_1(A)$ are defined as:

$$P_1(A) = fA \geq f0; 1g^N \quad N \quad j \quad A = A^T; \quad (13)$$

$$kA \quad A \quad k_0 \quad 2Q;$$

$$kA_v \quad A_v k_0 \quad q_v; \delta v \geq Vg$$

where A_v is the v -th column of A . $P_1(\cdot)$ will be used in graph classification since the graphs in benchmarks are usually undirected and relatively small.

For node classification, literature benchmarks usually (i) are large directed graphs, e.g., 3000 nodes, (ii) have many node features, e.g., 3000 features, (iii) have small average degree, e.g., 1–3. If admitting adding edges, then the graph domain is too large to optimize over. Therefore, the state-of-the-art (Zügner & Günnemann, 2020) only considers removing edges, where a L -hop neighborhood around the target node t is sufficient. For a MPNN with L message passing steps without perturbations, nodes outside a L -hop neighborhood cannot affect the prediction of t . Since adding edges is not allowed, the L -hop neighborhood of t after perturbations is always a subset of the unperturbed neighborhood. Similar to the literature, we define a more restrictive perturbation space for large graphs:

$$P_2(A) = fA \geq f0; 1g^N \quad N \quad j \quad (14)$$

$$A_{u,v} \quad A_{u,v}; \delta u; v \geq V;$$

$$kA \quad A \quad k_0 \quad Q;$$

$$kA_v \quad A_v k_0 \quad q_v; \delta v \geq Vg$$

where global budget is replaced by Q since the graph is directed. $P_2(\cdot)$ will be used in node classification. For our later analysis, however, we focus on $P_1(\cdot)$ since $P_2(\cdot)$ is more like a special case without adding edges.

3.3. Static bounds tightening

Note that large budgets in Eq. (13) make the verification problems meaningless since the perturbed graph could be any graph. The very basic assumption is that the perturbed graph is similar to the original one, which brings us to propose the first bounds tightening approach. The rough idea is to consider budgets when computing bounds of $\mathbf{x}_v^{(l)}$ based on bounds of $\mathbf{x}_{u|v}^{(l-1)}$ in Eq. (4). Instead of considering

all contributions from all nodes, we first calculate the bounds based on original neighbors and then maximally perturb the bounds with given budgets. Mathematically, $lb(x_{v;f^0}^{(l)})$ is found by solving the following optimization problem:

$$\begin{aligned} \min_{A; \mathbf{x}^{(l-1)}} & \prod_{u \in \mathcal{N}(v)} A_{u,v} \prod_{f \in \mathcal{F}_{l-1}} W_{u!v;f!f^0}^{(l)} x_{u;f}^{(l-1)} + b_{v;f^0}^{(l)} \\ s:t: & A \geq P_1(A) \\ & \mathbf{x}^{(l-1)} \geq [lb(\mathbf{x}^{(l-1)}); ub(\mathbf{x}^{(l-1)})] \end{aligned} \quad (15)$$

where $\mathbf{x}^{(l-1)} := \{x_{u;f}^{(l-1)}\}_{u \in \mathcal{N}(v); f \in \mathcal{F}_{l-1}}$, $W_{u!v;f!f^0}^{(l)}$ is the $(f; f^0)$ -th element in $W_{u!v}^{(l)}$.

Remark 3.1. For brevity, we omit the superscripts of layers for all variables, and subscripts of edges in weights, i.e., rewriting Eq. (15) as:

$$\begin{aligned} lb(x_{v;f^0}) = \min_{A; \mathbf{x}} & \prod_{u \in \mathcal{N}(v)} A_{u,v} \prod_{f \in \mathcal{F}_{l-1}} W_{f;f^0} x_{u;f} + b_{v;f^0} \\ s:t: & A \geq P_1(A) \\ & \mathbf{x} \geq [lb(\mathbf{x}); ub(\mathbf{x})] \end{aligned} \quad (16)$$

Property 3.2. Eq. (16) is equivalent to:

$$lb(x_{v;f^0}) = \prod_{u \in \mathcal{N}(v)} lb_{u!v} + b_{v;f^0} + \min_{j \in V_{lb} \setminus q_v} \prod_{u \in \mathcal{N}(v)} x_{u!v} \quad (17)$$

where $\mathcal{N}(v)$ denotes the original neighbor set of node v , $lb_{u!v}$ represents the contribution of node u to the lower bound of node v when u is a neighbor of v , $x_{u!v}$ denotes the change of lower bound of node v caused by modifying edge $u!v$, V_{lb} is the set of nodes consisting of removed/added neighbors of node v .

Calculating $lb_{u!v}$ is straightforward:

$$\begin{aligned} lb_{u!v} = & \prod_{f \in \mathcal{F}_{l-1}} W_{f;f^0} |_{W_{f;f^0} > 0} lb(x_{u;f}) \\ & + \prod_{f \in \mathcal{F}_{l-1}} W_{f;f^0} |_{W_{f;f^0} < 0} ub(x_{u;f}) \end{aligned} \quad (18)$$

which is used to derive $x_{u!v}$ as:

$$x_{u!v} = \begin{cases} lb_{u!v}; & u \in \mathcal{N}(v) \\ ub_{u!v}; & u \notin \mathcal{N}(v) \end{cases} \quad (19)$$

where two cases correspond to removing neighbor u and adding u as a neighbor, respectively. Furthermore, the last minimal term in Eq. (17) is equivalent to choosing at most q_v smallest negative terms among $\{x_{u!v}\}_{u \in \mathcal{N}(v)}$. The time complexity to compute lower bounds following Eq. (17) for each feature in l -th layer is $O(N^2 d_{l-1} d_l + N \log N)$. Upper bounds could be defined similarly, which are not included here due to space limitation.

Algorithm 2 Aggressive bounds tightening (*abt*)

Input: Input features $\mathbf{x}_v^{(0)}$, weights $W_{u!v}^{(l)}$, biases $b_v^{(l)}$.
Initialize $lb(\mathbf{x}^{(0)}) = ub(\mathbf{x}^{(0)}) = \mathbf{x}^{(0)}$.

for each node in branch-and-bound tree **do**

for $l = 1$ to L **do**

 Update q_v^l using Eq. (23).

 Update $lb(\mathbf{x}^{(l)})$ using Eq. (22).

 Update $ub(\mathbf{x}^{(l)})$ using Eq. (10).

 Update $lb(\mathbf{x}^{(l)})$ and $ub(\mathbf{x}^{(l)})$ in a similar way.

end for

end for

Remark 3.3. The plain strategy without considering graph structure and budgets *basic* is:

$$lb(x_{v;f^0}) = \prod_{u \in \mathcal{N}(v)} \min_{f \in \mathcal{F}_{l-1}} W_{u!v;f!f^0} x_{u;f} + b_{v;f^0} \quad (20)$$

and the time complexity is $O(N^2 d_{l-1} d_l)$.

Algorithm 1 calculates *sbt* bounds (and *basic* bounds) in a single forward pass of the model. As shown in the Figure 1 example, the bounds derived from *basic* is $[9; 9]$, which is improved to $[4; 9]$ after applying static bounds tightening *sbt*.

3.4. Aggressive bounds tightening

Consider any node in a branch-and-bound tree, values of several $A_{u,v}$ are already decided during the path from root to current node, with which we can further tighten bounds in the subtree rooted by this node. Belotti et al. (2016) refer to the idea of tightening bounds in the branch-and-bound tree as *aggressive bounds tightening*. In MPNN verification, there are three types of $A_{u,v}$ in Eq. (16): (i) $A_{u,v}$ is fixed to 0, (ii) $A_{u,v}$ is fixed to 1, and (iii) $A_{u,v}$ is not fixed yet. Denote $V_0 = \{u \in \mathcal{N}(v) \mid A_{u,v} = 0\}$ and $V_1 = \{u \in \mathcal{N}(v) \mid A_{u,v} = 1\}$, then Eq. (16) in the current node is restricted as:

$$\begin{aligned} lb(x_{v;f^0}) = \min_A & \prod_{u \in \mathcal{N}(v)} A_{u,v} lb_{u!v} + b_{v;f^0} \\ s:t: & A \geq P_1(A) \\ & A_{u,v} = 0; \quad \forall u \in V_0 \\ & A_{u,v} = 1; \quad \forall u \in V_1 \end{aligned} \quad (21)$$

Property 3.4. Eq. (21) is equivalent to:

$$\begin{aligned} lb(x_{v;f^0}) = & \prod_{u \in (\mathcal{N}(v) \setminus V_0) \cap V_1} lb_{u!v} + b_{v;f^0} \\ & + \min_{V_{lb} \setminus V_0 \cap V_1} \prod_{j \in V_{lb} \setminus q_v} x_{u!v} \end{aligned} \quad (22)$$

where q_v^l is the currently available budget for node v .

In Eq. (22), the first term sums over $(N(v) \cap V_0) \setminus V_1$ to represent the contributions from current neighbors. The last term excludes all fixed edges and only considers changeable neighbors. Similarly, this minimal term equals to choose at most q_v^0 smallest negative terms among $\sum_{u \in V_1} g_{u2V \cap (V_0 \setminus V_1)}$. q_v^0 is derived from the remaining local and global budgets:

$$q_v^0 = \min_{Q} \left(\sum_{v \in V_0} e_r(v) + \sum_{u \in V_1} e_a(u) g \right) \quad (23)$$

where $e_r(v) := |N(v) \setminus V_0|$ is the number of removed edges around node v , $e_a(v) := |N(v) \cap V_1|$ is the number of added edges around node v .

Algorithm 2 describes the *abt* steps. Since we derive *abt* bounds within the branch-and-bound tree (when solvers will not directly change variable bounds), we add local cutting planes to implement *abt* bounds (see Appendix B example).

Remark 3.5. The bounds tightening inside the branch-and-bound tree can be interpreted as applying the Section 3.3 bounds tightening to a modified target graph with a reduced budget. The spent budget changes the neighbors of node v from $N(v)$ to $(N(v) \cap V_0) \setminus V_1$.

As shown in Figure 1, aggressive bounds tightening *abt* gives tighter bounds [1;4] comparing to *basic* and *sbt*. The branch-and-bound tree in Figure 1 shows *abt* in action.

Table 1: Information on benchmarks. For multiple graphs, we compute the average number of nodes and edges.

benchmark	#graphs	#nodes	#edges	#features	#classes
MUTAG	188	17.9	39.6	7	2
ENZYMES	600	32.6	124.3	3	6
Cora	1	2708	5429	1433	7
CiteSeer	1	3312	4715	3703	6

3.5. Bounds tightening for ReLU

Aggressive bounds tightening could also tighten ReLU interval bounds in feed-forward neural networks (NNs). Although we did not directly tighten ReLU bounds in MPNNs, local cutting planes representing tighter big-M coefficients $lb(x_{v,f}^{(l)}); ub(x_{v,f}^{(l)})$ in Eq. (7) are added after applying *abt*. For each ReLU, if its pre-activation variable $x_{v,f}^{(l)}$ has a non-negative lower bound or a non-positive upper bound, the binary variable $\delta_{v,f}^{(l)}$ controlling on/off of this ReLU will be fixed due to these local cutting planes. Therefore, *abt* implies a dynamic tightening on ReLU interval bounds in MPNNs. This idea could be applied to NNs since a NN is an MPNN with a single node in the graph.

