The Fourth International Verification of Neural Networks Competition (VNN-COMP 2023): Summary and Results

Christopher Brix[1], Stanley Bak[2], Changliu Liu[3], and Taylor T. Johnson[4]

[1] RWTH Aachen University, Aachen, Germany
brix@cs.rwth-aachen.de
[2] Stony Brook University, Stony Brook, New York, USA
stanley.bak@stonybrook.edu
[3] Carnegie Mellon University, Pittsburgh, Pennsylvania, USA
cliu6@andrew.cmu.edu
[4] Vanderbilt University, Nashville, Tennessee, USA
taylor.johnson@vanderbilt.edu

Abstract

This report summarizes the 4th International Verification of Neural Networks Competition (VNN-COMP 2023), held as a part of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems (FoMLAS), that was collocated with the 35th International Conference on Computer-Aided Verification (CAV). VNN-COMP is held annually to facilitate the fair and objective comparison of state-of-the-art neural network verification tools, encourage the standardization of tool interfaces, and bring together the neural network verification community. To this end, standardized formats for networks (ONNX) and specification (VNN-LIB) were defined, tools were evaluated on equal-cost hardware (using an automatic evaluation pipeline based on AWS instances), and tool parameters were chosen by the participants before the final test sets were made public. In the 2023 iteration, 7 teams participated on a diverse set of 10 scored and 4 unscored benchmarks. This report summarizes the rules, benchmarks, participating tools, results, and lessons learned from this iteration of this competition.

1 Introduction

Deep learning based systems are increasingly being deployed in a wide range of domains, including recommendation systems, computer vision, and autonomous driving. While the nominal performance of these methods has increased significantly over the last years, often even surpassing human performance, they largely lack formal guarantees on their behavior. However, in safety-critical applications, including autonomous systems, robotics, cybersecurity, and cyber-physical systems, such guarantees are essential for certification and confident deployment.

While the literature on the verification of traditionally designed systems is wide and successful, neural network verification remains an open problem, despite significant efforts over the last years. In 2020, the International Verification of Neural Networks Competition (VNN-COMP) was established to facilitate comparison between existing approaches, bring researchers working on this problem together, and help shape future directions of the field. In 2023, the 4th iteration of the annual VNN-COMP[1] was held as a part of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems (FoMLAS) that was collocated with the 35th International Conference on Computer-Aided Verification (CAV).

This fourth iteration of the VNN-COMP continues last year's trend of increasing standardization and automatization, aiming to enable a fair comparison between the participating

[1] https://sites.google.com/view/vnn2023/home

tools and to simplify the evaluation of a large number of tools on a variety of (real-world) problems. As in the last iteration, VNN-COMP 2023 standardizes 1) neural network and specification formats, ONNX for neural networks and VNN-LIB [9] for specifications, 2) evaluation hardware, providing participants the choice of a range of cost-equivalent AWS instances with different trade-offs between CPU and GPU performance, and 3) evaluation pipelines, enforcing a uniform interface for the installation and evaluation of tools.

The competition was kicked off with the solicitation for participation in February 2023. By March, several teams had registered, allowing the rule discussion to be finalized in April 2023 (see an overview in Section 2). From April to June 2023, benchmarks were proposed and discussed. Meanwhile, the organizing team decided to continue using AWS as the evaluation platform and started to implement an automated submission and testing system for both benchmarks and tools. By mid-July 2023, seven teams submitted their tools and the organizers evaluated all entrants to obtain the final results, discussed in Section 5 and presented at FoMLAS on July 18, 2023. Discussions were structured into three issues on the official GitHub repository[2]: rules discussion, benchmarks discussion, and tool submission. All submitted benchmarks[3] and final results[4] were aggregated in separate GitHub repositories.

The remainder of this report is organized as follows: Section 2 discusses the competition rules, Section 3 lists all participating tools, Section 4 lists all benchmarks, Section 5 summarizes the results, and Section 6 concludes the report, discussing potential future improvements.

[2] https://github.com/stanleybak/vnncomp2023/issues
[3] https://github.com/ChristopherBrix/vnncomp2023_benchmarks
[4] https://github.com/ChristopherBrix/vnncomp2023_results

2 Rules

Terminology An *instance* is defined by a property specification (pre- and post-condition), a network, and a timeout). For example, one instance might consist of an MNIST classifier with one input image, a given local robustness threshold ϵ, and a specific timeout. A *benchmark* is defined as a set of related instances. For example, one benchmark might consist of a specific MNIST classifier with 100 input images, potentially different robustness thresholds ϵ, and one timeout per input.

Run-time caps Run-times are capped on a per-instance basis, i.e., any verification instance will timeout (and be terminated) after at most X seconds, determined by the benchmark proposer. These can be different for each instance. The total per-benchmark runtime (sum of all per-instance timeouts) may not exceed 6 hours per benchmark. For example, a benchmark proposal could have six instances with a one-hour timeout, or 100 instances with a 3.6-minute timeout, each. To enable a fair comparison, we measure the startup overhead for each tool by running it on a range of tiny networks and subtract the minimal overhead from the total runtime.

Hardware To allow for comparability of results, all tools were evaluated on equal-cost hardware using Amazon Web Services (AWS). Each team could decide between a range of AWS instance types (see Table 1) providing a CPU, GPU, or mixed focus.

Table 1: Available AWS instances.

	vCPUs	RAM [GB]	GPU
p3.2xlarge	8	61	V100 GPU with 16 GB memory
m5.16xlarge	64	256	✗
g5.8xlarge	32	128	A10G GPU with 24 GB memory

Scoring The final score is aggregate as the sum of all benchmark scores. Each benchmark score is the number of points (sum of instance scores discussed below) achieved by a given tool, normalized by the maximum number of points achieved by any tool on that benchmark. Thus, the tool with the highest sum of instance scores for a benchmark will get a benchmark score of 100, ensuring that all benchmarks are weighted equally, regardless of the number of constituting instances.

Instance score Each instance is scored is as follows:

- Correct hold (property proven): 10 points;
- Correct violated (counterexample found): 10 points;
- Incorrect result: -150 points (penalty increased compared to 2022).

However, the ground truth for any given instance is generally not known a priori. In the case of disagreement between tools, we, therefore, place the burden of proof on the tool claiming that a specification is violated, i.e. that a counterexample can be found, and deem it correct exactly if it produces a valid counterexample.

The provided counterexamples were supposed to define both the input and the resulting output of the networks. However, for some tools and instances, the output definition was either missing or differed from the network output as computed by the onnxruntime package used to evaluate counterexamples (by performing inference given the inputs). The competition

rules were ambiguous how this would be handled. We decided to discard all outputs in the counterexample files and base the evaluation solely on the given inputs and their respective outputs as computed by the onnxruntime. A ranking using the alternative evaluation, where incorrect or missing outputs result in a penalty can be found in Appendix B.

Time bonus As opposed to previous years, no time bonus was awarded. Instead, all tools that are compute the correct result within the time limit receive the same amount of points.

Overhead Correction The overhead of tools was measured, but only used to adapt the timeouts. It did not influence the scores, as no time bonus was awarded. To measure the tool-specific overhead, we created trivial network instances and included those in the measurements. We then observed the minimum verification time over all instances and considered that to be the overhead time for the tool.

Format As in 2022, we standardized neural networks to be in `onnx` format, specifications in `vnnlib` format, and counterexamples in a format similar to the `vnnlib` format. Further, tool authors were required to provide scripts fully automating the installation process of their tool, including the acquisition of any licenses that might be needed. Similar to the previous year, a preparation and execution script had to be provided for running their tool on a specific instance consisting of a network file, specification file, and timeout. The specifications are interpreted as definitions of counterexamples, meaning that a property is proven "correct" if the specification is shown to be unsatisfiable, conversely, the property is shown to be violated if a counterexample fulfilling the specification is found. Specifications consisted of disjunctions over conjunctions in both pre- and post-conditions, allowing a wide range of properties from adversarial robustness over multiple hyper-boxes to safety constraints to be encoded. For example, robustness with respect to inputs in a hyper-box had to be encoded as disjunctive property, where any of the other classes is predicted.

3 Participants

We list the tools and teams that participated in the VNN-COMP 2023 in Table 2 and reproduce their own descriptions of their tools below.

Table 2: Summary of the key features of participating tools. The hardware column describes the used AWS instance with `p3` and `g5` making GPUs available, see Table 1 for more details. Licenses refer to the external licenses required to use the corresponding tool, not the licensing of the tool itself.

Tool	References	Organizations	Place	Hardware	Licenses
α,β-CROWN	[49, 50, 43, 51, 28]	UIUC, CMU, UCLA, Drexel, Columbia, RWTH Aachen, Sun Yat-Sen, UMich.	**1**	`g5`	GUROBI
FastBATLLNN	[14]	University of California	7	t2	-
Marabou	[18]	Hebrew University of Jerusalem, Stanford University, NRI Secure	2	m5	GUROBI
NeuralSAT	[11]	George Mason University	4	g5	GUROBI
nnenum	[7, 6]	Stony Brook University	5	m5	-
NNV	[41, 24]	Vanderbilt University, University of Nebraska	6	m5	MATLAB
PyRAT	[4]	Universite Paris-Saclay, CEA, List	3	m5	-

3.1 α,β-CROWN

Team α,β-CROWN is developed by a multi-institutional team from UIUC, CMU, UCLA, Drexel, Columbia University, RWTH Aachen University, Sun Yat-Sen University, and the University of Michigan.

