General **cutting planes** for bound-propagation based **neural network verification**

Huan Zhang* (CMU), Shiqi Wang* (Columbia), Kaidi Xu* (Drexel University)
Linyi Li (UIUC), Bo Li (UIUC), Suman Jana (Columbia), Cho-Jui Hsieh (UCLA),
Zico Kolter (CMU/Bosch) (*co-first authors)

Paper: arxiv.org/pdf/**2208.05740**.pdf
Code: abcrown.org

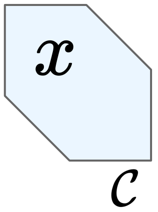
NEURAL INFORMATION PROCESSING SYSTEMS

αβ CROWN

Winner of International Verification of Neural Networks Competitions (VNN-COMP 2021,2022)

What is Neural Network Verification?

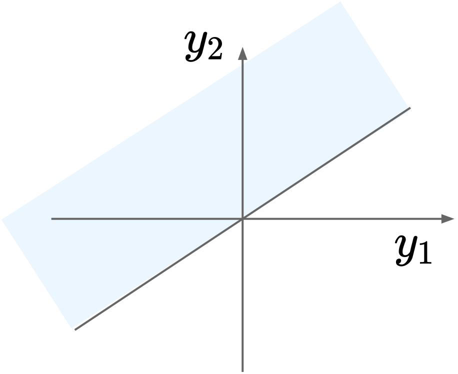
Input specifications x → Neural Network (non-linear, non-convex) → $y = f(x)$ Output specifications



$$\forall x \in \mathcal{C}$$

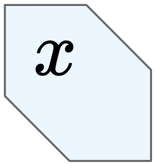
Prove or disprove $g(y) \geq 0$

Formally guarantees safety, robustness, fairness, correctness, trustworthiness, etc



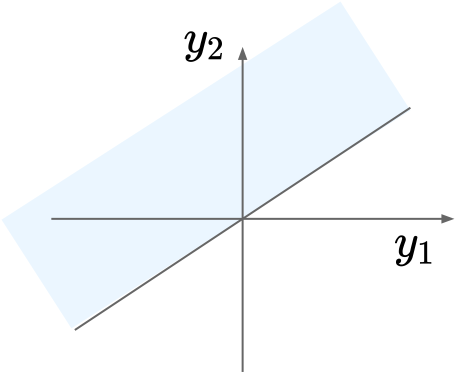
What is Neural Network Verification?

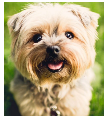
Input specifications x → **Neural Network (non-linear, non-convex)** → $y = f(x)$ Output specifications



$$\forall x \in \mathcal{C}$$

Prove or disprove $g(y) \geq 0$



e.g.,  + 

x Adv. noise

Formally guarantees that network prediction is robust

e.g., $g(y) := y_2 - y_1$

(prediction stays the same)

Why the Verification Problem is Challenging?

This is the fundamental problem we want to solve (Wong & Kolter 2018, Salman et al. 2019):

$$f^* = \min x^{(L)}$$

Last layer output f(x), at layer L

pre-activation

$$\text{s.t.} \quad x^{(i)} = W^{(i)} \hat{x}^{(i-1)} + b^{(i)} \qquad i \in \{1, \cdots, L\}$$

Linear constraints

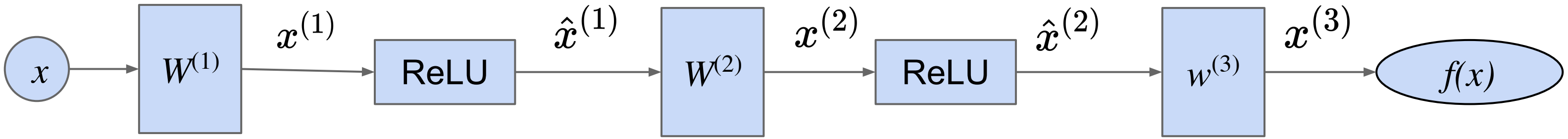
$$\hat{x}^{(i)} = \sigma(x^{(i)}) \qquad i \in \{1, \cdots, L-1\}$$