For ReLU NNs, bounds tightening techniques yielding tighter bounds than interval arithmetic include: FastLin (Weng et al., 2018), CROWN (Zhang et al., 2018), DeepPoly (Singh et al., 2018; 2019b), and optimization-based bound tightening (OBBT) (Tjeng et al., 2019; Tsay et al., 2021). But these techniques are rarely applied in GNN verification. Although a few works (Zügner & Günnemann, 2019a; Jin et al., 2020) involve convex relaxations for ReLUs, they do not tighten bounds for ReLUs.

OBBT is the only one of the existing advanced approaches which could immediately apply to GNNs. OBBT yields tighter bounds with high computational cost of solving many linear programs (LPs) or mixed-integer programs (MIPs). Incorporating computationally-effective methods like FastLin, CROWN or DeepPoly is difficult for GNNs because we lose the linearity between layers. Such linearity is crucial to these approaches, e.g., the linear lower/upper bounds in FastLin/CROWN, or the zonotopy abstraction of DeepPoly. When the input graph structure is fixed, bounds tightening on ReLUs in GNNs is equivalent to its counterpart in NNs (Wu et al., 2022). But for non-fixed graphs, the links between two adjacent layers in GNNs are controlled by an adjacency matrix, whose elements are binary variables. So it is considering topological perturbations that loses the linearity between layers and inspires *sbt* and *abt*.

4. Experiments

This section empirically evaluates the impact of static and aggressive bounds tightening on verifying MPNNs by solving a mixed-integer program (MIP) as shown in Section 2 and Section 3. All GNNs are built and trained using PyG (PyTorch Geometric) 2.1.0 (Fey & Lenssen, 2019). All MIPs are implemented in C/C++ using the open-source MIP solver SCIP 8.0.4 (Bestuzheva et al., 2023); all LP relaxations are solved using Soplex 6.0.4 (Gamrath et al., 2020). We used the GNN pull request (Zhang et al., 2024) in the Optimization and Machine Learning Toolkit OMLT (Ceccon et al., 2022) to debug the implementation. Observe that we could have alternatively extended a similar tool in SCIP (Turner et al., 2023). For each verification problem, we apply the basic implementation (SCIPbasic), static bounds tightening (SCIPabt), and aggressive bounds tightening (SCIPsbt). Our experiments also include a basic and static bounds tightening implementation of Gurobi 10.0.3 (GRBbasic, GRBsbt) (Gurobi Optimization, LLC, 2023). The code is available at GitHub, also see Hojny & Zhang (2024).

4.1. Implementation details

Our code models and solves the verification problem in three stages (for basic and static bounds tightening) or four stages (for aggressive bounds tightening). First, parameters

Table 2: Summary of results for graph classification with local attack strength $s = 2$. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 600 for ENZYMES and 188 for MUTAG, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

benchmark	method	all instances				robust instances			
		#	avg-time(s)	sgm-time(s)	# solved	#	avg-time(s)	sgm-time(s)	# solved
ENZYMES	SCIPbasic	5915	605.97	37.81	5579	3549	278.58	12.51	3444
	SCIPsbt	5915	230.59	21.26	5831	3549	82.89	6.65	3528
	SCIPabt	5915	246.02	21.77	5817	3549	88.95	6.71	3522
	GRBbasic	5915	86.03	7.59	5892	3549	32.80	2.82	3542
	GRBsbt	5915	74.87	7.09	5895	3549	22.90	2.50	3548
	MUTAG	SCIPbasic	1589	679.86	189.75	1575	44	798.47	202.93
	SCIPsbt	1589	196.07	75.17	1589	44	336.41	100.86	44
	SCIPabt	1589	207.50	82.43	1589	44	238.10	91.11	44
	GRBbasic	1589	34.58	4.06	1589	44	162.25	12.11	44
	GRBsbt	1589	59.93	22.40	1589	44	73.78	15.00	44

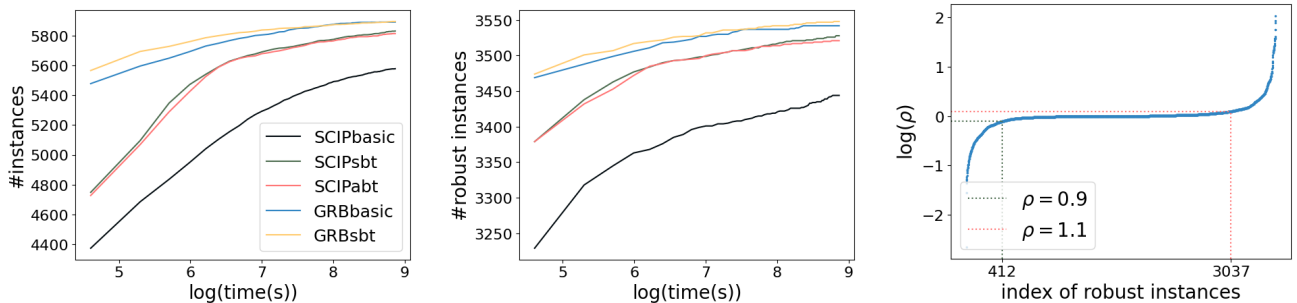


Figure 2: ENZYMES benchmark. **(left)** Number of instances solved by each method below different time costs. **(middle)** Number of robust instances solved by each method below different time costs. **(right)** Consider ρ , the ratio of time cost between SCIPabt and SCIPsbt on each robust instance. SCIPabt is at least 10% faster than SCIPsbt on 412 robust instances.

of a trained MPNN (weights, biases) and the verification problem (predictive label c , attack label \mathcal{C} , global budget Q , local budget q_v) are read. Second, lower and upper bounds on the variables in Eqs. (5) and (7) are computed for SCIPbasic and SCIPsbt. Third, a MIP model of the verification problem is created and solved. We do not solve the MIP models to global optimality as we only ask: *Is this instance robust or not?* We interrupt the optimization once a solution with negative objective value is found, i.e., the instance is non-robust, or the dual bound of the branch-and-bound tree is positive, i.e., the instance is robust.

In case of SCIPabt, the model is created as for static bounds tightening. During the solving process, a fourth step takes place. This step collects, at each node of the SCIP branch-and-bound tree, the A -variables that have been fixed to 0 and 1. We then iterate through the layers of the MPNN and, for

each layer, we: (i) recompute the variable bounds as in the second step for the current layer, taking the fixed variables into account following Section 3.4; (ii) check if any inequality in Eqs. (5) and (7) using the newly computed bounds is violated by the current linear programming solution; (iii) if a violated inequality is detected, we add this inequality as a local cutting plane to the model. That is, we inform SCIP that the inequality is only valid at the current node of the branch-and-bound tree and all its children, which is necessary as the inequality is based on fixed variables at the current node. The separation of local cutting planes has been implemented via a separator callback in SCIP.

To compare with the interval arithmetic approaches, we implemented an OBBT routine. For each variable, we changed the objective to the variable’s lower or upper bound, relaxed all binary variables, and solved the resulting LP.

(a) SCIPbasic (b) SCIPsbt (c) SCIPabt

Figure 3: For each SCIP-based method with local attack strength α on ENZYMES (the first row) and MUTAG (the second row), we count the number of robust graphs (green), nonrobust graphs (red), and time out (white). The percentage of the number of edges is the global budget.

Besides our implementation in SCIP, we also conducted experiments using Gurobi 10.0.3 (Gurobi Optimization, LLC, 2023) to compare baseline (GRBbasic) and static bounds tightening (GRBsbt). We used our SCIP implementation to create the MIP models, store them on the hard drive, and read them via Gurobi's Python interface. We did not investigate aggressive bounds tightening in Gurobi as Gurobi does not support local cutting planes. As for SCIP, we interrupt the solving process after deriving (non-) robustness.

4.2. Experimental setup

All datasets are available in PyG and summarized in Table 1. The attack label for each graph/node is $(c + 1) \bmod C$, where c is the predictive label. For graph classification, we train a MPNN with 3 SAGEConv (Hamilton et al., 2017) layers with 16 hidden channels, followed by an add pooling and a dense layer as the final classifier. ReLU is used in the first 2 SAGEConv layers. Following Jin et al. (2020), 10% of the graphs are used to train the model. The local budget of each node is $q_v = \max\{0, d_v - \max_{u \in \mathcal{N}(v)} d_u + s\}$, where d_v is the degree of node v , s is the so-called local attack strength. In our experiments, we choose $s \in \{2, 3, 4\}$, and use percentage of the number of edges as the global budget g , where $g \in \{1, 2, 3, 4\}$. We report results for $s = 2$ in the paper and results for $s \in \{3, 4\}$ in Appendix C. For node classification, we train a MPNN with 2 SAGEConv layers with 32 hidden channels. Similar to Günnemann & Günnemann (2020), 10% of the nodes are used for training. We set 10 as the global budget and q_v as the local budget. For each node, we extract 2-hop neighborhood to build the corresponding verification problem. It is noteworthy that 2-hop neighborhood is sufficient for 2 message passing steps in MPNN, but insufficient for 2 graph convolutional steps in GCN. The reason is that removing an edge within a 3-hop neighborhood may influence the degree of a node within a 2-hop neighborhood.

We evaluate the performance of various verification methods on benchmarks including: (i) MUTAG and ENZYMES (Morris et al., 2020) for graph classification, and (ii) Cora and CiteSeer (Yang et al., 2023) for node classification.

All models are trained 200 epochs with learning rate 0.01, weight decay 10^{-4} , and dropout 0.5. We set 2 hours as the time limit for verifying a graph in graph classification, and 30 minutes for verifying a node in node classification.

4.3. Numerical results

For each verification problem, we will get one of three results: robust (objective has non-negative lower bound), nonrobust (a feasible attack, i.e., solution with negative objective, is found), or time out (inconclusive within time limit). For each benchmark, we consider three criteria for each method: (i) average solving time (avg-time), (ii) shifted geometric mean (sgm-time) of solving time, and (iii) number of solved instances within time limit. A commonly used measure to compare MIP-based methods, the shifted geometric mean of t_1, \dots, t_n is $(\prod_{i=1}^n (t_i + s))^{1/n} - s$, where shift $s = 10$. Since all approaches use the same model, we ignore model creation time (0:036s). We also exclude the negligible time (0:005s) spent computing variable bounds for both basic and sbt. For abt, we include the time calculating variable bounds since tightening happens at each branch-and-bound tree node. We cannot know the ground truth of a robust instance without complete enumeration or relying on a solver's numerical tolerances, so we classify an instance as robust if all methods claim it is robust except for time out. Three criteria are evaluated for each method on each benchmark over all instances and all robust instances, respectively. We exclude all instances with inconsistencies, i.e., SCIP and Gurobi declares differently, for a fair comparison. See the Appendix B for more details.