- Team leaders: Huan Zhang (UIUC/CMU) and Linyi Li (UIUC)
- VNN-COMP 2023 team members: Zhouxing Shi (UCLA), Christopher Brix (RWTH Aachen University), Kaidi Xu (Drexel University), Xiangru Zhong (Sun Yat-Sen University), Qirui Jin (University of Michigan), Zhuowen Yuan (UIUC).
- Advisors: Bo Li (UIUC), Suman Jana (Columbia University), Cho-Jui Hsieh (UCLA), Zico Kolter (CMU)

Description α,β-CROWN (`alpha-beta-CROWN`) is an efficient neural network verifier based on the linear bound propagation framework, built on a series of works on bound-propagation-based neural network verifiers: CROWN [52, 49], α-CROWN [50], β-CROWN [43], GCP-CROWN [51], and nonlinear branch-and-bound [28]. The core techniques in α,β-CROWN combine the efficient and GPU-accelerated linear bound propagation method with specialized branch-and-bound methods. Previously, branch-and-bound was conducted for ReLU neural networks only. In this year, we extended branch-and-bound for general nonlinearities [28], and enabled strong verification for generic nonlinear computation graphs, such as in the ML4ACOPF

benchmark. In addition, we also recently added the support for bounding neural network preimages and the use of output constraints to tighten robustness verification [22].

The linear bound propagation algorithms in α,β-CROWN are based on our `auto_LiRPA` library [49], which supports general neural network architectures (including convolutional layers, pooling layers, residual connections, recurrent neural networks, and Transformers) and a wide range of nonlinear functions (e.g., ReLU, tanh, trigonometric functions, sigmoid, max pooling and average pooling), and is efficiently implemented on GPUs with Pytorch and CUDA. We jointly optimize intermediate layer bounds and final layer bounds using gradient ascent (referred to as α-CROWN or optimized CROWN/LiRPA [50]). Most importantly, we use branch and bound [8] (BaB) and incorporate split constraints in BaB into the bound propagation procedure efficiently via the β-CROWN algorithm [43], use cutting-plane method in GCP-CROWN [51] to further tighten the bound, and support general nonlinearities in the branch-and-bound [28]. For smaller networks, we also use a mixed integer programming (MIP) formulation [34] combined with tight intermediate layer bounds from α-CROWN (referred to as α-CROWN + MIP [51]). The combination of efficient, optimizable and GPU-accelerated bound propagation with BaB produces a powerful and scalable neural network verifier.

Link https://github.com/Verified-Intelligence/alpha-beta-CROWN (latest version)

Competition submission https://github.com/Verified-Intelligence/alpha-beta-CROWN_vnncomp23 (only for reproducing competition results; please use the latest version for other purposes)

Hardware and licenses CPU and GPU with 32-bit or 64-bit floating point; Gurobi license required for the `gtrsb` benchmark.

Participated benchmarks All benchmarks.

3.2 FastBATLLNN

Team James Ferlez (Developer), Haitham Khedr (Tester), and Yasser Shoukry (Supervisor) (University of California, Irvine)

Description FastBATLLNN [14] is a fast verifier of box-like (hyper-rectangle) output properties for Two-Level Lattice (TLL) Neural Networks (NN). FastBATLLNN uses both the unique semantics of the TLL architecture and the decoupled nature of box-like output constraints to provide a fast, polynomial-time verification algorithm: that is, polynomial-time in the number of neurons in the TLL NN to be verified (for a fixed input dimension). FastBATLLNN fundamentally works by converting the TLL verification problem into a region enumeration problem for a hyperplane arrangement (the arrangement is jointly derived from the TLL NN and the verification property). However, as FastBATLLNN leverages the unique properties of TLL NNs and box-like output constraints, its use is necessarily limited to verification problems formulated in those terms. Hence, FastBATLLNN can only compete on the `tllverifybench` benchmark.

Link https://github.com/jferlez/FastBATLLNN-VNNCOMP

Commit 6100258b50a3fadf9792aec4ccf39ed14778e338

Hardware and licenses CPU; no licenses required

Participated benchmarks `tllverifybench`

3.3 Marabou

Team Haoze Wu (Stanford University), Clark Barrett (Stanford University), Guy Katz (Hebrew University of Jerusalem)

Description Marabou [19] is a user-friendly Neural Network Verification toolkit that can answer queries about a network's properties by encoding and solving these queries as constraint satisfaction problems. It has both Python/C++ APIs through which users can load neural networks and define arbitrary linear properties over the neural network. Marabou supports many different linear, piecewise-linear, and non-linear [47, 44] operations and architectures (e.g., FFNNs, CNNs, residual connections, Graph Neural Networks [45]).

Under the hood, Marabou employs a uniform solving strategy for a given verification query. In particular, Marabou performs complete analysis that employs a specialized convex optimization procedure [48] and abstract interpretation [32, 45]. It also uses the Split-and-Conquer algorithm [46] for parallelization. [5]

Link https://github.com/NeuralNetworkVerification/Marabou

Commit 1a3ca6010b51bba792ef8ddd5e1ccf9119121bd8

Hardware and Licenses CPU, no license required. Can also be accelerated with Gurobi (which requires a license)

Participated benchmarks `acasxu`, `cgan`, `collins_rul_cnn`, `dist_shift`, `ml4acopf`, `nn4sys`, `tllverifybench`, `traffic_signs_recognition`, `vggnet16`, `vit`.

3.4 nnenum

Team Ali Arjomandbigdeli (Student), Stanley Bak (Supervisor) (Stony Brook University)

Description The nnenum tool [6] uses multiple levels of abstraction to achieve high-performance verification of ReLU networks without sacrificing completeness [5]. The core verification method is based on reachability analysis using star sets [39], combined with the ImageStar method [36] to propagate stes through all linear layers supported by the ONNX runtime, such as convolutional layers with arbitrary parameters. The tool is written in Python 3 and uses GLPK for LP solving. New this year, we added support for single lower and single upper bounds propagation in addition to zonotopes, similar to the DeepPoly method or the CROWN approach. We also added an option to use Gurobi instead of GLPK for LP solving.

Link https://github.com/aliabigdeli/nnenum

Commit 2e14cdf09f66d8b3c4622ad29d8bedd815d056e8

Hardware and licences CPU, Gurobi license (optional)

Participated benchmarks `acasxu`, `cgan`, `collins-rul-cnn`, `nn4sys`, `tllverifybench`, `vggnet16`.

[5] Thanks to the authors of the $\alpha - \beta$-CROWN team, an unsoundness issue of the competition version of Marabou on the `ViT` benchmarks was discovered. The networks in that benchmark contain bilinear and softmax connections. For this benchmark, the competition version of Marabou first performs DeepPoly-style abstract interpretation and then encodes the verification problem in the Gurobi optimizer. It turns out that Gurobi can report "Infeasible" on benchmarks where counter-examples are expected. The Marabou team is actively looking into resolving this issue.

3.5 NNV

Team Diego Manzanas Lopez (Vanderbilt University), Neelanjana Pal (Vanderbilt University), Samuel Sasaki (Vanderbilt University), Hoang-Dung Tran (University of Nebraska-Lincoln), Taylor T. Johnson (Vanderbilt University)

Description The Neural Network Verification (NNV) Tool [41, 24] is a formal verification software tool for deep learning models and cyber-physical systems with neural network components written in MATLAB and available at `https://github.com/verivital/nnv`. NNV uses a star-set state-space representation and reachability algorithm that allows for a layer-by-layer computation of exact or overapproximate reachable sets for feed-forward [39], convolutional [36], semantic segmentation (SSNN) [40], and recurrent (RNN)[38] neural networks, as well as neural network control systems (NNCS) [37, 41] and neural ordinary differential equations (Neural ODEs) [26]. The star-set based algorithm is naturally parallelizable, which allows NNV to be designed to perform efficiently on multi-core platforms. Additionally, if a particular safety property is violated, NNV can be used to construct and visualize the complete set of counterexample inputs for a neural network (exact-analysis). For this competition, we make use of the same verification approach across all properties (i.e., no fine-tuning for individual benchmarks). First, we perform a simulation-guided search for counterexamples for a fixed number of samples. If no counterexamples are found (i.e., demonstrate that the property is SAT), then we utilize an iterative refinement approach using reachability analysis to verify the property (UNSAT). This consists of performing reachability analysis using a relax-approximation method [40], if not verified, then a less conservative approximation based on zonotope pre-filtering approach [35], and finally using the exact analysis when possible [36] until the specification is verified or there is a timeout.

Link `https://github.com/verivital/nnv`

Commit fb858b070d7c2fb1f036ac7fc374e1b9dfb5055e

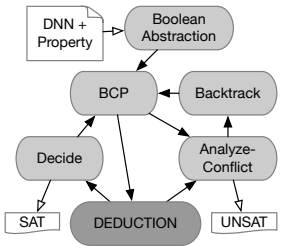
Hardware and licenses CPU, MATLAB license.