post-activation

$$\hat{x}^{(0)} = x, \qquad x \in \mathcal{C}$$

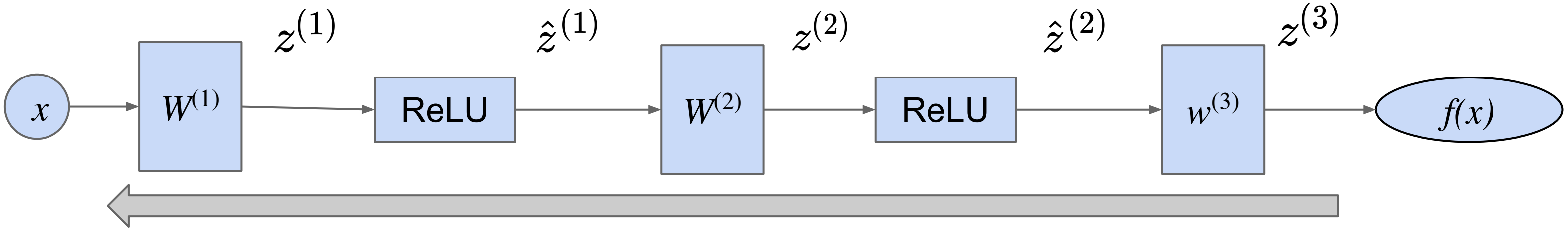
Input set

ReLU can be encoded as a mixed integer programming (**MIP**) problem (Tjeng et al. 2017), but very slow and can hardly scale up



Bound-propagation-based neural network verifiers

- Solve *relaxed* problems: **GPU accelerated, without LP/MIP solvers**
  - **CROWN** (Zhang et al., 2018) initially for feedforward networks
  - **auto_LiRPA** (Xu et al., 2020): Generalization to general computational graphs (2019)
  - ***α*-CROWN** (Xu et al., 2021) with optimizable and tighter bounds



Propagate bounds backwards (very scalable)

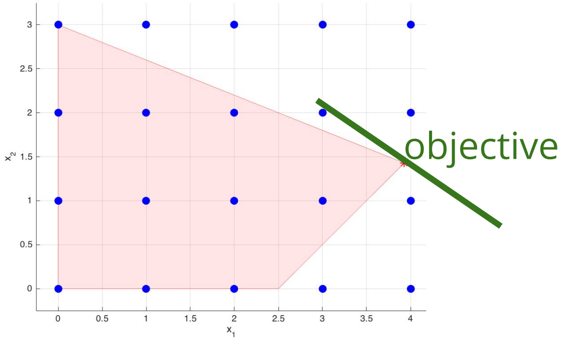
$$\min_{x \in \mathcal{C}} f(x) \geq \min_{x \in \mathcal{C}} \boldsymbol{a}_{\text{CROWN}}^{\top} x + c_{\text{CROWN}} := \min_{x \in \mathcal{C}} f_{\text{CROWN}}(x)$$

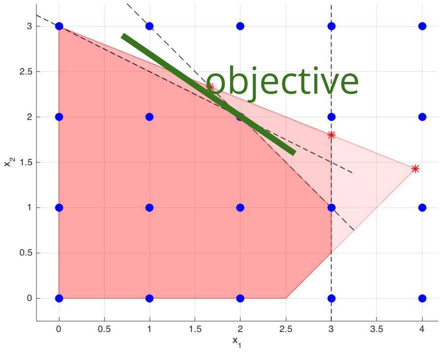
Bound-propagation-based neural network verifiers

- Solve *relaxed* problems: **GPU accelerated, without LP/MIP solvers**
  - **CROWN** (Zhang et al., 2018) initially for feedforward networks
  - **auto_LiRPA** (Xu et al., 2020): Generalization to general computational graphs (2019)
  - ***α*-CROWN** (Xu et al., 2021) with optimizable and tighter bounds
- Branch and bound (BaB)
  - ***β*-CROWN** (Wang et al., 2021): bound propagation based BaB, **won VNN-COMP 2021**
- Cutting plane methods
  - **GCP-CROWN**, this work, **won VNN-COMP 2022**
  - Aim to solve more difficult verification problems

Why using cutting plane methods for NN verification?