The results for node classification are reported in Table 3 in Appendix C. Only removing edges results in simple verification problems: all methods can solve all instances instantly. Adding more cutting planes is not helpful as this could hinder heuristics to find a feasible attack (in case of non-robustness) or result in solving more (difficult) LPs due to additional cutting planes (in case of robustness). Therefore, we skip the comparison with GRBbasic and GRBsbt.

For graph classification, we analyze results with local attack strengths = 2 in the main text and put results for s 2 f 3; 4g in Appendix C. Our numerical analysis for s 2 is consistent with the s 2 f 3; 4g results. Table 2 summarizes the results for all instances. Figure 2 visualizes the number of solved (robust) instances below different time costs, and compares the time costs between SCIPabt and SCIPsbt for ENZYMES. See Appendix C for similar plots for MUTAG. Figure 3 plots the number of different types of graphs (robust, nonrobust, time out) with various global budgets 10 for ENZYMES and MUTAG.

As shown in Table 2, SCIPsbt is around three times faster than SCIPbasic and solves more instances within the same time limit. From the comparison between GRBbasic and

GRBsbt, we can still notice the speed-up from static bounds tightening. Considering all instances from MUTAG, it seems like static bounds tightening slows down the solving process. The reason is that most MUTAG instances are not robust, i.e., finding good bounds on the objective value is unnecessary, finding a feasible attack instead is sufficient.

Our numerical results reflect similar performance between SCIPsbt and SCIPabt w.r.t. times in Table 2. SCIPabt might even be slower than SCIPsbt in some instances. On the one hand, we proposed abt as a general extension of sbt and expect it outperforms sbt in harder verification problems. One can easily create larger problems from different aspects, e.g., increasing budgets, incorporating feature perturbations, and enlarging size of models. However, larger problems do not imply harder verification problems since the instance could be nonrobust. Then the extra cutting planes added in abt may slow down finding a feasible attack. On the other hand, as shown in Appendix C, when comparing OBBT and SCIPsbt over robust instances, OBBT bounds are indeed tighter but still perform similarly to SCIPsbt. The phenomenon that tighter bounds can result in slower solving times has been previously reported (Badilla et al., 2023).

Observe in Figure 2 that, of the 3549 robust instances from the ENZYMES benchmark, 412 are significantly faster when using SCIPabt and 512 are significantly faster when using SCIPsbt. Similarly for the MUTAG benchmark, 19 of the 44 robust instances are substantially faster using SCIPabt rather than SCIPsbt (see Figure 7). This is also reflected in Table 2 when considering the robust instances only. For the MUTAG instances (which are harder to solve than ENZYMES), SCIPabt is roughly 10% faster than SCIPsbt in shifted geometric mean (29% in arithmetic mean). This effect is even more pronounced for the most difficult MUTAG instances with a global budget of 2% and 3%, where SCIPabt is 10.8% and 22.0%, respectively, faster than SCIPsbt in shifted geometric mean (29.4% and 35.8% in arithmetic mean), see Table 9 in Appendix C.

We therefore propose running SCIPsbt and SCIPabt in parallel, this idea corresponds to the common observation in MIP that parallelizing multiple strategies (here: SCIPsbt and SCIPabt) yields more benefits than parallelizing just one algorithm (Carvajal et al., 2014).

5. Conclusion

We propose topology-based bounds tightening approaches to verify message-passing neural networks. By exploiting graph structures and available budgets, our techniques compute tighter bounds for variables and thereby help certify robustness. Numerical results show the improvement of topology-based bounds tightening w.r.t. the solving time and the number of solved instances.

Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/W003317/1], an Imperial College Hans Rausing PhD Scholarship to SZ, and a BASF/RAEng Research Chair in Data-Driven Optimisation to RM. This collaboration was initiated at the Mittag-Leffler Institute seminar titled Learning from Both Sides: Linear and Nonlinear Mixed-Integer Optimization

Impact Statement

Despite the widespread use and outstanding performance of GNNs in various fields, their vulnerabilities are frequently detected, even with slight perturbations. In safety-critical applications such as drug design and autonomous driving, such vulnerabilities could result in unacceptable consequences. For safety considerations, two major directions are investigated by many researchers: (i) verification: how to measure the robustness of GNNs, and (ii) robust training: how to train GNNs that are more robust. This work considers verification problems on MPNNs, a classic GNN framework whose robustness is seldom studied in the literature. With the proposed bounds tightening strategies, we hope that the robustness of MPNNs deployed in real-world applications could be verified efficiently to avoid unreliable predictions.

References

- Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., and Vielma, J. P. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming* 183(1-2):3–39, 2020.
- Badilla, F., Goycoolea, M., Muñoz, G., and Serra, T. Computational tradeoffs of optimization-based bound tightening in ReLU networks, 2023.
- Belotti, P., Bonami, P., Fischetti, M., Lodi, A., Monaci, M., Nogales-Gomez, A., and Salvagnin, D. On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications* 65:545–566, 2016.
- Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., et al. Enabling research through the SCIP optimization suite 8. *ACM Transactions on Mathematical Software* 49(2):1–21, 2023.
- Bojchevski, A. and Günnemann, S. Certified robustness to graph perturbations. *NeurIPS* 2019.
- Bojchevski, A., Gasteiger, J., and Günnemann, S. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. *ICML*, 2020.
- Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., and Misener, R. Efficient verification of ReLU-based neural networks via dependency analysis. *AAAI*, 2020.
- Carvajal, R., Ahmed, S., Nemhauser, G., Furman, K., Goel, V., and Shao, Y. Using diversification, communication and parallelism to solve mixed-integer linear programs. *Operations Research Letters* 42(2):186–189, 2014.
- Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C. D., and Misener, R. OMLT: Optimization & machine learning toolkit. *Journal of Machine Learning Research* 23(349):1–8, 2022.
- Chen, J., Xu, H., Wang, J., Xuan, Q., and Zhang, X. Adversarial detection on graph structured data. *PRMLP*, 2020.
- Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. *ICML*, 2019.
- Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. Adversarial attack on graph structured data. *ICML*, 2018.
- Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *CLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- Fischetti, M. and Jo, J. Deep neural networks and mixed integer linear optimization. *Constraints* 23(3):296–309, 2018.
- Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.-K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., et al. The SCIP optimization suite 7.0. Technical Report 20-10, ZIB, Takustr. 7, 14195 Berlin, 2020.
- Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *CLR*, 2019.
- Geisler, S., Schmidt, T., Sirin, H., Zügner, D., Bojchevski, A., and Günnemann, S. Robustness of graph neural networks at scale. *NeurIPS* 2021.
- Günnemann, S. Graph neural networks: Adversarial robustness. *Graph Neural Networks: Foundations, Frontiers, and Applications* pp. 149–176, 2022.
- Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL <https://www.gurobi.com>.
- Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *NeurIPS* 2017.

- Hojny, C. and Zhang, S. SCIP-MPNN: Code for the paper “Verifying message-passing neural networks via topology-based bounds tightening” <https://doi.org/10.5281/zenodo.11208355>, 2024.
- Jin, H., Shi, Z., Peruri, V. J. S. A., and Zhang, X. Certified robustness of graph convolution networks for graph classification under topological attacks. *NeurIPS* 2020.
- Jin, H., Yu, Z., and Zhang, X. Certifying robust graph classification under orthogonal Gromov-Wasserstein threats. *NeurIPS* 2022.
- Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.
- Lomuscio, A. and Maganti, L. An approach to reachability analysis for feed-forward ReLU neural networks. *arXiv preprint arXiv:1706.07351*, 2017.
- Ma, J., Ding, S., and Mei, Q. Towards more practical adversarial attacks on graph neural networks. *NeurIPS* 2020.
- McDonald, T., Tsay, C., Schweidtmann, A. M., and Yorke-Smith, N. Mixed-integer optimisation of graph neural networks for computer-aided molecular design. *Computers & Chemical Engineering* 185:108660, 2024.
- Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)* 2020.
- Müller, M. N., Makarchuk, G., Singh, G., Pschel, M., and Vechev, M. PRIMA: general and precise neural network certification via scalable convex hull approximations. *Proceedings of the ACM on Programming Languages* 6(POPL):1–33, 2022.
- Sälzer, M. and Lange, M. Fundamental limits in formal verification of message-passing neural networks. *ICLR*, 2023.
- Singh, G., Gehr, T., Mirman, M., Pschel, M., and Vechev, M. Fast and effective robustness certification. *NeurIPS* 2018.
- Singh, G., Ganvir, R., Pschel, M., and Vechev, M. Beyond the single neuron convex barrier for neural network certification. In *NeurIPS* pp. 15098–15109, 2019a.
- Singh, G., Gehr, T., Pschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages* 3(POPL):1–30, 2019b.
- Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., and Honavar, V. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. *IRWWW* 2020.
- Takahashi, T. Indirect adversarial attacks via poisoning neighbors for graph convolutional networks. *IEEE Big Data*, 2019.
- Tjandraatmadja, C., Anderson, R., Huchette, J., Ma, W., PATEL, K. K., and Vielma, J. P. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. In *NeurIPS* volume 33, pp. 21675–21686, 2020.
- Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. In *ICLR*, 2019.
- Tsay, C., Kronqvist, J., Thebelt, A., and Misener, R. Partition-based formulations for mixed-integer optimization of trained ReLU neural networks. *NeurIPS* 2021.
- Turner, M., Chmiela, A., Koch, T., and Winkler, M. PySCIPOpt-ML: Embedding trained machine learning models into mixed-integer programs. *arXiv preprint arXiv:2312.08074* 2023.
- Wang, B., Jia, J., Cao, X., and Gong, N. Z. Certified robustness of graph neural networks against adversarial structural perturbation. *SIGKDD*, 2021.
- Wang, J., Luo, M., Suya, F., Li, J., Yang, Z., and Zheng, Q. Scalable attack on graph data by injecting vicious nodes. *Data Mining and Knowledge Discovery* 34:1363–1389, 2020.
- Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. Towards fast computation of certified robustness for ReLU networks. In *ICML*, 2018.
- Wu, H., Barrett, C., Sharif, M., Narodytska, N., and Singh, G. Scalable verification of GNN-based job schedulers. *Proceedings of the ACM on Programming Languages* 6(OOPSLA2):1036–1065, 2022.
- Xia, Z., Yang, H., Wang, B., and Jia, J. GNNCert: Deterministic certification of graph neural networks against adversarial perturbations. *ICLR*, 2024.
- Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X. Topology attack and defense for graph neural networks: An optimization perspective. *ICAI*, 2019.
- Yang, R., Shi, J., Xiao, X., Yang, Y., Bhowmick, S. S., and Liu, J. PANE: scalable and effective attributed network embedding. *The VLDB Journal* pp. 1–26, 2023.

- Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. *NeurIPS*, 2018.
- Zhang, S., Campos, J. S., Feldmann, C., Walz, D., Sandfort, F., Mathea, M., Tsay, C., and Misener, R. Optimizing over trained GNNs via symmetry breaking. In *NeurIPS*, 2023.
- Zhang, S., Campos, J. S., Feldmann, C., Sandfort, F., Mathea, M., and Misener, R. Augmenting optimization-based molecular design with graph neural networks. *Computers & Chemical Engineering*, 186:108684, 2024.
- Zhao, H., Hijazi, H., Jones, H., Moore, J., Tanneau, M., and Van Hentenryck, P. Bound tightening using rolling-horizon decomposition for neural network verification. *arXiv preprint arXiv:2401.05280*, 2024.
- Zügner, D. and Günnemann, S. Certifiable robustness and robust training for graph convolutional networks. In *SIGKDD*, 2019a.
- Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019b.
- Zügner, D. and Günnemann, S. Certifiable robustness of graph convolutional networks under structure perturbations. In *SIGKDD*, 2020.
- Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *SIGKDD*, 2018.

A. Proofs of Properties 3.2 & 3.4

Proof of Property 3.2. Since $A_{u,v} = 0$, we only need to consider the lower bound of $\prod_{f \in F_1} W_{f,f^0} X_{u,f}$ in Eq. (16), which is denoted as $lb_{u|v}$. Using bounds for $X_{u,f}$ gives $lb_{u|v}$ defined as Eq. (18). Then Eq. (16) becomes:

$$lb(X_{V;f^0}) = \min_{A \in P_1(A)} \prod_{u \in V} A_{u,v} lb_{u|v} + b_{V;f^0}. \quad (24)$$

Recall the definition of $P_1(A)$, we can remove/add at most q_v neighbors of node v . Denote the set of removed/added neighbors as V_{lb} . Then the summation $u \in V$ can be divided into: $u \in N(v)$ (original neighbors), $u \in V_{lb} \cap N(v)$ (added neighbors), $u \in N(v) \setminus V_{lb}$ (removed neighbors), and $u \in V \cap (N(v) \setminus V_{lb})$ (nodes without contribution either before or now), from which we obtain that:

$$\begin{aligned} \prod_{u \in V} A_{u,v} lb_{u|v} &= \prod_{u \in N(v)} A_{u,v} lb_{u|v} + \prod_{u \in V_{lb} \cap N(v)} A_{u,v} lb_{u|v} + \prod_{u \in N(v) \setminus V_{lb}} A_{u,v} lb_{u|v} + \prod_{u \in V \cap (N(v) \setminus V_{lb})} A_{u,v} lb_{u|v} \\ &= \prod_{u \in N(v)} lb_{u|v} + \prod_{u \in V_{lb}} lb_{u|v} \end{aligned} \quad (25)$$

with $lb_{u|v}$ defined as Eq. (19). Moving the minimization to the only changeable term $\prod_{u \in V_{lb}} lb_{u|v}$ yields Eq. (17). \square

Proof of Property 3.4. This property can be proved similarly to Property 3.2. Here we give a simple way to prove it using the conclusion of Property 3.2. As mentioned in Remark 3.5, we can modify the original graph with previous decisions, i.e., current neighborhood of node v is $(N(v) \cap V_0) \setminus V_1$ and the remaining budget is q_v^0 defined as Eq. (23). Since we already decided the values of $A_{u,v}$ for $u \in V_0 \setminus V_1$, V_{lb} should not include any node in $V_0 \setminus V_1$. Applying Property 3.2 on the current graph with budgets q_v^0 gives Eq. (22) and finishes this proof. \square

B. Implementation details

B.1. Local cutting planes

A local cutting plane in SCIP is any inequality together with the first node where it is valid. As shown in the right side of Figure 1, after branching variable $A_{1,0} = 0$, we can improve the bounds of node 0 from $[4;9]$ to $[3;6]$. These improved bounds can be incorporated into the model by adding inequalities $x \leq 3$ and $x \geq 6$ to the left branch. However, these inequalities cannot be added to the root node as they are invalid unless $A_{1,0} = 0$. At each node in the branch-and-bound tree, all local cutting planes associated with its ancestors are valid. Including all of these local cutting planes is correct but inefficient. Therefore, SCIP will decide whether an inequality will be added. For example, if node 0 equals to 7 at the left child of the root node, then SCIP only adds $x \geq 6$ into the model since this inequality is violated.

B.2. Inconsistent instances

We exclude instances with inconsistent results from different solvers, which results in the numbers in column “#” of Table 6 – Table 11 being smaller than the total number of graphs from each benchmark. This is because we found and reported a small bug in Gurobi wherein several instances were declared *infeasible* despite having a feasible solution found by SCIP. This bug was easily fixed in Gurobi and is now incorporated into a recent minor release. We continued using the Gurobi version with the bug after the ICML rebuttal period because there was insufficient time to redo all the experiments. So, in fairness to the Gurobi solver, we slightly modified the optimization problems by increasing the right-hand side of Eq. (7c) by 10^{-11} . This change eliminated the bug, but then meant that the Gurobi solver frequently returns solutions that are infeasible, i.e., violating some constraints of the MIP model by more than tolerance 10^{-6} . To compare on an equal footing, we only report results on instances where all solvers return consistent answers. In other words, if one solver wrongly declares an instance *infeasible* or returns an infeasible solution, we do not report results.

C. Full numerical results

Table 3 gives results for node classification. Tables 4 and 5 summarize the results for graph classification with local attack strength $s \in \{2, 3, 4\}$ on ENZYMES and MUTAG, respectively. Detailed results with different global budgets are reported in Tables 6–11. Figures 6 and 7 plot the number of different types of graphs (robust, nonrobust, time out) with various budgets. Figures 4 and 5 visualize the number of solved (robust) instances below different time costs, and compares the time costs between SCIPabt and SCIPsbt. For graph classification, our numerical analysis in Section 4.3 for $s = 2$ is consistent with results for $s \in \{3, 4\}$, as shown here.

Furthermore, we report some numerical results to show that applying OBBT in GNN verification is not recommended. Our OBBT implementation is straight-forward: for each variable, change the objective in our verification problem to its lower/upper bound and relax all binary variables into continuous variables. Due to the high time cost for solving each instance, as well as our numerical observation that tighter bounds may slow down the process to find a feasible attack for a non-robust instance, we only apply OBBT to the 44 robust instances for MUTAG with $s = 2$. The average time spent on calculating OBBT bounds for all variables is 6045.02s (3.57s per LP), and the average solving time with OBBT bounds is 331.60s. As shown in Table 2, the average solving time for SCIPsbt is 336.41s. Therefore, the improvement of OBBT w.r.t. the solving time is quite limited, despite its high time cost on computing bounds. We also compare the bounds derived from *sbt* and OBBT using the relative bound tightness (RBT) used in Badilla et al. (2023): $RBT = \frac{SBT - OBBT}{OBBT + 10^{-10}}$, where SBT/OBBT represents the length of interval bounds derived from *sbt*/OBBT, respectively. For each MPNN layer, we average RBT over all variables in this layer for all 44 instances. The resulting RBT values are 0.07; 0.22; 0.57, which means that the *sbt* bounds are quite closed to OBBT bounds for early layers and decently tight for deeper layers.

Table 3: Summary of results for node classification. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), and SCIP aggressive bounds tightening (SCIPabt). For each global budget, the number of instances is 2708 for Cora and 3312 for CiteSeer. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 30 minutes (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

benchmark	method	all instances				robust instances			
		#	avg-time(s)	sgm-time(s)	# solved	#	avg-time(s)	sgm-time(s)	# solved
Cora	SCIPbasic	2708	0.10	0.10	2708	2223	0.08	0.08	2223
	SCIPsbt	2708	0.17	0.16	2708	2223	0.10	0.10	2223
	SCIPabt	2708	0.46	0.38	2708	2223	0.16	0.14	2223
CiteSeer	SCIPbasic	3312	0.07	0.06	3312	2917	0.06	0.06	2917
	SCIPsbt	3312	0.07	0.07	3312	2917	0.06	0.06	2917
	SCIPabt	3312	0.31	0.17	3312	2917	0.12	0.10	2917

Table 4: Summary of results for graph classification on ENZYMES with local attack strength $s \in \{2, 3, 4\}$. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 600, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

	method	all instances			robust instances				
		#	avg-time(s)	sgm-time(s)	# solved	#	avg-time(s)	sgm-time(s)	# solved
$s = 2$	SCIPbasic	5915	605.97	37.81	5579	3549	278.58	12.51	3444
	SCIPsbt	5915	230.59	21.26	5831	3549	82.89	6.65	3528
	SCIPabt	5915	246.02	21.77	5817	3549	88.95	6.71	3522
	GRBbasic	5915	86.03	7.59	5892	3549	32.80	2.82	3542
	GRBsbt	5915	74.87	7.09	5895	3549	22.90	2.50	3548
$s = 3$	SCIPbasic	5855	2628.71	460.25	4149	1554	1423.02	99.43	1308
	SCIPsbt	5855	1533.12	214.04	4977	1554	750.70	48.68	1440
	SCIPabt	5855	1583.67	224.59	4943	1554	738.67	48.11	1447
	GRBbasic	5855	845.77	85.82	5549	1554	367.89	22.54	1530
	GRBsbt	5855	738.26	74.83	5524	1554	238.48	18.89	1545
$s = 4$	SCIPbasic	5901	4163.96	1492.61	3003	577	2030.34	319.19	448
	SCIPsbt	5901	2802.04	581.20	4012	577	1353.31	154.83	504
	SCIPabt	5901	2841.31	602.57	4013	577	1318.26	152.16	509
	GRBbasic	5901	1780.68	306.11	5125	577	862.25	62.75	547
	GRBsbt	5901	1372.88	233.52	5290	577	589.89	49.66	570

Table 5: Summary of results for graph classification on MUTAG with local attack strength $s \in \{2, 3, 4\}$. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 188, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

	method	all instances				robust instances			
		#	avg-time(s)	sgm-time(s)	# solved	#	avg-time(s)	sgm-time(s)	# solved
$s = 2$	SCIPbasic	1589	679.86	189.75	1575	44	798.47	202.93	40
	SCIPsbt	1589	196.07	75.17	1589	44	336.41	100.86	44
	SCIPabt	1589	207.50	82.43	1589	44	238.10	91.11	44
	GRBbasic	1589	34.58	4.06	1589	44	162.25	12.11	44
	GRBsbt	1589	59.93	22.40	1589	44	73.78	15.00	44
	SCIPbasic	1840	1355.17	262.63	1670	68	643.89	427.71	66
$s = 3$	SCIPsbt	1840	720.13	174.32	1793	68	380.43	179.95	66
	SCIPabt	1840	669.39	169.55	1797	68	373.90	172.83	66
	GRBbasic	1840	601.86	30.92	1816	68	208.24	15.38	68
	GRBsbt	1840	315.93	70.03	1834	68	234.78	27.21	66
	SCIPbasic	1844	1745.56	487.75	1644	64	386.72	340.95	64
	SCIPsbt	1844	1039.90	180.30	1734	64	155.26	144.16	64
$s = 4$	SCIPabt	1844	1004.92	177.84	1748	64	147.46	136.26	64
	GRBbasic	1844	1113.20	58.17	1758	64	11.90	10.96	64
	GRBsbt	1844	420.78	84.82	1837	64	23.04	21.29	64