Participated Benchmarks `acasxu`, `cgan`, `collins-rul-cnn`, `dist-shift`, `nn4sys`, `tllverifybench`.

3.6 NeuralSAT

Team Hai Duong and Thanhvu Nguyen (George Mason).

Description NeuralSAT [11] integrates *clause learning* in modern constraint solving to enhance performance and address SAT challenges like backtracking, with an *abstraction-based theory solver* in SMT solving and abstraction-based DNN verification to expedite infeasibility checking. The figure on the right gives an overview of NeuralSAT, which implements the DPLL(T) framework used in modern SMT solvers such as Z3 and CVC. The design of NeuralSAT is inspired by the core algorithms used in SMT solvers such as CDCL components (light shades) and theory solving (dark shade). NeuralSAT can be considered as a native satisfiability solver for DNNs and inherits the benefits of modern SMT solvers. The tool is written in Python and uses Gurobi for LP solving. Since this VNN-COMP participation, NeuralSAT has been updated with parallel DPLL search and many other optimizations.



Link `https://github.com/dynaroars/neuralsat`

Commit 5693c3da130942283744ce56c2e74ac6c16eef94

Hardware and licences GPU, Gurobi License

Participated benchmarks `acasxu`, `cgan`, `collins-rul-cnn`, `dist-shift`, `nn4sys`, `vggnet16`, `tllverifybench`, `trafic-signs-recognition`, `reach-prob-density`, `metaroom`.

3.7 PyRAT

Team Augustin Lemesle, Julien Lehmann, Serge Durand, Zakaria Chihani (CEA-List)

Description PyRAT (Python Reachability Assessment Tool) is a tool for verifying various types of neural networks based on Python and abstract interpretation techniques. PyRAT can leverage multiple abstract domains such as Intervals, Zonotopes, or Polyhedras to efficiently compute bounds for different architectures of neural networks such as dense, convolutional, residual or recurrent neural networks. It supports ReLU, Sigmoid, Tanh, Softmax activation functions. PyRAT can efficiently work on both CPU and GPU and is sound w.r.t. floating points computation when using CPU. PyRAT is correct as the output bounds reached will always be a sound over approximation of the results. Depending on the benchmark, several verification modes and domains can be selected. For smaller networks and problems, PyRAT can leverage branch and bound strategies on the inputs with heuristics like ReCIPH [12]. While on larger network, PyRAT will use GPU computation as well as local counterexample checking to verify or falsify a property.

Link https://git.frama-c.com/pub/pyrat

Commit 50288df457653d8767c2d6f5481cc3e72e3d6727

Hardware and licenses CPU and GPU, closed source CEA licence.

Participated benchmarks `acasxu`, `cgan`, `collins_rul_cnn`, `dist_shift`, `nn4sys`, `tllverifybench`, `traffic_signs_recognition`, `vggnet16`.

4 Benchmarks

In this section, we provide an overview of all benchmarks, reproducing the benchmark proposers' descriptions.

Table 3: Overview of all scored benchmarks.

Category	Benchmark	Application	Network Types	# Params	Effective Input Dim
Complex	cGAN	Image Generation & Image Prediction	Complex (Conv. + Vision Transformer)	500k - 68M	5
	NN4Sys	Dataset Indexing & Cardinality Prediction	Complex (ReLU + Sigmoid)	33k - 37M	1-308
	ml4acopf	Power System	Complex (ReLU + Trigonometric + Sigmoid)	4k-680k	22 - 402
	ViT	Vision	Conv. + Residual + Softmax + BatchNorm	68k - 76k	3072
CNN & ResNet	Collins RUL CNN	Condition Based Maintenance	Conv. + ReLU, Dropout	60k - 262k	400 - 800
	VGGNet16	Image Classification	Conv. + ReLU + MaxPool	138M	150k
	Traffic Signs Recognition	Image Classification	Conv. + Sign + MakPool + BatchNorm	905k - 1.7M	2.7k - 12k
FC	TLL Verify Bench	Two-Level Lattice NN	Two-Level Lattice NN (FC. + ReLU)	17k - 67M	2
	Acas XU	Collision Detection	FC. + ReLU	13k	5
	Dist Shift	Distribution Shift Detection	FC. + ReLU + Sigmoid	342k - 855k	792

4.1 cGAN

Proposed by Feiyang Cai, Ali Arjomandbigdeli, Stanley Bak (Stony Brook University)

Motivation While existing neural network verification benchmarks focus on discriminative models, the exploration of practical and widely used generative networks remains neglected in terms of robustness assessment. This benchmark introduces a set of image generation networks specifically designed for verifying the robustness of the generative networks.

Networks The generative networks are trained using conditional generative adversarial networks (cGAN), whose objective is to generate camera images that contain a vehicle obstacle located at a specific distance in front of the ego vehicle, where the distance is controlled by the input distance condition. The network to be verified is the concatenation of a generator and a discriminator. The generator takes two inputs: 1) a distance condition (1D scalar) and 2) a noise vector controlling the environment (4D vector). The output of the generator is the generated image. The discriminator takes the generated image as input and outputs two values: 1) a real/fake score (1D scalar) and 2) a predicted distance (1D scalar). Several different models with varying architectures (CNN and vision transformer) and image sizes (32x32, 64x64) are provided for different difficulty levels.

Specifications The verification task is to check whether the generated image aligns with the input distance condition, or in other words, verify whether the input distance condition matches the predicted distance of the generated image. In each specification, the inputs (condition distance and latent variables) are constrained in small ranges, and the output is the predicted distance with the same center as the condition distance but with slightly larger range.

Link https://github.com/feiyang-cai/cgan_benchmark2023

4.2 NN4Sys

Proposed by the α,β-CROWN team with collaborations with Cheng Tan, Haoyu He and Shuyi Lin at Northeastern University.

Application The benchmark contains networks for database learned index, video streaming learned adaptive bitrate, and learned cardinality estimation which map inputs from various dimensions to 1-dimension outputs.

- *Background*: learned index, learned cardinality, and learned adaptive bitrate are all instances in neural networks for computer systems (NN4Sys), which are neural network based methods performing system operations. These classes of methods show great potential but have one drawback—the outputs of an NN4Sys model (a neural network) can be arbitrary, which may lead to unexpected issues in systems.

- *What to verify*: our benchmark provides multiple pairs of (1) trained NN4Sys model and (2) corresponding specifications. We design these pairs with different parameters such that they cover a variety of user needs and have varied difficulties for verifiers. We describe benchmark details in our NN4SysBench report: http://naizhengtan.github.io/doc/papers/nn4sys23lin.pdf.

- *Translating NN4Sys applications to a VNN benchmark*: the original NN4Sys applications have some sophisticated structures that are hard to verify. We tailored the neural networks and their specifications to be suitable for VNN-COMP. For example, learned index [23] contains multiple NNs in a tree structure that together serve one purpose. However, this cascading structure is inconvenient/unsupported to verify because there is a "switch" operation—choosing one NN in the second stage based on the prediction of the first stage's NN. To convert learned indexes to a standard form, we train a monolithic (larger) NN.

- *A note on broader impact*: using NNs for systems is a broad topic, but many existing works lack strict safety guarantees. We believe that NN Verification can help system developers gain confidence to apply NNs to critical systems. We hope our benchmark can be an early step toward this vision.

Networks This benchmark has twelve networks with different parameters: two for learned indexes, four for learned cardinality estimation and six for learned adaptive bitrate. The learned index uses fully-connected feed-forward neural networks. The other two—the learned cardinality and the learned adaptive bitrate—has a relatively sophisticated internal structure. Please see our NN4SysBench report (URL listed above) for details

Specifications For learned indexes, the specification aims to check if the prediction error is bounded. The specification is a collection of pairs of input and output intervals such that any input in the input interval should be mapped to the corresponding output interval. For learned cardinality estimation and learned adaptive bitrate, the specifications check the prediction error bounds (similar to the learned indexes) and monotonicity of the networks. By monotonicity specifications, we mean that for two inputs, the network should produce a larger output for the larger input, which is required by cardinality estimation or adaptive bitrate.

Link: https://github.com/Khoury-srg/VNNComp23_NN4Sys

4.3 Collins-RUL-CNN

Proposed by Collins Aerospace, Applied Research & Technology (website).

Motivation Machine Learning (ML) is a disruptive technology for the aviation industry. This particularly concerns safety-critical aircraft functions, where high-assurance design and verification methods have to be used in order to obtain approval from certification authorities for the new ML-based products. Assessment of correctness and robustness of trained models, such as neural networks, is a crucial step for demonstrating the absence of unintended functionalities [13, 21]. The key motivation for providing this benchmark is to strengthen the interaction between the VNN community and the aerospace industry by providing a realistic use case for neural networks in future avionics systems [20].