- Cutting plane method is an MIP solving technique that produces tighter bounds, which can be very helpful for NN verification
- We found cases where a MIP solver can solve instantly with cutting planes, while previous SOTA verifier (β-CROWN) cannot verify them even with 10min
- However, existing bound propagation based methods **cannot handle general cutting planes** used in MIP solvers





Adding cutting plane constraints improves bounds for a MIP problem

NN Verification with cutting plane constraint

- Avoid using a LP solver, need to solve this problem using bound propagation

$$\boxed{f^*} = \min_{\boldsymbol{x},\hat{\boldsymbol{x}},\mathbf{z}} f(\boldsymbol{x}) \qquad \text{s.t. } f(\boldsymbol{x}) = \boldsymbol{x}^{(L)}; \quad \hat{\boldsymbol{x}}^{(0)} = \boldsymbol{x}; \quad \boldsymbol{x} \in \mathcal{C};$$

$$\boldsymbol{x}^{(i)} = \mathbf{W}^{(i)}\hat{\boldsymbol{x}}^{(i-1)} + \mathbf{b}^{(i)}; \quad i \in [L],$$

$$\hat{x}_j^{(i)} \geq 0; \quad j \in \mathcal{I}^{(i)}, i \in [L-1]$$

$$\hat{x}_j^{(i)} \geq x_j^{(i)}; \quad j \in \mathcal{I}^{(i)}, i \in [L-1]$$

$$\hat{x}_j^{(i)} \leq u_j^{(i)} z_j^{(i)}; \quad j \in \mathcal{I}^{(i)}, i \in [L-1]$$

$$\hat{x}_j^{(i)} \leq x_j^{(i)} - l_j^{(i)}(1 - z_j^{(i)}); \quad j \in \mathcal{I}^{(i)}, i \in [L-1]$$

$$\boxed{z_j^{(i)} \in \{0,1\};} \quad j \in \mathcal{I}^{-(i)}, i \in [L-1]$$

$$\hat{x}_j^{(i)} = x_j^{(i)}; \quad j \in \mathcal{I}^{+(i)}, i \in [L-1]$$

$$\hat{x}_j^{(i)} = 0; \quad j \in \mathcal{I}^{-(i)}, i \in [L-1]$$

These equations represent the MIP formulation of ReLU

Relax MIP to LP

$$0 \leq z_j^{(i)} \leq 1$$

$$\sum_{i=1}^{L-1} \left(\boldsymbol{H}^{(i)}\boldsymbol{x}^{(i)} + \boldsymbol{G}^{(i)}\hat{\boldsymbol{x}}^{(i)} + \boldsymbol{Q}^{(i)}\mathbf{z}^{(i)} \right) \leq \boldsymbol{d}$$

Additional N cutting plane constraints

Existing bound propagation methods cannot handle this constraint

GCP-CROWN: bound-propagation with cutting planes

Theorem 3.1 (Bound propagation with general cutting planes). *Given any optimizable parameters $0 \le \alpha_j^{(i)} \le 1$ and $\boldsymbol{\beta} \ge 0$, $f^*_{\text{LP-cut}}$ is lower bounded by the following objective function:*

Optimizable variables

Conceptually inspired by (Wong & Kolter 2018)

$$g(\boldsymbol{\alpha}, \boldsymbol{\beta}) = -\epsilon \|\boldsymbol{\nu}^{(1)\top} \mathbf{W}^{(1)} \boldsymbol{x}_0\|_1 - \sum_{i=1}^{L} \boldsymbol{\nu}^{(i)\top} \mathbf{b}^{(i)} - \boldsymbol{\beta}^\top \boldsymbol{d} + \sum_{i=1}^{L-1} \sum_{j \in \mathcal{I}^{(i)}} h_j^{(i)}(\boldsymbol{\beta})$$

where variables $\boldsymbol{\nu}^{(i)}$ are obtained by propagating $\boldsymbol{\nu}^{(L)} = -1$ throughout all $i \in [L-1]$:

$$\nu_j^{(i)} = \boldsymbol{\nu}^{(i+1)\top} \mathbf{W}_{:,j}^{(i+1)} - \boldsymbol{\beta}^\top \boxed{(\boldsymbol{H}_{:,j}^{(i)} + \boldsymbol{G}_{:,j}^{(i)}),} \quad j \in \mathcal{I}^{+(i)}$$

$$\nu_j^{(i)} = -\boldsymbol{\beta}^\top \boldsymbol{H}_{:,j}^{(i)}, \quad j \in \mathcal{I}^{-(i)}$$