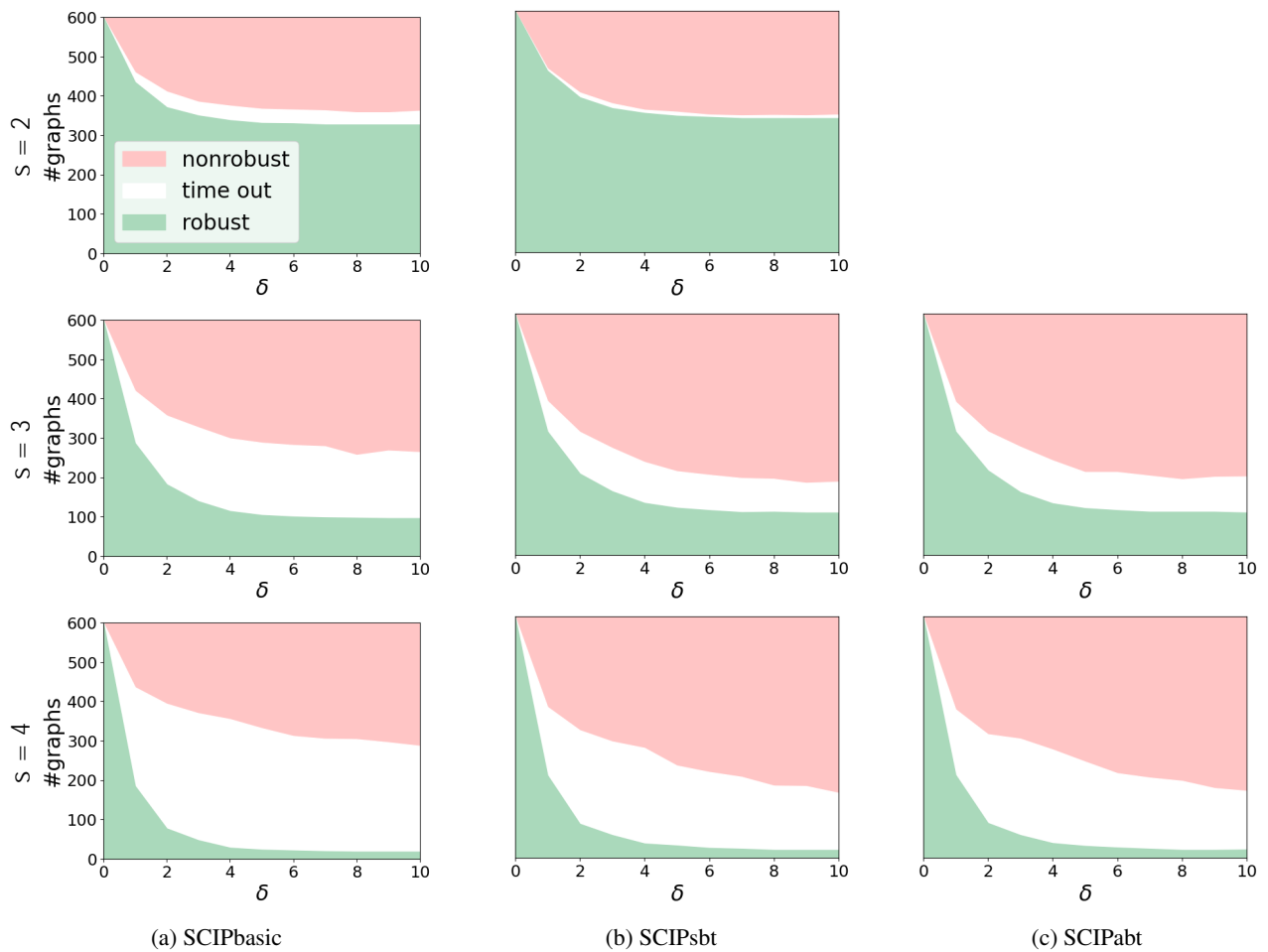


Figure 4: For each SCIP-based method on ENZYMES with local attack strength $s \in \{2, 3, 4\}$, we count the number of robust graphs (green), nonrobust graphs (red), and time out (white). The percentage of the number of edges is the global budget.

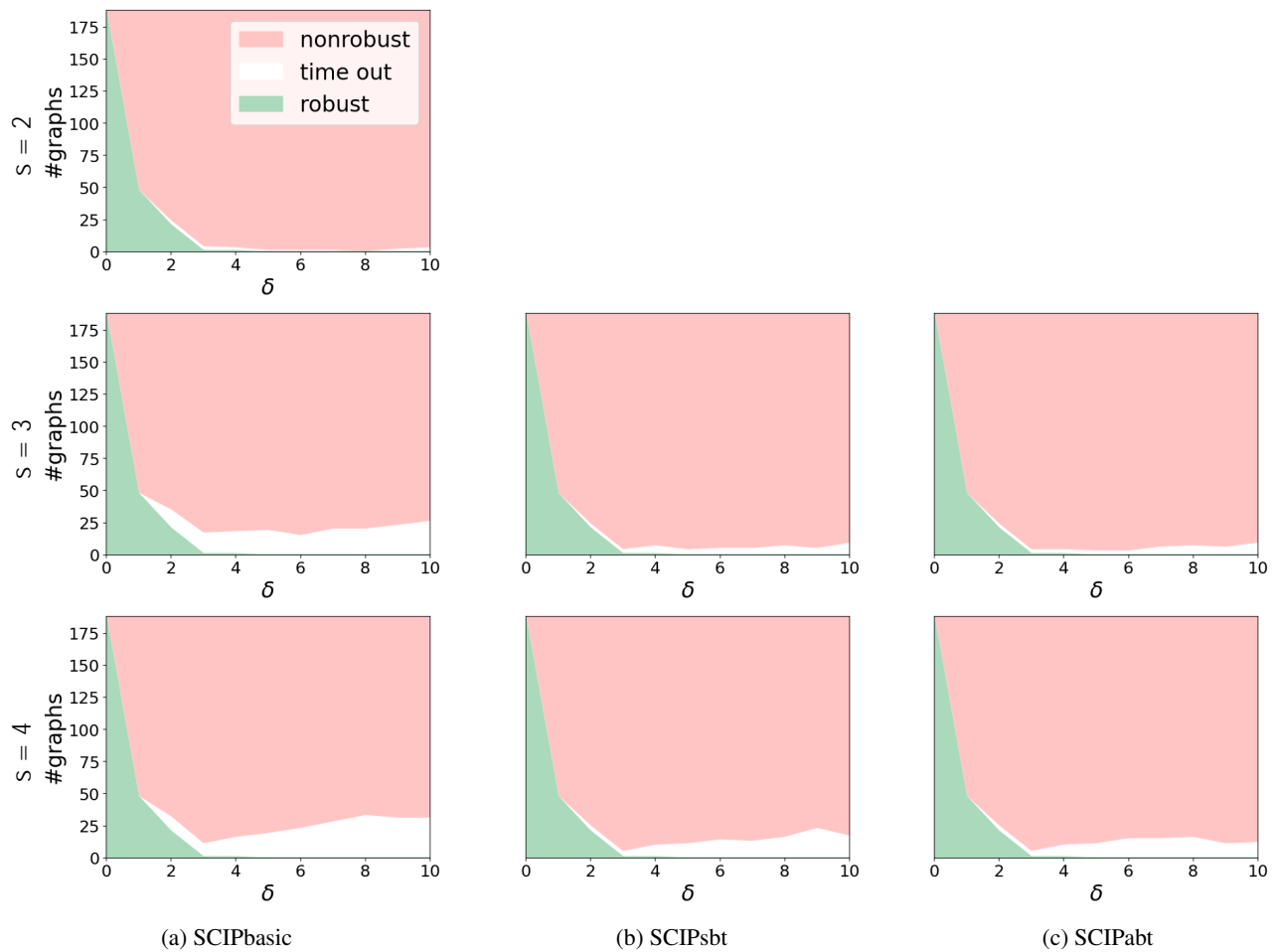


Figure 5: For each SCIP-based method on MUTAG with local attack strength $s \in \{2, 3, 4\}$, we count the number of robust graphs (green), nonrobust graphs (red), and time out (white). The percentage of the number of edges is the global budget.

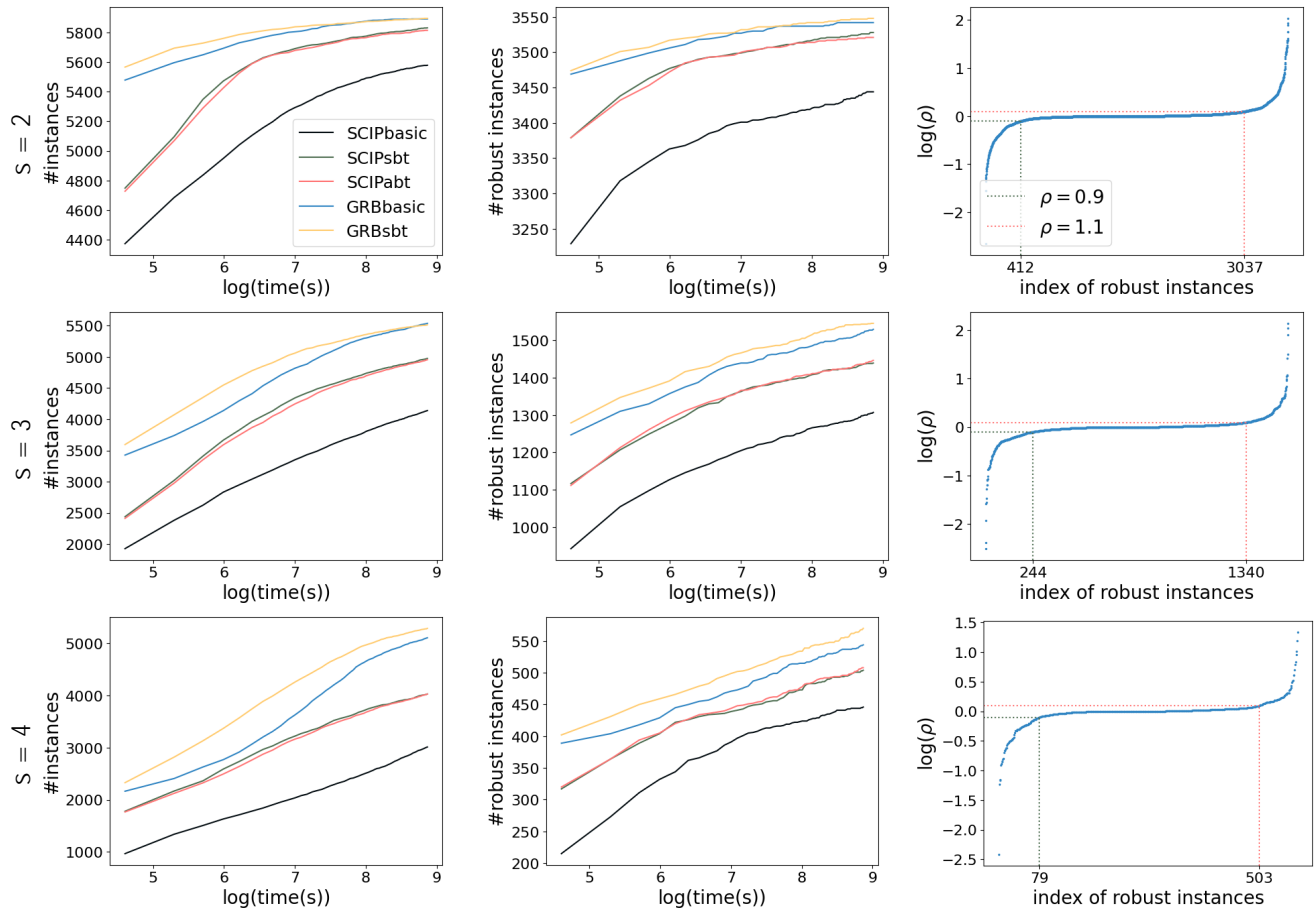


Figure 6: ENZYMES benchmark with local attack strength $s \in \{2, 3, 4\}$. **(left)** Number of instances solved by each method below different time costs. **(middle)** Number of robust instances solved by each method below different time costs. **(right)** Consider ρ , the ratio of time cost between SCIPabt and SCIPsbt on each robust instance.