Application Remaining Useful Life (RUL) is a widely used metric in Prognostics and Health Management (PHM) that manifests the remaining lifetime of a component (e.g., mechanical bearing, hydraulic pump, aircraft engine). RUL is used for Condition-Based Maintenance (CBM) to support aircraft maintenance and flight preparation. It contributes to such tasks as augmented manual inspection of components and scheduling of maintenance cycles for components, such as repair or replacement, thus moving from preventive maintenance to *predictive* maintenance (do maintenance only when needed, based on component's current condition and estimated future condition). This could allow to eliminate or extend service operations and inspection periods, optimize component servicing (e.g., lubricant replacement), generate inspection and maintenance schedules, and obtain significant cost savings. Finally, RUL function can also be used in airborne (in-flight) applications to dynamically inform pilots on the health state of aircraft components during flight. Multivariate time series data is often used as RUL function input, for example, measurements from a set of sensors monitoring the component state, taken at several subsequent time steps (within a time window). Additional inputs may include information about the current flight phase, mission, and environment. Such highly multi-dimensional input space motivates the use of Deep Learning (DL) solutions with their capabilities of performing automatic feature extraction from raw data.

Networks The benchmark includes 3 convolutional neural networks (CNNs) of different complexity: different numbers of filters and different sizes of the input space. All networks contain only convolutional and fully connected layers with ReLU activations. All CNNs perform the regression function. They have been trained on the same dataset (time series data for mechanical component degradation during flight).

Specifications We propose 3 properties for the NN-based RUL estimation function. First, two properties (robustness and monotonicity) are local, i.e., defined around a given point. We provide a script with an adjustable random seed that can generate these properties around input points randomly picked from a test dataset. For robustness properties, the input perturbation (delta) is varied between 5% and 40%, while the number of perturbed inputs varies between 2 and 16. For monotonicity properties, monotonic shifts between 5% and 20% from a given point are considered. Properties of the last type ("if-then") require the output (RUL) to be in an expected value range given certain input ranges. Several if-then properties of different complexity are provided (depending on range widths).

Link `https://github.com/loonwerks/vnncomp2022`

Paper Available in [20] or on request.

4.4 VGGNET16 2023

Proposed by Stanley Bak, Stony Brook University

Motivation This benchmark tries to scale up the size of networks being analyzed by using the well-studied VGGNET-16 architecture [31] that runs on ImageNet. Input-output properties are proposed on pixel-level perturbations that can lead to image misclassification.

Networks All properties are run on the same network, which includes 138 million parameters. The network features convolution layers, ReLU activation functions, as well as max pooling layers.

Specifications Properties analyzed ranged from single-pixel perturbations to perturbations on all 150528 pixles (L-infinity perturbations). A subset of the images was used to create the specifications, one from each category, which was randomly chosen to attack. Pixels to perturb were also randomly selected according to a random seed.

Link https://github.com/stanleybak/vggnet16_benchmark2022/

4.5 TLL Verify Bench

Proposed by James Ferlez (University of California, Irvine)

Motivation This benchmark consists of Two-Level Lattice (TLL) NNs, which have been shown to be amenable to fast verification algorithms (e.g. [14]). Thus, this benchmark was proposed as a means of comparing TLL-specific verification algorithms with general-purpose NN verification algorithms (i.e. algorithms that can verify arbitrary deep, fully-connected ReLU NNs).

Networks The networks in this benchmark are a subset of the ones used in [14, Experiment 3]. Each of these TLL NNs has $n = 2$ inputs and $m = 1$ output. The architecture of a TLL NN is further specified by two parameters: N, the number of local linear functions, and M, the number of selector sets. This benchmark contains TLLs of sizes $N = M = 8, 16, 24, 32, 40, 48, 56, 64$, with 30 randomly generated examples of each (the generation procedure is described in [14, Section 6.1.1]). At runtime, the specified verification timeout determines how many of these networks are included in the benchmark so as to achieve an overall 6-hour run time; this selection process is deterministic. Finally, a TLL NN has a natural representation using multiple computation paths [14, Figure 1], but many tools are only compatible with fully-connected networks. Hence, the ONNX models in this benchmark implement TLL NNs by "stacking" these computation paths to make a fully connected NN (leading to sparse weight matrices: i.e. with many zero weights and biases). The `TLLnet` class (https://github.com/jferlez/TLLnet) contains the code necessary to generate these implementations via the `exportONNX` method.

Specifications All specifications have as input constraints the hypercube $[-2, 2]^2$. Since all networks have only a single output, the output properties consist of a randomly generated real number and a randomly generated inequality direction. Random output samples from the network are used to roughly ensure that the real number property has an equal likelihood of being within the output range of the NN and being outside of it (either above or below all NN outputs on the input constraint set). The inequality direction is generated independently and with each direction having an equal probability. This scheme biases the benchmark towards verification problems for which counterexamples exist.

Link https://github.com/jferlez/TLLVerifyBench

Commit 199d2c26d0ec456e62906366b694a875a21ff7ef

4.6 Traffic Signs Recognition

Proposed by Mădălina Eraşcu and Andreea Postovan (West University of Timisoara, Romania)

Motivation Traffic signs play a crucial role in ensuring road safety and managing traffic flow in both city and highway driving. The recognition of these signs, a vital component of autonomous driving vision systems, faces challenges such as susceptibility to adversarial examples [33] and occlusions [53], stemming from diverse traffic scene conditions.

Networks Binary neural networks (BNNs) show promise in computationally limited and energy-constrained environments within the realm of autonomous driving [16]. BNNs, where weights and/or activations are binarized to ± 1, offer reduced model size and simplified convolution operations for image recognition compared to traditional neural networks (NNs).

We trained and tested various BNN architectures using the German Traffic Sign Recognition Benchmark (GTSRB) dataset [3]. This multi-class dataset, containing images of German road signs across 43 classes, poses challenges for both humans and models due to factors like perspective change, shade, color degradation, and lighting conditions. The dataset was also tested using the Belgian Traffic Signs [1] and Chinese Traffic Signs [2] datasets. The Belgium Traffic Signs dataset, with 62 classes, had 23 overlapping classes with GTSRB. The Chinese Traffic Signs dataset, with 58 classes, shared 15 classes with GTSRB. Pre-processing steps involved relabeling classes in the Belgium and Chinese datasets to match those in GTSRB and eliminating non-overlapping classes (see [27] for details).

We provide three models with the structure in Figures 1, 2, and 3. They contain QConv, Batch Normalization (BN), Max Pooling (ML), Fully Connected/Dense (D) layers. Note that the QConv layer binarizes the corresponding convolutional layer. All models were trained for 30 epochs. The model from Figure 1 was trained with images having the dimension 64px x 64 px, the one from Figure 2 with 48px x 48 px and the one from Figure 3 with 30px x 30 px. The two models involving Batch Normalization layers introduce real valued parameters besides the binary ones, while the third one contains only binary parameters (see Table 4) for statistics.

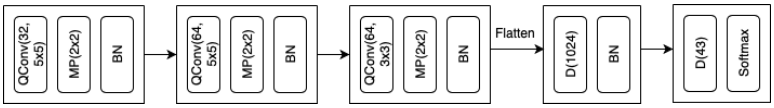


Figure 1: Accuracy Efficient Architecture for GTSRB and Belgium dataset

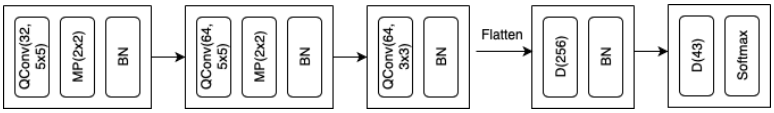


Figure 2: Accuracy Efficient Architecture for Chinese dataset

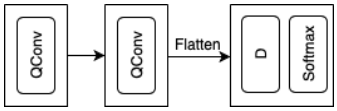


Figure 3: XNOR(QConv) architecture

Table 4: Training and Testing Statistics

Input size	Model name	Accuracy			#Params		
		German	China	Belgium	Binary	Real	Total
64px × 64px	Figure 1	96.45	81.50	88.17	1772896	2368	1775264
48px × 48px	Figure 2	95.28	83.90	87.78	904288	832	905120
30px × 30px	Figure 3	81.54	N/A	N/A	1005584	0	1005584

Specifications To evaluate the *adversarial robustness* of the networks above, we assessed perturbations within the infinity norm around zero, with the radius denoted as $\epsilon = \{1, 3, 5, 10, 15\}$. This involved randomly selecting three distinct images from the GTSRB dataset's test set for each model and generating VNNLIB files for each epsilon in the set. In total, we created 45 VNNLIB files. Due to a 6-hour total timeout constraint for solving all instances, each instance had a maximum timeout of 480 seconds. To review the generated VNNLIB specification files submitted to VNNCOMP 2023, as well as to generate new ones, please refer to https://github.com/apostovan21/vnncomp2023.

Link https://github.com/apostovan21/vnncomp2023

4.7 ViT

Proposed by the α,β-CROWN team.

Motivation Transformers [42] based on the self-attention mechanism have much more complicated architectures and contain more kinds of nonlinerities, compared to simple feedforward networks with relatively simple activation functions. It makes verifying Transformers challenging. We aim to encourage the development of verification techniques for Transformer-based models, and we also aim to benchmark neural network verifiers on relatively complicated neural network architectures and more general nonlinearities. Therefore, we propose a new benchmark with Vision Transformers (ViTs) [10]. This benchmark is developed based on our work on neural network verification for models with general nonlinearities [28].