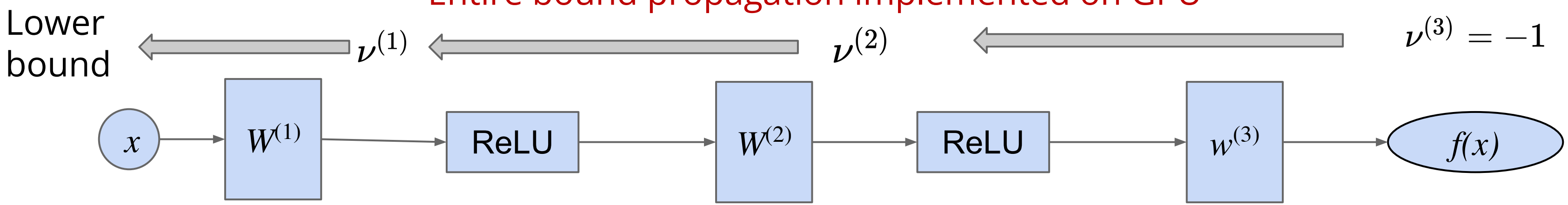
$$\nu_j^{(i)} = \pi_j^{(i)*} - \alpha_j^{(i)} [\hat{\nu}_j^{(i)}]_- - \boldsymbol{\beta}^\top \boxed{\boldsymbol{H}_{:,j}^{(i)},} \quad j \in \mathcal{I}^{(i)}$$

ν is the propagated variable (bound propagation)

Cutting plane coefficients (previous works can be seen as **special cases** with H and G being 0)

Here $\hat{\nu}_j^{(i)}$, $\pi_j^{(i)*}$ and $h_j^{(i)}(\boldsymbol{\beta})$ are defined for each unstable neuron $j \in \mathcal{I}^{(i)}$ (see paper for detailed formulation)

Entire bound propagation implemented on GPU

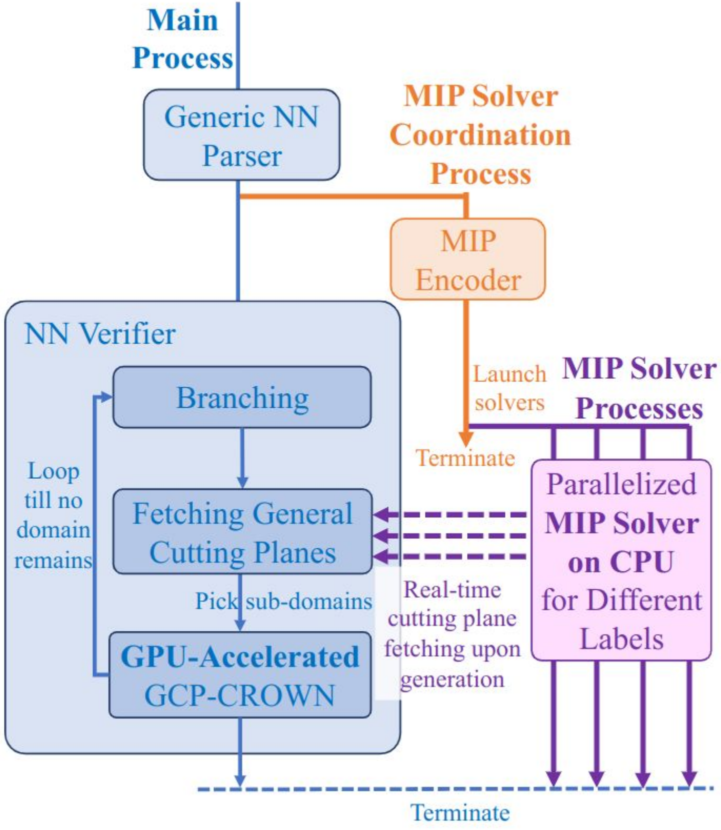


How to find cutting planes?