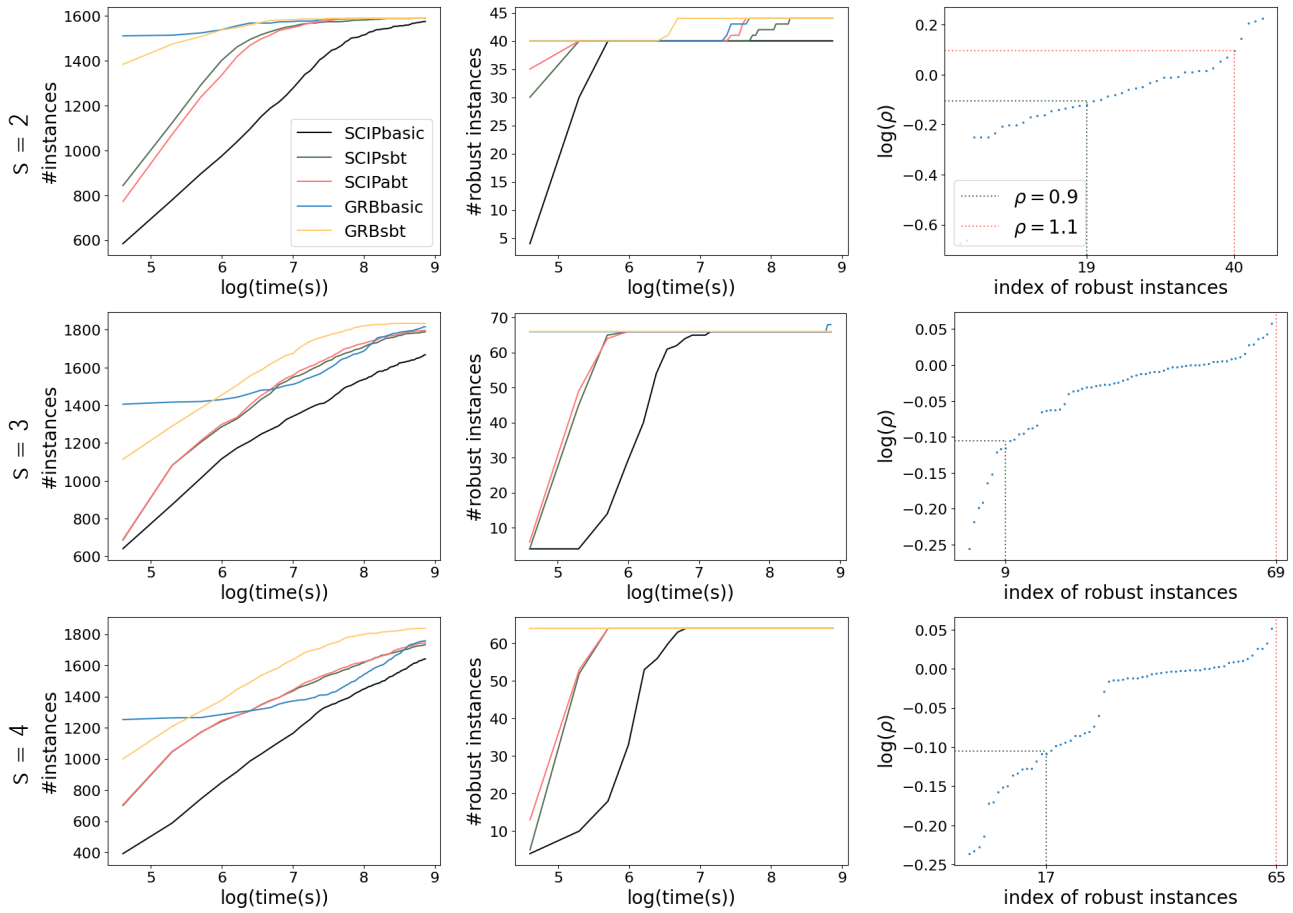


Figure 7: MUTAG benchmark with local attack strength $s \in \{2, 3, 4\}$. **(left)** Number of instances solved by each method below different time costs. **(middle)** Number of robust instances solved by each method below different time costs. **(right)** Consider ρ , the ratio of time cost between SCIPabt and SCIPsbt on each robust instance.

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

Table 6: Results for ENZYMES with local attack strength $s = 2$ and different global budgets. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 600, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

method	all instances				robust instances			
	#	avg-time	sgm-time	# solved	#	avg-time	sgm-time	# solved
global budget: 0:01 <i>jEje</i>								
SCIPbasic	600	465.47	25.13	575	453	420.33	18.76	435
SCIPsbt	600	193.02	14.46	594	453	166.90	10.52	451
SCIPabt	600	190.05	14.58	592	453	161.85	10.53	449
GRBbasic	600	119.88	7.08	594	453	84.71	5.76	451
GRBsbt	600	87.31	5.90	596	453	47.40	4.89	453
global budget: 0:02 <i>jEje</i>								
SCIPbasic	597	630.47	37.42	557	392	457.99	18.92	371
SCIPsbt	597	248.67	20.65	585	392	181.41	10.44	385
SCIPabt	597	260.41	21.16	584	392	194.53	10.58	385
GRBbasic	597	170.91	9.36	588	392	111.73	5.05	388
GRBsbt	597	111.66	7.59	592	392	66.45	4.33	391
global budget: 0:03 <i>jEje</i>								
SCIPbasic	591	628.77	37.75	556	359	328.18	14.17	347
SCIPsbt	591	252.04	20.87	579	359	117.80	7.67	355
SCIPabt	591	262.35	21.03	577	359	120.84	7.67	354
GRBbasic	591	108.75	8.55	587	359	45.55	3.34	358
GRBsbt	591	79.77	7.05	588	359	22.37	2.79	359
global budget: 0:04 <i>jEje</i>								
SCIPbasic	593	633.12	39.29	557	346	256.36	11.96	336
SCIPsbt	593	234.00	21.84	585	346	61.20	6.07	344
SCIPabt	593	253.13	22.90	583	346	62.85	6.13	345
GRBbasic	593	80.87	7.59	591	346	6.44	2.19	346
GRBsbt	593	61.60	6.79	591	346	5.48	1.98	346
global budget: 0:05 <i>jEje</i>								
SCIPbasic	591	647.96	41.10	555	338	234.96	10.74	329
SCIPsbt	591	241.04	22.37	581	338	48.19	5.58	337
SCIPabt	591	293.33	23.46	576	338	75.63	5.66	335
GRBbasic	591	72.31	7.61	591	338	6.47	1.99	338
GRBsbt	591	70.81	7.34	589	338	13.86	1.83	338
global budget: 0:06 <i>jEje</i>								
SCIPbasic	594	623.67	41.01	560	334	204.73	10.09	327
SCIPsbt	594	210.26	22.99	588	334	37.46	5.22	333
SCIPabt	594	240.04	23.29	585	334	55.18	5.36	332
GRBbasic	594	73.46	7.56	594	334	8.41	1.85	334
GRBsbt	594	57.18	7.04	594	334	22.34	1.72	334
global budget: 0:07 <i>jEje</i>								
SCIPbasic	590	635.63	40.99	554	332	195.72	9.84	325
SCIPsbt	590	221.80	23.07	583	332	37.65	5.08	331
SCIPabt	590	259.83	23.68	583	332	54.13	5.25	330
GRBbasic	590	53.62	7.07	590	332	8.03	1.78	332
GRBsbt	590	62.36	7.46	590	332	6.69	1.77	332
global budget: 0:08 <i>jEje</i>								
SCIPbasic	583	608.75	39.97	553	332	197.92	9.95	325
SCIPsbt	583	240.35	22.52	575	332	38.82	5.11	331
SCIPabt	583	214.77	22.42	577	332	38.26	5.11	331
GRBbasic	583	60.73	6.92	582	332	8.32	1.85	332
GRBsbt	583	76.61	7.26	582	332	6.03	1.64	332
global budget: 0:09 <i>jEje</i>								
SCIPbasic	588	585.39	38.98	557	331	197.91	10.02	324
SCIPsbt	588	241.76	23.02	581	331	43.36	5.08	330
SCIPabt	588	249.56	23.67	580	331	37.99	5.05	330
GRBbasic	588	72.99	7.50	587	331	8.08	1.85	331
GRBsbt	588	75.35	7.23	586	331	6.08	1.66	331
global budget: 0:10 <i>jEje</i>								
SCIPbasic	588	602.12	39.14	555	332	205.53	10.01	325
SCIPsbt	588	223.54	22.05	580	332	46.58	5.09	331
SCIPabt	588	237.00	22.88	580	332	41.39	5.07	331
GRBbasic	588	44.81	6.83	588	332	7.72	1.96	332
GRBsbt	588	65.58	7.31	587	332	16.41	1.78	332