Networks The benchmark contains two ViTs, as shown in Table 5. Considering the difficulty of verifying ViTs, we modify the ViTs and make the models relatively shallow and narrow, with significantly reduced number of layers and attention heads. Following [30], we also replace the layer normalization with batch normalization. The models are mainly trained with PGD training [25], and we also add a weighted IBP [15, 29] loss for one of the models as a regularization.

Table 5: Networks in the ViT benchmark.

Model	PGD_2_3_16	IBP_3_3_8
Layers	2	3
Attention heads	3	3
Patch size	16	8
Weight of IBP loss	0	0.01
Training ϵ	$\frac{2}{255}$	$\frac{1}{255}$
Clean accuracy	59.78%	62.21%

Specifications The specifications are generated from the robustness verification problem with ℓ_∞ perturbation. We use the CIFAR-10 dataset with perturbation size $\epsilon = \frac{1}{255}$ at test time.

We have filtered the CIFAR-10 test set to exclude instances where either adversarial examples can be found (by PGD attack [25] with 100 steps and 1000 restarts) or the vanilla CROWN-like method [52, 30] can already easily verify. We randomly keep 100 instances for each model, with a timeout threshold of 100 seconds.

Link https://github.com/shizhouxing/ViT_vnncomp2023

4.8 Real-world distribution shifts

Proposed by the Marabou team.

Motivation While robustness against handcrafted perturbations (e.g., norm-bounded) for perception networks are more commonly investigated, robustness against real-world distribution shifts [47] are less studied but of practical interests. This benchmark set contains queries for verifying the latter type of robustness.

Networks The network is a concatenation of a generative model and a MNIST classifier. The generative model is trained to take in an unperturbed image and an embedding of a particular type of distribution shifts in latent space, and produce a perturbed image. The distribution shift captured in this case is the "shear" perturbation.

Specifications The verification task is to certify that a classifier correctly classifies all images in a perturbation set, which is a set of images generated by the generative model given a fixed image and a ball centering the mean perturbations on this image (in the latent space). This mean perturbation is computed by a prior network.

Link https://github.com/wu-haoze/dist-shift-vnn-comp

4.9 ml4acopf

Proposed by Haoruo Zhao, Michael Klamkin, Mathieu Tanneau, Wenbo Chen, and Pascal Van Hentenryck (Georgia Institute of Technology), and Hassan Hijazi, Juston Moore, and Haydn Jones (Los Alamos National Laboratory).

Motivation Machine learning models are utilized to predict solutions for an optimization model known as AC Optimal Power Flow (ACOPF) in the power system. Since the solutions are continuous, a regression model is employed. The objective is to evaluate the quality of these machine learning model predictions, specifically by determining whether they satisfy the constraints of the optimization model. Given the challenges in meeting some constraints, the goal is to verify whether the worst-case violations of these constraints are within an acceptable tolerance level.

Networks The neural network designed comprises two components. The first component predicts the solutions of the optimization model, while the second evaluates the violation of each constraint that needs checking. The first component consists solely of general matrix multiplication (GEMM) and rectified linear unit (ReLU) operators. However, the second component has a more complex structure, as it involves evaluating the violation of AC constraints using nonlinear functions, including sigmoid, quadratic, and trigonometric functions such as sine and cosine. This complex evaluation component is incorporated into the network due to a limitation of the VNNLIB format, which does not support trigonometric functions. Therefore, these constraints violation evaluation are included in the neural network.

Specifications In this benchmark, four different properties are checked, each corresponding to a type of constraint violation:

1. Power balance constraints: the net power at each bus node is equal to the sum of the power flows in the branches connected to that node.
2. Thermal limit constraints: power flow on a transmission line is within its maximum and minimum limits.
3. Generation bounds: a generator's active and reactive power output is within its maximum and minimum limits.
4. Voltage magnitude bounds: a voltage's magnitude output is within its maximum and minimum limits.

The input to the model is the active and reactive load. The chosen input point for perturbation is a load profile for which a corresponding feasible solution to the ACOPF problem is known to exist. For the feasibility check, the input load undergoes perturbation. Although this perturbation does not exactly match physical laws, the objective is to ascertain whether a machine learning-predicted solution with the perturbation can produce a solution that does not significantly violate the constraints.

The scale of the perturbation and the violation threshold are altered by testing whether an adversarial example can be easily found using projected gradient descent with the given perturbation. The benchmark, provided with a fixed random seed, is robust against the simple projected gradient descent that is implemented.

Link https://github.com/AI4OPT/ml4acopf_benchmark

4.10 ACAS Xu

Networks The ACASXu benchmark consists of ten properties defined over 45 neural networks used to issue turn advisories to aircraft to avoid collisions. The neural networks have 300 neurons arranged in 6 layers, with ReLU activation functions. There are five inputs corresponding to the aircraft states, and five network outputs, where the minimum output is used as the turn advisory the system ultimately produces.

Specifications We use the original 10 properties [17], where properties 1-4 are checked on all 45 networks as was done in later work by the original authors [19]. Properties 5-10 are checked on a single network. The total number of benchmarks is therefore 186. The original verification times ranged from seconds to days—including some benchmark instances that did not finish. This year we used a timeout of around two minutes (116 seconds) for each property, in order to fit within a total maximum runtime of six hours.

5 Results

Each tool was run on each of the benchmarks and produced a `csv` result file, that was provided as feedback to the tool authors using the online execution platform. The final `csv` files for each tool as well as scoring scripts are available online: https://github.com/ChristopherBrix/vnncomp2023_results. The results were analyzed automatically to compute scores and create the statistics presented in this section.

Penalties usually occur when a tool produces an incorrect result.

Table 6: Overall Score

#	Tool	Score
1	α-β-CROWN	930.9
2	Marabou	594.1
3	PyRAT	585.5
4	NeuralSAT	547.0
5	nnenum	441.9
6	NNV	176.4
7	FastBATLLNN	100.0

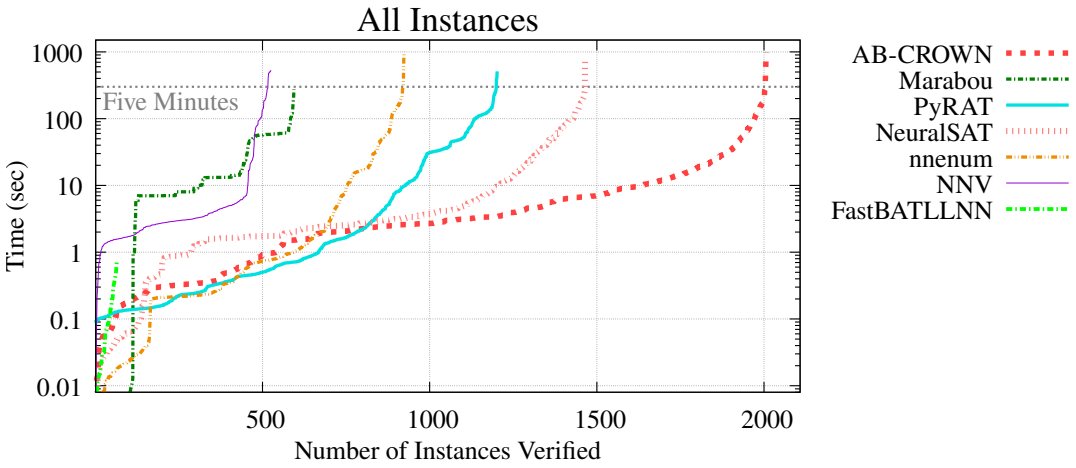


Figure 4: Cactus Plot for All Instances.

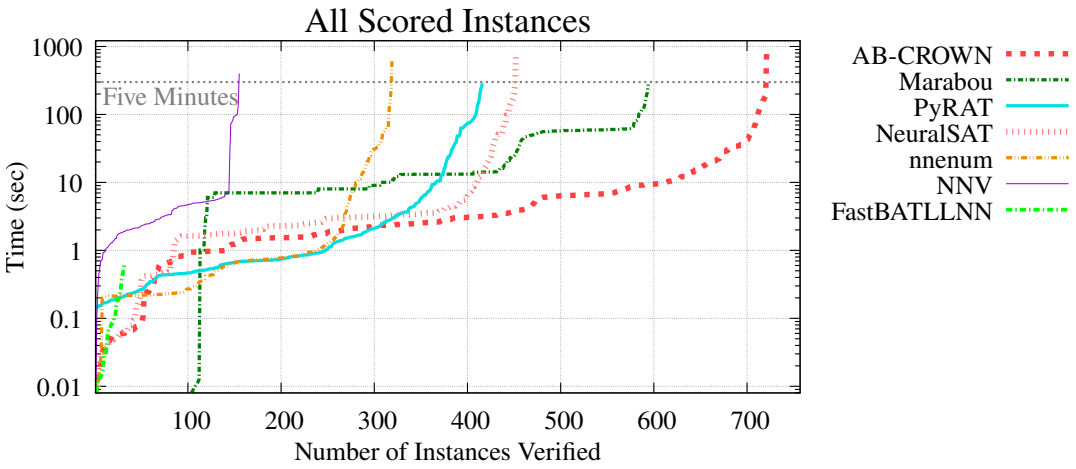


Figure 5: Cactus Plot for All Scored Instances.