Cutting plane constraint:

$$\sum_{i=1}^{L-1}\left(\boldsymbol{H}^{(i)}\boldsymbol{x}^{(i)}+\boldsymbol{G}^{(i)}\hat{\boldsymbol{x}}^{(i)}+\boldsymbol{Q}^{(i)}\mathbf{z}^{(i)}\right)\leq \boldsymbol{d}$$

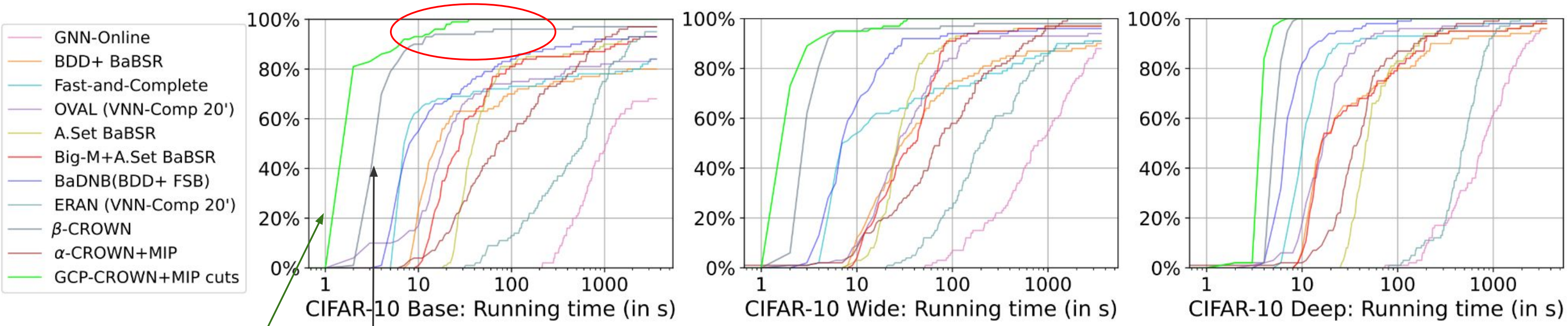
- So far, we export the cutting planes from a MIP solver running in parallel
- Future work: more efficient ways to find cutting planes; specialized cutting planes for NN verification



Results: VNN-COMP 2020 (oval20)

- Completely solved (no timeout **for the first time in literature**)
- average time of <5 seconds per instances

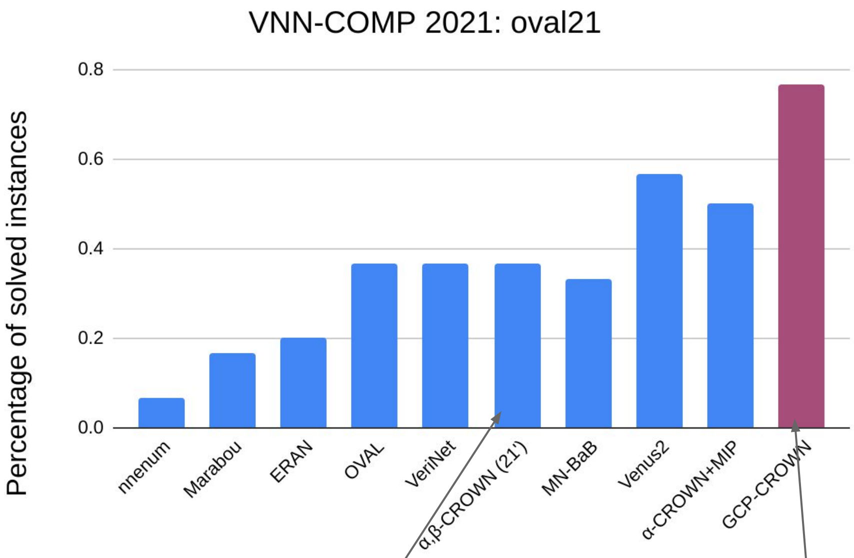
GCP-CROWN solved 100% instances within ~20s
β-CROWN (VNN-COMP 2021 winner) cannot solve 3 hard instances even using a hour