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

Table 7: Results for ENZYMES with local attack strength $s = 3$ and different global budgets. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 600, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

method	all instances				robust instances			
	#	avg-time	sgm-time	# solved	#	avg-time	sgm-time	# solved
global budget: 00:01 <i>jEje</i>								
SCIPbasic	597	2041.24	248.53	464	319	1088.49	94.41	287
SCIPsbt	597	1160.38	109.23	521	319	481.74	43.92	307
SCIPabt	597	1129.15	109.77	523	319	478.08	43.68	307
GRBbasic	597	985.15	52.53	542	319	370.09	20.68	310
GRBsbt	597	700.88	39.96	558	319	193.68	16.28	318
global budget: 00:02 <i>jEje</i>								
SCIPbasic	587	2738.05	498.94	413	218	1896.50	185.55	179
SCIPsbt	587	1683.98	226.92	484	218	1004.56	88.59	199
SCIPabt	587	1623.45	227.82	491	218	970.46	86.96	207
GRBbasic	587	1059.95	99.76	544	218	495.46	43.64	215
GRBsbt	587	728.29	66.57	552	218	352.89	34.88	217
global budget: 00:03 <i>jEje</i>								
SCIPbasic	590	2835.26	542.92	403	169	1685.32	142.10	139
SCIPsbt	590	1745.83	245.95	482	169	883.20	67.97	157
SCIPabt	590	1757.43	252.98	477	169	857.83	66.17	155
GRBbasic	590	926.16	94.63	551	169	346.30	30.00	166
GRBsbt	590	597.79	64.60	565	169	266.98	25.74	168
global budget: 00:04 <i>jEje</i>								
SCIPbasic	588	2793.58	539.45	404	141	1623.34	113.29	114
SCIPsbt	588	1684.95	241.19	486	141	844.50	54.50	129
SCIPabt	588	1779.51	259.53	481	141	869.72	53.91	128
GRBbasic	588	705.60	84.77	568	141	302.69	23.50	141
GRBsbt	588	498.54	63.82	572	141	268.36	20.55	139
global budget: 00:05 <i>jEje</i>								
SCIPbasic	589	2730.97	502.64	407	128	1462.98	88.07	104
SCIPsbt	589	1559.45	230.58	499	128	825.88	44.16	117
SCIPabt	589	1588.79	241.82	500	128	821.32	43.98	116
GRBbasic	589	653.18	82.60	572	128	363.43	20.28	128
GRBsbt	589	597.25	69.81	571	128	282.16	17.86	126
global budget: 00:06 <i>jEje</i>								
SCIPbasic	587	2678.33	483.01	409	120	1405.21	82.54	99
SCIPsbt	587	1523.67	226.15	502	120	798.68	41.36	110
SCIPabt	587	1611.51	243.19	496	120	813.46	42.08	110
GRBbasic	587	719.60	81.19	564	120	415.24	19.88	118
GRBsbt	587	664.13	78.00	558	120	214.15	17.07	120
global budget: 00:07 <i>jEje</i>								
SCIPbasic	576	2729.50	492.39	400	115	1290.92	70.80	97
SCIPsbt	576	1518.06	226.43	493	115	699.68	35.11	105
SCIPabt	576	1605.08	240.04	489	115	682.72	34.36	106
GRBbasic	576	762.36	88.48	552	115	332.71	16.17	114
GRBsbt	576	780.03	86.20	541	115	209.06	13.87	114
global budget: 00:08 <i>jEje</i>								
SCIPbasic	579	2527.15	450.55	425	116	1288.86	70.91	97
SCIPsbt	579	1445.35	212.99	499	116	675.81	35.33	107
SCIPabt	579	1492.87	226.64	500	116	700.39	36.03	107
GRBbasic	579	828.08	90.22	552	116	307.12	16.05	114
GRBsbt	579	842.19	93.20	539	116	146.60	13.40	116
global budget: 00:09 <i>jEje</i>								
SCIPbasic	586	2594.38	476.06	415	113	1190.02	64.52	96
SCIPsbt	586	1498.64	232.34	512	113	701.85	33.91	104
SCIPabt	586	1634.99	246.41	499	113	620.21	32.32	106
GRBbasic	586	863.88	96.14	561	113	260.83	14.23	112
GRBsbt	586	925.87	99.95	539	113	178.31	12.64	113
global budget: 00:10 <i>jEje</i>								
SCIPbasic	576	2627.36	459.95	409	115	1292.84	69.60	96
SCIPsbt	576	1513.94	234.73	499	115	746.69	35.91	105
SCIPabt	576	1620.34	252.07	487	115	727.18	35.21	105
GRBbasic	576	952.10	99.60	543	115	388.85	16.03	112
GRBsbt	576	1058.39	112.22	529	115	225.30	13.55	114

Table 8: Results for ENZYMES with local attack strength $s = 4$ and different global budgets. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 600, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

method	all instances				robust instances			
	#	avg-time	sgm-time	# solved	#	avg-time	sgm-time	# solved
global budget: $\alpha 0:01$ $jEje$								
SCIPbasic	594	3421.76	773.05	345	211	1260.49	170.39	182
SCIPsbt	594	2352.82	345.23	425	211	593.60	74.72	202
SCIPabt	594	2282.73	337.01	432	211	569.32	74.28	203
GRBbasic	594	2107.45	170.88	456	211	470.19	24.40	204
GRBsbt	594	1595.10	130.26	493	211	260.64	20.56	211
global budget: $\alpha 0:02$ $jEje$								
SCIPbasic	593	4550.98	1874.90	280	103	2833.99	581.49	76
SCIPsbt	593	3259.56	770.71	362	103	2135.50	324.52	82
SCIPabt	593	3120.26	741.51	374	103	2049.08	316.62	84
GRBbasic	593	2373.24	422.28	463	103	1195.25	153.67	98
GRBsbt	593	1658.01	233.42	502	103	819.20	108.68	102
global budget: $\alpha 0:03$ $jEje$								
SCIPbasic	591	4476.55	1883.37	270	65	2733.32	583.01	45
SCIPsbt	591	3307.01	869.32	360	65	1900.42	315.89	54
SCIPabt	591	3340.71	888.09	353	65	1820.57	298.76	54
GRBbasic	591	1984.46	403.78	504	65	1329.91	164.43	60
GRBsbt	591	1283.41	191.51	531	65	993.78	123.35	63
global budget: $\alpha 0:04$ $jEje$								
SCIPbasic	593	4469.33	1890.99	268	46	3027.57	602.22	28
SCIPsbt	593	3339.44	808.49	357	46	2376.84	332.05	35
SCIPabt	593	3312.35	824.98	362	46	2295.64	327.83	36
GRBbasic	593	1643.02	335.49	537	46	1407.86	154.94	41
GRBsbt	593	1039.69	175.10	550	46	1024.66	106.28	44
global budget: $\alpha 0:05$ $jEje$								
SCIPbasic	595	4379.75	1691.84	286	34	2571.61	438.08	23
SCIPsbt	595	2956.30	659.83	396	34	1625.51	205.03	30
SCIPabt	595	3034.27	691.12	385	34	1651.50	201.25	29
GRBbasic	595	1538.31	300.14	539	34	843.47	86.46	32
GRBsbt	595	1035.63	198.01	561	34	696.91	67.81	34
global budget: $\alpha 0:06$ $jEje$								
SCIPbasic	588	4108.15	1427.47	302	29	2220.69	366.12	21
SCIPsbt	588	2879.51	624.42	399	29	1571.68	174.19	24
SCIPabt	588	2891.41	644.29	404	29	1619.69	174.26	25
GRBbasic	588	1472.87	283.85	543	29	1209.09	79.81	27
GRBsbt	588	1035.34	214.00	551	29	518.49	49.90	29
global budget: $\alpha 0:07$ $jEje$								
SCIPbasic	592	4147.98	1484.03	309	24	1749.90	288.38	19
SCIPsbt	592	2685.22	543.80	414	24	1005.48	122.89	22
SCIPabt	592	2732.23	556.59	416	24	1053.82	119.60	22
GRBbasic	592	1609.26	294.58	532	24	679.49	41.95	24
GRBsbt	592	1298.98	257.15	541	24	155.27	32.76	24
global budget: $\alpha 0:08$ $jEje$								
SCIPbasic	588	4085.50	1440.95	308	22	1622.94	281.69	18
SCIPsbt	588	2489.10	505.68	430	22	1091.16	114.08	19
SCIPabt	588	2693.02	563.16	418	22	1089.48	114.84	19
GRBbasic	588	1574.46	297.33	532	22	472.77	33.43	21
GRBsbt	588	1399.73	308.64	536	22	502.12	34.93	22
global budget: $\alpha 0:09$ $jEje$								
SCIPbasic	579	4023.93	1476.52	310	22	1739.77	287.68	18
SCIPsbt	579	2443.85	480.58	422	22	1384.42	127.34	18
SCIPabt	579	2599.45	530.96	427	22	1384.49	127.40	18
GRBbasic	579	1726.07	315.56	504	22	920.76	45.09	20
GRBsbt	579	1574.30	346.97	517	22	961.86	45.76	21
global budget: $\alpha 0:10$ $jEje$								
SCIPbasic	588	3971.18	1391.04	325	21	1375.79	221.98	18
SCIPsbt	588	2294.64	426.34	447	21	1111.89	107.09	18
SCIPabt	588	2398.19	460.41	442	21	1079.78	103.04	19
GRBbasic	588	1772.66	311.55	515	21	632.60	34.92	20
GRBsbt	588	1814.80	403.30	508	21	695.35	34.77	20

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

Table 9: Results for MUTAG with local attack strength $s = 2$ and different global budgets. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 188, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

method	all instances				robust instances			
	#	avg-time	sgm-time	# solved	#	avg-time	sgm-time	# solved
global budget: 00:01 <i>jEje</i>								
SCIPbasic	124	74.28	45.27	124	24	176.24	162.61	24
SCIPsbt	124	33.89	22.76	124	24	84.37	79.12	24
SCIPabt	124	32.31	22.18	124	24	77.22	72.94	24
GRBbasic	124	2.30	2.16	124	24	4.91	4.82	24
GRBsbt	124	4.10	3.69	124	24	9.39	8.98	24
global budget: 00:02 <i>jEje</i>								
SCIPbasic	141	437.96	87.20	139	16	1028.39	226.02	14
SCIPsbt	141	127.08	37.84	141	16	407.73	115.97	16
SCIPabt	141	98.31	36.72	141	16	287.84	103.45	16
GRBbasic	141	94.67	11.83	141	16	205.18	15.21	16
GRBsbt	141	26.14	8.90	141	16	93.51	17.19	16
global budget: 00:03 <i>jEje</i>								
SCIPbasic	180	542.88	162.56	178	3	4807.52	1181.75	1
SCIPsbt	180	152.61	57.72	180	3	2080.69	545.97	3
SCIPabt	180	134.96	58.51	180	3	1334.79	425.77	3
GRBbasic	180	87.25	10.17	180	3	1245.54	325.46	3
GRBsbt	180	34.48	16.03	180	3	507.82	176.89	3
global budget: 00:04 <i>jEje</i>								
SCIPbasic	174	613.49	199.84	172	1	26.09	26.09	1
SCIPsbt	174	175.59	66.82	174	1	11.45	11.45	1
SCIPabt	174	133.90	67.08	174	1	13.23	13.23	1
GRBbasic	174	49.70	4.59	174	1	1.51	1.51	1
GRBsbt	174	47.76	21.59	174	1	1.55	1.55	1
global budget: 00:05 <i>jEje</i>								
SCIPbasic	167	707.14	233.17	166	0	—	—	—
SCIPsbt	167	185.08	81.61	167	0	—	—	—
SCIPabt	167	170.37	82.49	167	0	—	—	—
GRBbasic	167	23.07	2.30	167	0	—	—	—
GRBsbt	167	61.18	27.22	167	0	—	—	—
global budget: 00:06 <i>jEje</i>								
SCIPbasic	158	832.55	270.16	157	0	—	—	—
SCIPsbt	158	190.53	92.80	158	0	—	—	—
SCIPabt	158	215.46	102.23	158	0	—	—	—
GRBbasic	158	23.74	2.55	158	0	—	—	—
GRBsbt	158	57.60	27.95	158	0	—	—	—
global budget: 00:07 <i>jEje</i>								
SCIPbasic	159	841.40	265.97	158	0	—	—	—
SCIPsbt	159	219.35	94.60	159	0	—	—	—
SCIPabt	159	264.65	112.42	159	0	—	—	—
GRBbasic	159	43.72	2.98	159	0	—	—	—
GRBsbt	159	57.12	25.68	159	0	—	—	—
global budget: 00:08 <i>jEje</i>								
SCIPbasic	159	859.84	265.24	159	0	—	—	—
SCIPsbt	159	254.42	109.18	159	0	—	—	—
SCIPabt	159	299.46	134.91	159	0	—	—	—
GRBbasic	159	4.31	2.18	159	0	—	—	—
GRBsbt	159	65.85	27.45	159	0	—	—	—
global budget: 00:09 <i>jEje</i>								
SCIPbasic	165	877.25	254.34	163	0	—	—	—
SCIPsbt	165	274.26	113.80	165	0	—	—	—
SCIPabt	165	333.40	137.86	165	0	—	—	—
GRBbasic	165	4.26	2.07	165	0	—	—	—
GRBsbt	165	89.31	31.92	165	0	—	—	—
global budget: 00:10 <i>jEje</i>								
SCIPbasic	162	864.13	251.70	159	0	—	—	—
SCIPsbt	162	307.52	120.64	162	0	—	—	—
SCIPabt	162	352.19	149.21	162	0	—	—	—
GRBbasic	162	6.29	2.89	162	0	—	—	—
GRBsbt	162	141.44	49.11	162	0	—	—	—