5.1 Other Stats

This section presents other statistics related to the measurements that are interesting but did not play a direct role in scoring this year.

Table 7: Overhead

#	Tool	Seconds
1	FastBATLLNN	0.3
2	nnenum	0.9
3	Marabou	1.1
4	PyRAT	3.4
5	NeuralSAT	3.6
6	α-β-CROWN	5.7
7	NNV	15.9

Table 8: Num Benchmarks Participated

#	Tool	Count
1	α-β-CROWN	10
2	Marabou	9
3	PyRAT	8
4	NeuralSAT	8
5	nnenum	6
6	NNV	6
7	FastBATLLNN	1

Table 9: Num Instances Verified

#	Tool	Count
1	α-β-CROWN	721
2	Marabou	594
3	NeuralSAT	452
4	PyRAT	416
5	nnenum	319
6	NNV	155
7	FastBATLLNN	32

Table 10: Num SAT

#	Tool	Count
1	α-β-CROWN	151
2	Marabou	128
3	PyRAT	109
4	NeuralSAT	107
5	nnenum	101
6	NNV	60
7	FastBATLLNN	17

Table 11: Num UNSAT

#	Tool	Count
1	α-β-CROWN	570
2	Marabou	466
3	NeuralSAT	345
4	PyRAT	307
5	nnenum	218
6	NNV	95
7	FastBATLLNN	15

6 Conclusion and Ideas for Future Competitions

This report summarizes the 4$^{\text{rd}}$ Verification of Neural Networks Competition (VNN-COMP), held in 2023. While we observed a significant increase in the diversity, complexity, and scale of the proposed benchmarks, the best-performing tools seem to converge to GPU-enabled linear bound propagation methods using a branch-and-bound framework. In addition to the standardization of input formats (`onnx` and `vnnlib`) and evaluation hardware, introduced for VNN-COMP 2021, VNN-COMP 2023 also continued the standardized format for counter-examples and fully automated evaluation pipeline introduced in VNN-COMP 2022, requiring authors to provide complete installation scripts. We hope that this increased standardization and automatization does not only simplify the evaluation during the competition but also enables practitioners and researchers to more easily apply a range of state-of-the-art verification methods to their individual problems.

VNN-COMP 2023, successfully implemented a range of improvement opportunities identified during the previous iteration. These included requiring witnesses of found counter-examples to disambiguate tool disagreement, increasing automatization to enable a smoother final evaluation, and making a broader range of AWS instances available to allow for a better fit with tools' requirements. Further ideas for future competitions include the use of scored benchmarks specifically designed for year-on-year progress tracking, the reduction of tool tuning, a batch-processing mode, and more rigorous soundness evaluation.

Table 12: Incorrect Results (or Missing CE)

#	Tool	Count
1	NeuralSAT[a]	35
2	NNV	35
3	α-β-CROWN	1

[a]All penalties were due to incorrectly parsed constraints in the nn4sys benchmark.

Acknowledgements

This research was supported in part by the Air Force Research Laboratory Information Directorate, through the Air Force Office of Scientific Research Summer Faculty Fellowship Program, Contract Numbers FA8750-15-3-6003, FA9550-15-0001 and FA9550-20-F-0005. This material is based upon work supported by the Air Force Office of Scientific Research under award numbers FA9550-19-1-0288, FA9550-21-1-0121, and FA9550-22-1-0019, the National Science Foundation (NSF) under grant numbers 1918450, 1911017, 2028001, 2107035, 2220401, 2220418, 2220426, and 2238133, and the Defense Advanced Research Projects Agency (DARPA) Assured Autonomy and Assured Neuro Symbolic Learning and Reasoning (ANSR) programs through contract numbers FA8750-18-C-0089 and FA8750-23-C-0518. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force, DARPA, nor NSF.

Tool and benchmark authors listed in Section 3 and Section 4 participated in the preparation and review of this report.

References

[1] Belgian Traffic Sign Database. https://www.kaggle.com/datasets/shazaelmorsh/trafficsigns. Accessed: March 25th, 2023.

[2] Chinese Traffic Sign Database. https://www.kaggle.com/datasets/dmitryyemelyanov/chinese-traffic-signs. Accessed: March 25th, 2023.

[3] German Traffic Sign Recognition Benchmark. https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?datasetId=82373&language=Python. Accessed: March 25th, 2023.

[4] PyRAT Analyzer website. https://pyrat-analyzer.com/. Accessed: December 15th, 2023.

[5] Stanley Bak. Execution-guided overapproximation (ego) for improving scalability of neural network verification, 2020.

[6] Stanley Bak. nnenum: Verification of relu neural networks with optimized abstraction refinement. In *NASA Formal Methods Symposium*, pages 19–36. Springer, 2021.

[7] Stanley Bak, Hoang-Dung Tran, Kerianne Hobbs, and Taylor T. Johnson. Improved geometric path enumeration for verifying ReLU neural networks. In *32nd International Conference on Computer-Aided Verification (CAV)*, July 2020.

[8] Rudy Bunel, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 2018.

[9] Stefano Demarchi, Dario Guidotti, Luca Pulina, and Armando Tacchella. Supporting standardization of neural networks verification with vnnlib and coconet. In Nina Narodytska, Guy Amir, Guy Katz, and Omri Isac, editors, *Proceedings of the 6th Workshop on Formal Methods for ML-Enabled Autonomous Systems*, volume 16 of *Kalpa Publications in Computing*, pages 47–58. EasyChair, 2023.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[11] Hai Duong, Linhan Li, ThanhVu Nguyen, and Matthew Dwyer. A DPLL(T) Framework for Verifying Deep Neural Networks, 2023. arXiv, 25 pages.

[12] Serge Durand, Augustin Lemesle, Zakaria Chihani, Caterina Urban, and François Terrier. Reciph: Relational coefficients for input partitioning heuristic. In *1st Workshop on Formal Verification of Machine Learning (WFVML 2022)*, 2022.

[13] EASA and Collins Aerospace. Formal Methods use for Learning Assurance (ForMuLA). Technical report, April 2023.

[14] James Ferlez, Haitham Khedr, and Yasser Shoukry. Fast BATLLNN: fast box analysis of two-level lattice neural networks. In Ezio Bartocci and Sylvie Putot, editors, *HSCC '22: 25th ACM International Conference on Hybrid Systems: Computation and Control, Milan, Italy, May 4 - 6, 2022*, pages 23:1–23:11. ACM, 2022.

[15] Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

[16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. *Advances in Neural Information Processing Systems*, 29, 2016.

[17] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[18] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth

Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljic, David L. Dill, Mykel J. Kochenderfer, and Clark W. Barrett. The marabou framework for verification and analysis of deep neural networks. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, volume 11561 of *Lecture Notes in Computer Science*, pages 443–452. Springer, 2019.

[19] Guy Katz, Derek A Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljić, et al. The marabou framework for verification and analysis of deep neural networks. In *International Conference on Computer Aided Verification*, pages 443–452. Springer, 2019.

[20] Dmitrii Kirov and Simone Fulvio Rollini. Benchmark: remaining useful life predictor for aircraft equipment. In *International Conference on Bridging the Gap between AI and Reality*, pages 299–304. Springer, 2023.

[21] Dmitrii Kirov, Simone Fulvio Rollini, Luigi Di Guglielmo, and Darren Cofer. Formal verification of a neural network based prognostics system for aircraft equipment. In *International Conference on Bridging the Gap between AI and Reality*, pages 225–240. Springer, 2023.

[22] Suhas Kotha, Christopher Brix, Zico Kolter, Huan Zhang, et al. Provably bounding neural network preimages. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[23] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, 2018.

[24] Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T. Johnson. NNV 2.0: The neural network verification tool. In *35th International Conference on Computer-Aided Verification (CAV)*, July 2023.

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[26] Diego Manzanas Lopez, Patrick Musau, Nathaniel Hamilton, and Taylor Johnson. Reachability analysis of a general class of neural ordinary differential equation. In *Proceedings of the 20th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2022), Co-Located with CONCUR, FMICS, and QEST as part of CONFEST 2022.*, Warsaw, Poland, September 2022.

[27] Andreea Postovan and Mădălina Eraşcu. Architecturing binarized neural networks for traffic sign recognition. *arXiv preprint arXiv:2303.15005*, 2023.

[28] Zhouxing Shi, Qirui Jin, Huan Zhang, Zico Kolter, Suman Jana, and Cho-Jui Hsieh. Formal verification for neural networks with general nonlinearities via branch-and-bound. In *2nd Workshop on Formal Verification of Machine Learning (WFVML 2023)*, 2023.

[29] Zhouxing Shi, Yihan Wang, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Fast certified robust training with short warmup. *Advances in Neural Information Processing Systems*, 34:18335–18349, 2021.

[30] Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers. In *International Conference on Learning Representations*, 2019.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL):41:1–41:30, 2019.

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. *arXiv preprint arXiv:1312.6199*, 2013.

[34] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *ICLR*, 2019.