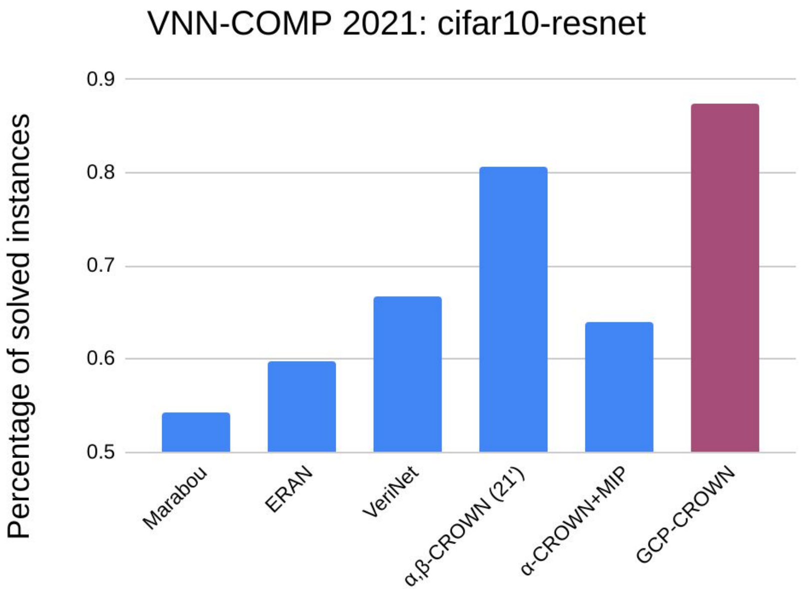


GCP-CROWN (ours, 2022)

β-CROWN (ours, 2021)

Results: VNN-COMP 2021 models





Almost **2x** instances verified compared to VNN-2021 winner

VNN-2021 winner

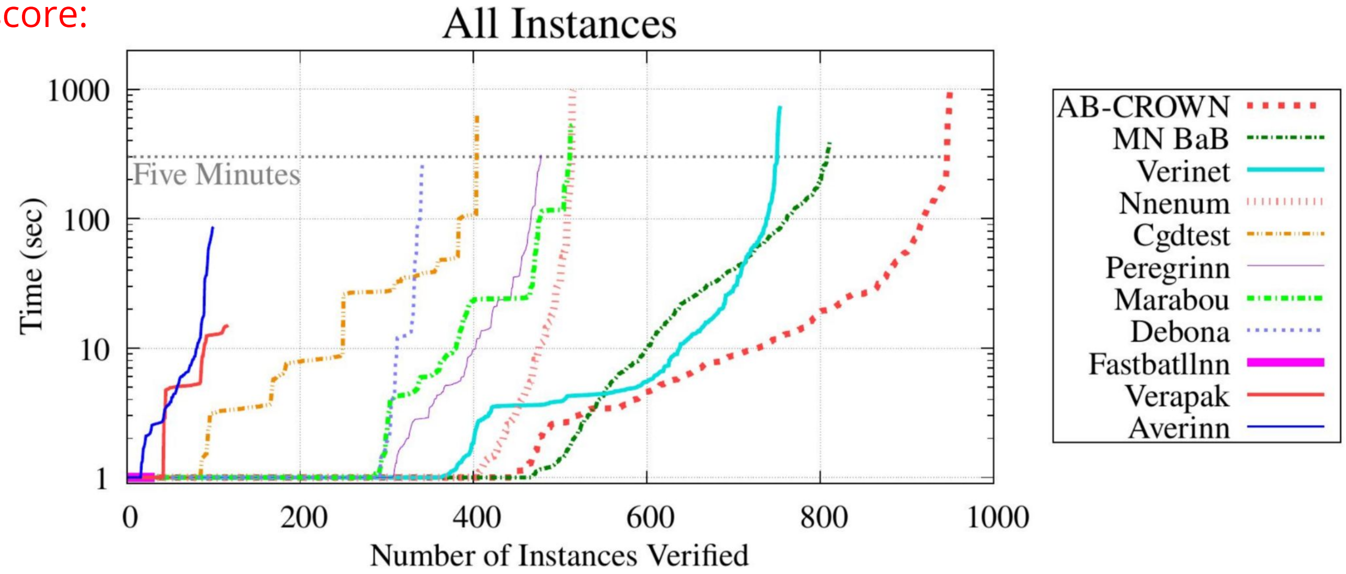
Results: VNN-COMP 2022 (latest competition)

GCP-CROWN has been **integrated into our α,β-CROWN verifier**, **winner of VNN-COMP 2022** (https://sites.google.com/view/vnn2022)

Total Score

#	Tool	Score
1	α,β-CROWN	1274.9
2	MN BaB	1017.3
3	Verinet	892.5
4	Nnenum	534.0
5	Cgdtest	406.4
6	Peregrinn	399.0
7	Marabou	380.6
8	Debona	222.9
9	Fastbatllnn	100.0
10	Verapak	98.2
11	Averinn	29.1

Max score: 1300



Thank you!

Email: huan@huan-zhang.com

α,β-CROWN Verification Tool:

abCROWN.org

(includes implementations of CROWN, α-CROWN, β-CROWN, GCP-CROWN and BaB-Attack)