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

Table 10: Results for MUTAG with local attack strength $s = 3$ and different global budgets. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 188, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

method	all instances				robust instances			
	#	avg-time	sgm-time	# solved	#	avg-time	sgm-time	# solved
global budget: 0:01 <i>jEje</i>								
SCIPbasic	178	216.94	138.74	178	43	484.49	438.14	43
SCIPsbt	178	97.37	68.28	178	43	191.70	180.49	43
SCIPabt	178	96.24	67.91	178	43	186.38	174.65	43
GRBbasic	178	6.75	5.92	178	43	13.62	12.68	43
GRBsbt	178	11.81	9.64	178	43	25.98	24.19	43
global budget: 0:02 <i>jEje</i>								
SCIPbasic	183	964.93	197.33	169	22	709.48	418.88	21
SCIPsbt	183	353.80	93.62	180	22	470.79	172.81	21
SCIPabt	183	348.24	92.84	180	22	461.02	162.62	21
GRBbasic	183	494.45	23.69	181	22	315.86	16.25	22
GRBsbt	183	216.83	20.71	180	22	347.31	27.91	21
global budget: 0:03 <i>jEje</i>								
SCIPbasic	185	1457.00	260.71	169	2	3634.79	747.48	1
SCIPsbt	185	424.80	119.43	182	2	3616.68	549.13	1
SCIPabt	185	418.05	118.39	182	2	3616.66	548.81	1
GRBbasic	185	744.52	83.02	183	2	3311.59	274.89	2
GRBsbt	185	257.34	44.27	182	2	3601.72	301.41	1
global budget: 0:04 <i>jEje</i>								
SCIPbasic	185	1473.13	290.37	168	1	73.10	73.10	1
SCIPsbt	185	605.01	151.85	179	1	35.09	35.09	1
SCIPabt	185	557.49	149.98	182	1	35.24	35.24	1
GRBbasic	185	910.40	82.52	182	1	2.25	2.25	1
GRBsbt	185	204.63	52.12	185	1	3.55	3.55	1
global budget: 0:05 <i>jEje</i>								
SCIPbasic	186	1464.64	307.77	167	0	—	—	—
SCIPsbt	186	661.42	211.00	182	0	—	—	—
SCIPabt	186	637.86	204.26	183	0	—	—	—
GRBbasic	186	1072.39	91.48	183	0	—	—	—
GRBsbt	186	282.76	77.28	186	0	—	—	—
global budget: 0:06 <i>jEje</i>								
SCIPbasic	186	1341.58	261.62	171	0	—	—	—
SCIPsbt	186	700.35	212.61	181	0	—	—	—
SCIPabt	186	626.75	198.82	183	0	—	—	—
GRBbasic	186	934.94	59.53	184	0	—	—	—
GRBsbt	186	334.13	93.00	186	0	—	—	—
global budget: 0:07 <i>jEje</i>								
SCIPbasic	186	1537.24	306.34	166	0	—	—	—
SCIPsbt	186	964.08	244.13	181	0	—	—	—
SCIPabt	186	919.51	235.79	180	0	—	—	—
GRBbasic	186	564.47	23.72	183	0	—	—	—
GRBsbt	186	375.94	104.92	186	0	—	—	—
global budget: 0:08 <i>jEje</i>								
SCIPbasic	186	1559.03	279.39	166	0	—	—	—
SCIPsbt	186	1065.84	242.82	179	0	—	—	—
SCIPabt	186	989.15	236.15	179	0	—	—	—
GRBbasic	186	388.40	13.63	184	0	—	—	—
GRBsbt	186	426.44	132.03	186	0	—	—	—
global budget: 0:09 <i>jEje</i>								
SCIPbasic	184	1720.80	331.98	161	0	—	—	—
SCIPsbt	184	1080.19	279.90	179	0	—	—	—
SCIPabt	184	993.48	271.16	178	0	—	—	—
GRBbasic	184	456.38	12.95	182	0	—	—	—
GRBsbt	184	496.92	164.96	184	0	—	—	—
global budget: 0:10 <i>jEje</i>								
SCIPbasic	181	1777.61	317.24	155	0	—	—	—
SCIPsbt	181	1231.10	271.38	172	0	—	—	—
SCIPabt	181	1090.15	259.43	172	0	—	—	—
GRBbasic	181	414.44	10.81	176	0	—	—	—
GRBsbt	181	545.06	187.13	181	0	—	—	—

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

Table 11: Results for MUTAG with local attack strength $s = 4$ and different global budgets. The approaches tested are SCIP basic (SCIPbasic), SCIP static bounds tightening (SCIPsbt), SCIP aggressive bounds tightening (SCIPabt), Gurobi basic (GRBbasic), and Gurobi static bounds tightening (GRBsbt). For each global budget, the number of instances is 188, but we only present comparisons when all methods give consistent results except for time out, as shown in column “#”. We compare times with respect to both average (“avg-time”) and the shifted geometric mean (“sgm-time”), as well as the number of solved instances within time limits 2 hours (“# solved”). Since robust instances are the ones where mixed-integer performance is most important (non-robust instances may frequently be found by more heuristic approaches), we also compare on the set of robust instances.

method	all instances				robust instances			
	#	avg-time	sgm-time	# solved	#	avg-time	sgm-time	# solved
global budget: 00:01 <i>jEje</i>								
SCIPbasic	178	191.35	123.98	178	42	424.81	387.63	42
SCIPsbt	178	85.15	59.78	178	42	169.78	161.44	42
SCIPabt	178	83.79	59.25	178	42	163.36	154.26	42
GRBbasic	178	6.28	5.43	178	42	13.03	12.04	42
GRBsbt	178	11.40	9.32	178	42	24.88	23.29	42
global budget: 00:02 <i>jEje</i>								
SCIPbasic	182	861.95	178.32	171	20	338.87	304.96	20
SCIPsbt	182	340.02	78.95	178	20	136.87	129.74	20
SCIPabt	182	373.47	79.59	178	20	125.34	119.02	20
GRBbasic	182	692.50	25.82	175	20	10.48	9.91	20
GRBsbt	182	292.40	22.45	179	20	21.10	19.82	20
global budget: 00:03 <i>jEje</i>								
SCIPbasic	186	1664.22	438.26	176	1	67.18	67.18	1
SCIPsbt	186	445.62	109.95	182	1	35.41	35.41	1
SCIPabt	186	469.51	109.72	182	1	35.42	35.42	1
GRBbasic	186	1246.43	111.16	179	1	2.32	2.32	1
GRBsbt	186	365.84	57.23	183	1	3.77	3.77	1
global budget: 00:04 <i>jEje</i>								
SCIPbasic	187	1910.56	613.64	172	1	63.21	63.21	1
SCIPsbt	187	766.90	144.99	178	1	33.13	33.13	1
SCIPabt	187	783.38	144.70	178	1	34.23	34.23	1
GRBbasic	187	1457.39	114.36	173	1	2.31	2.31	1
GRBsbt	187	327.42	64.44	187	1	3.74	3.74	1
global budget: 00:05 <i>jEje</i>								
SCIPbasic	185	1889.19	632.42	166	0	—	—	—
SCIPsbt	185	1167.71	244.49	174	0	—	—	—
SCIPabt	185	1156.75	242.43	174	0	—	—	—
GRBbasic	185	1553.69	113.65	171	0	—	—	—
GRBsbt	185	493.57	97.94	185	0	—	—	—
global budget: 00:06 <i>jEje</i>								
SCIPbasic	186	1990.98	693.58	163	0	—	—	—
SCIPsbt	186	1440.55	279.50	172	0	—	—	—
SCIPabt	186	1461.07	286.93	171	0	—	—	—
GRBbasic	186	1410.53	95.52	176	0	—	—	—
GRBsbt	186	524.40	116.61	186	0	—	—	—
global budget: 00:07 <i>jEje</i>								
SCIPbasic	183	2151.99	693.12	156	0	—	—	—
SCIPsbt	183	1403.40	246.99	171	0	—	—	—
SCIPabt	183	1452.16	254.92	169	0	—	—	—
GRBbasic	183	1303.05	82.57	176	0	—	—	—
GRBsbt	183	466.03	124.70	183	0	—	—	—
global budget: 00:08 <i>jEje</i>								
SCIPbasic	186	2252.93	732.66	153	0	—	—	—
SCIPsbt	186	1525.19	314.64	170	0	—	—	—
SCIPabt	186	1513.03	309.31	170	0	—	—	—
GRBbasic	186	1218.15	65.01	177	0	—	—	—
GRBsbt	186	535.35	181.75	185	0	—	—	—
global budget: 00:09 <i>jEje</i>								
SCIPbasic	186	2214.69	692.20	155	0	—	—	—
SCIPsbt	186	1599.46	285.69	163	0	—	—	—
SCIPabt	186	1333.37	260.48	175	0	—	—	—
GRBbasic	186	1037.36	41.38	178	0	—	—	—
GRBsbt	186	595.20	205.80	186	0	—	—	—
global budget: 00:10 <i>jEje</i>								
SCIPbasic	185	2251.07	734.90	154	0	—	—	—
SCIPsbt	185	1579.84	272.52	168	0	—	—	—
SCIPabt	185	1380.72	254.21	173	0	—	—	—
GRBbasic	185	1153.80	41.88	175	0	—	—	—
GRBsbt	185	578.28	217.22	185	0	—	—	—