[35] H. D. Tran, N. Pal, D. Lopez, P. Musau, X. Yang, W. Xiang L. Nguyen, S. Bak, , and T. T. Johnson. Verification of piecewise deep neural networks: A star set approach with zonotope pre-filter. *Formal aspects of computing*, 2021.

[36] Hoang-Dung Tran, Stanley Bak, Weiming Xiang, and Taylor T. Johnson. Verification of deep convolutional neural networks using imagestars. In *32nd International Conference on Computer-Aided Verification (CAV)*. Springer, July 2020.

[37] Hoang-Dung Tran, Feiyang Cei, Diego Manzanas Lopez, Taylor T. Johnson, and Xenofon Koutsoukos. Safety verification of cyber-physical systems with reinforcement learning control. In *ACM SIGBED International Conference on Embedded Software (EMSOFT'19)*. ACM, October 2019.

[38] Hoang Dung Tran, SungWoo Choi, Tomoya Yamaguchi, Bardh Hoxha, and Danil Prokhorov. Verification of recurrent neural networks using star reachability. In *The 26th ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, May 2023.

[39] Hoang-Dung Tran, Patrick Musau, Diego Manzanas Lopez, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. Star-based reachability analysis for deep neural networks. In *23rd International Symposium on Formal Methods (FM'19)*. Springer International Publishing, October 2019.

[40] Hoang-Dung Tran, Neelanjana Pal, Patrick Musau, Xiaodong Yang, Nathaniel P. Hamilton, Diego Manzanas Lopez, Stanley Bak, and Taylor T. Johnson. Robustness verification of semantic segmentation neural networks using relaxed reachability. In *33rd International Conference on Computer-Aided Verification (CAV)*. Springer, July 2021.

[41] Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *32nd International Conference on Computer-Aided Verification (CAV)*, July 2020.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[43] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021.

[44] Dennis Wei, Haoze Wu, Min Wu, Pin-Yu Chen, Clark Barrett, and Eitan Farchi. Convex bounds on the softmax function with applications to robustness verification. In *International Conference on Artificial Intelligence and Statistics*, pages 6853–6878. PMLR, 2023.

[45] Haoze Wu, Clark Barrett, Mahmood Sharif, Nina Narodytska, and Gagandeep Singh. Scalable verification of gnn-based job schedulers. 6(OOPSLA2), oct 2022.

[46] Haoze Wu, Alex Ozdemir, Aleksandar Zeljic, Kyle Julian, Ahmed Irfan, Divya Gopinath, Sadjad Fouladi, Guy Katz, Corina Pasareanu, and Clark Barrett. Parallelization techniques for verifying neural networks. In *# PLACEHOLDER_PARENT_METADATA_VALUE#*, volume 1, pages 128–137. TU Wien Academic Press, 2020.

[47] Haoze Wu, Teruhiro Tagomori, Alexander Robey, Fengjun Yang, Nikolai Matni, George Pappas, Hamed Hassani, Corina Pasareanu, and Clark Barrett. Toward certified robustness against real-world distribution shifts. *arXiv preprint arXiv:2206.03669*, 2022.

[48] Haoze Wu, Aleksandar Zeljić, Guy Katz, and Clark Barrett. Efficient neural network analysis with sum-of-infeasibilities. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 143–163. Springer, 2022.

[49] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33, 2020.

[50] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and Complete: Enabling complete neural network verification with rapid and massively parallel

incomplete verifiers. In *International Conference on Learning Representations*, 2021.

[51] Huan Zhang*, Shiqi Wang*, Kaidi Xu*, Linyi Li, Bo Li, Suman Jana, Cho-Jui Hsieh, and J Zico Kolter. General cutting planes for bound-propagation-based neural network verification. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[52] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in Neural Information Processing Systems*, 31:4939–4948, 2018.

[53] Jianming Zhang, Wei Wang, Chaoquan Lu, Jin Wang, and Arun Kumar Sangaiah. Lightweight Deep Network for Traffic Sign Classification. *Annals of Telecommunications*, 75:369–379, 2020.

A Extended Results

In this section, we provide more fine-grained results.

A.1 Scored Benchmarks

Table 13: Benchmark `2023-acasxu`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	nnenum	139	47	0	0	1860	100.0%
2	NeuralSAT	138	46	0	0	1840	98.9%
3	PyRAT	137	45	0	0	1820	97.8%
4	Marabou	135	46	0	0	1810	97.3%
5	α-β-CROWN	139	46	0	1	1700	91.4%
6	NNV	68	27	0	0	950	51.1%

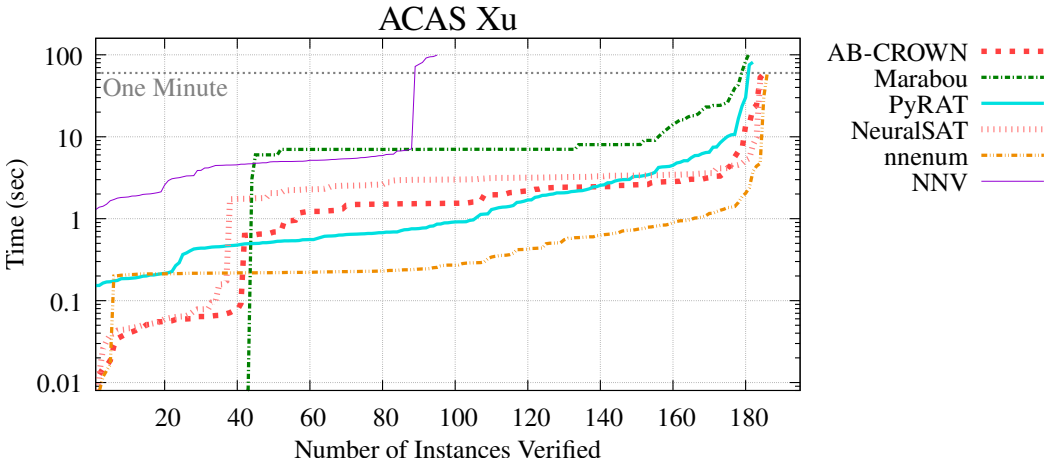


Figure 6: Cactus Plot for ACAS Xu.

Table 14: Benchmark `2023-cgan`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	8	13	0	0	210	100.0%
2	PyRAT	8	11	0	0	190	90.5%
3	nnenum	6	11	0	0	170	81.0%
4	NeuralSAT	3	11	0	0	140	66.7%
5	NNV	3	11	0	0	140	66.7%
6	Marabou	0	13	0	0	130	61.9%

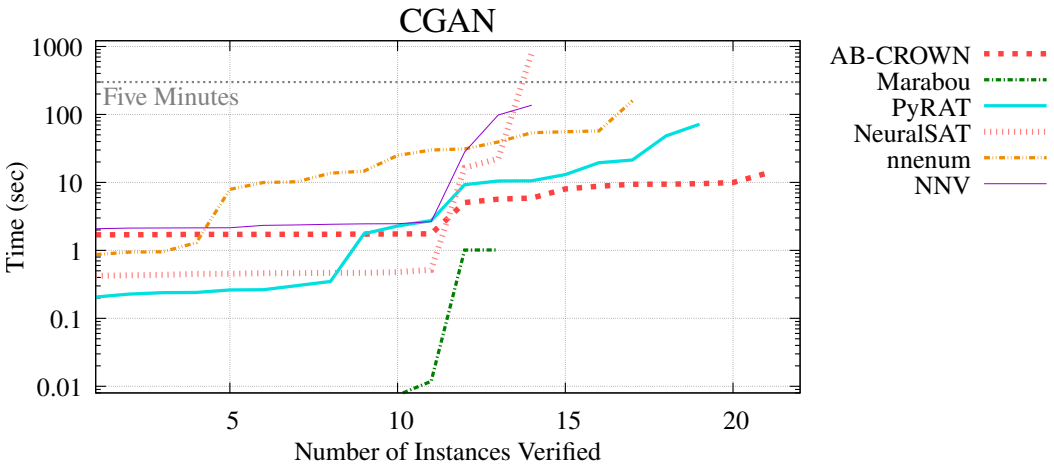


Figure 7: Cactus Plot for CGAN.

Table 15: Benchmark `2023-collins-rul-cnn`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	nnenum	35	27	0	0	620	100.0%
2	NeuralSAT	35	27	0	0	620	100.0%
3	Marabou	35	27	0	0	620	100.0%
4	α-β-CROWN	35	27	0	0	620	100.0%
5	PyRAT	33	27	0	0	600	96.8%
6	NNV	23	0	0	27	-3820	0%

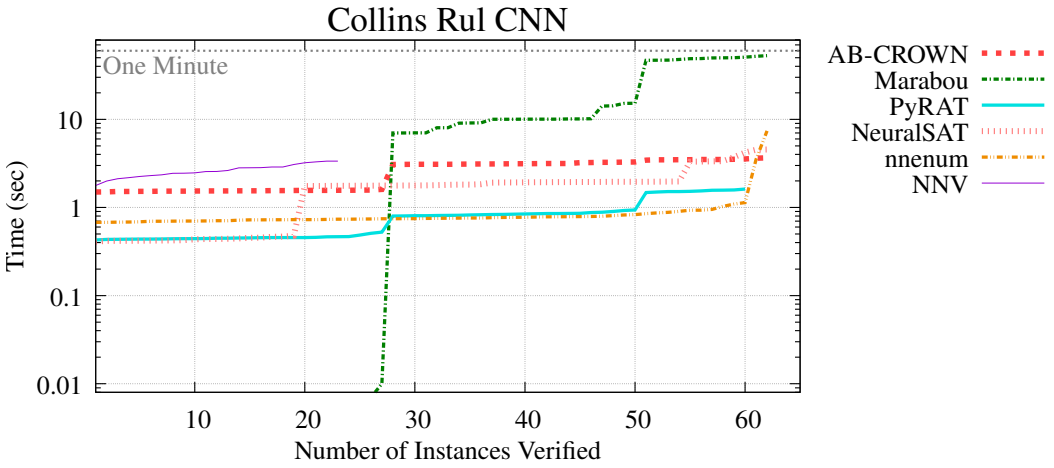


Figure 8: Cactus Plot for Collins Rul CNN.

Table 16: Benchmark `2023-dist-shift`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	67	5	0	0	720	100.0%
2	PyRAT	66	5	0	0	710	98.6%
3	NeuralSAT	66	5	0	0	710	98.6%
4	Marabou	55	5	0	0	600	83.3%
5	NNV	0	4	0	0	40	5.6%

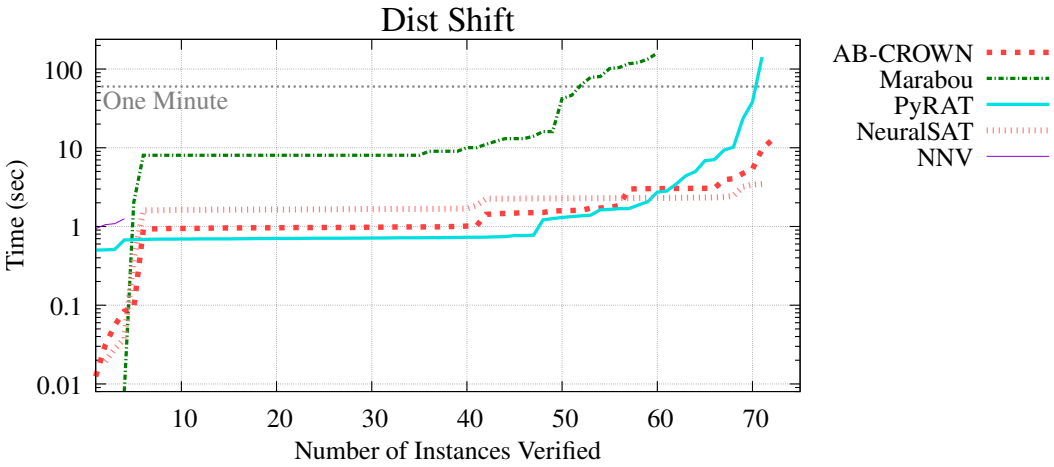


Figure 9: Cactus Plot for Dist Shift.

Table 17: Benchmark `2023-ml4acopf`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	18	1	0	0	190	100.0%

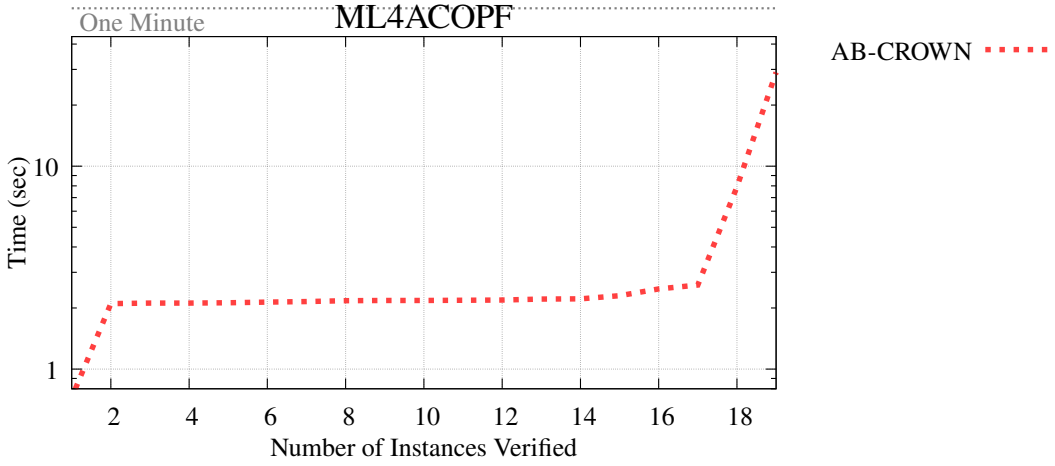


Figure 10: Cactus Plot for ML4ACOPF.

Table 18: Benchmark `2023-nn4sys`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	194	0	0	0	1940	100.0%
2	NeuralSAT	83	0	0	0	830	42.8%
3	PyRAT	40	0	0	0	400	20.6%
4	Marabou	34	0	0	0	340	17.5%
5	nnenum	22	0	0	0	220	11.3%
6	NNV	0	2	0	8	-1180	0%

Table 19: Benchmark `2023-tllverifybench`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	NeuralSAT	15	17	0	0	320	100.0%
2	FastBATLLNN	15	17	0	0	320	100.0%
3	α-β-CROWN	15	17	0	0	320	100.0%
4	PyRAT	10	12	0	0	220	68.8%
5	Marabou	5	17	0	0	220	68.8%
6	nnenum	2	16	0	0	180	56.2%
7	NNV	1	16	0	0	170	53.1%

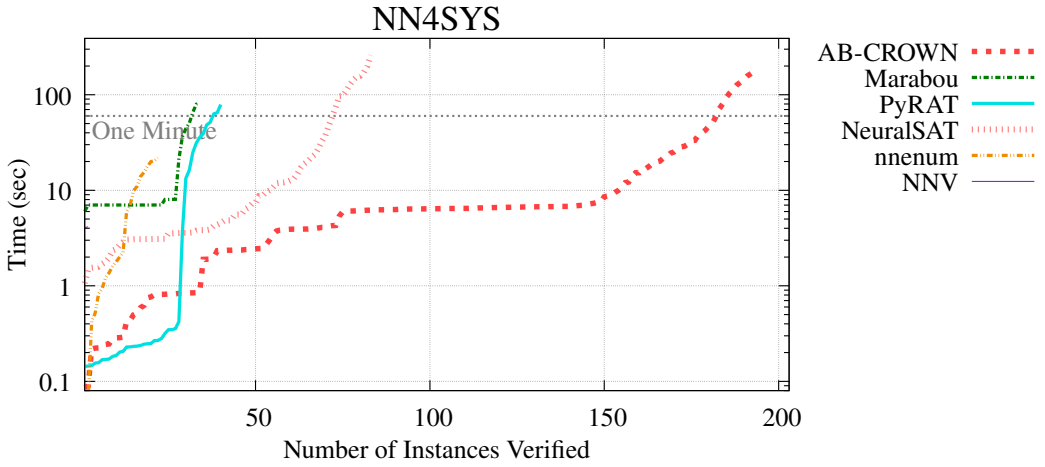


Figure 11: Cactus Plot for NN4SYS.

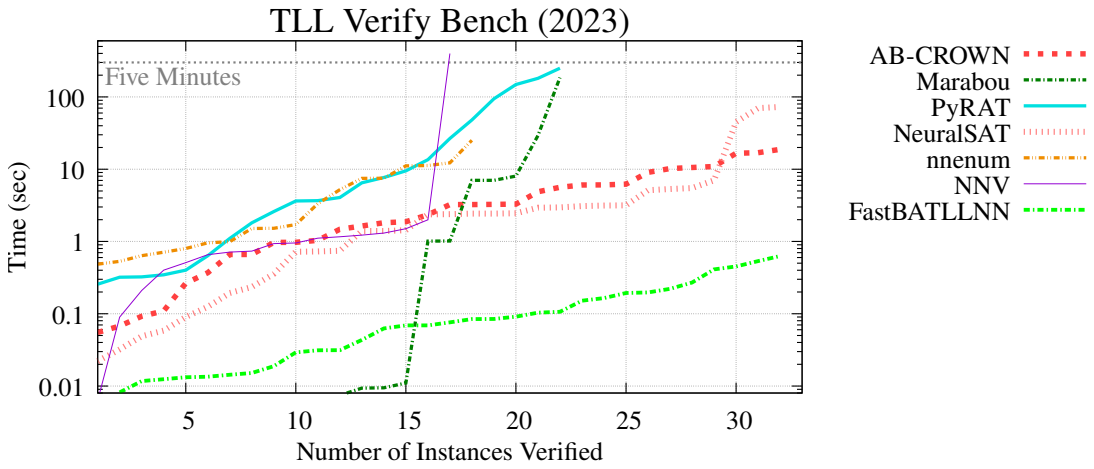


Figure 12: Cactus Plot for TLL Verify Bench (2023).

Table 20: Benchmark `2023-traffic-signs-recognition`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	0	42	0	0	420	100.0%
2	Marabou	0	19	0	0	190	45.2%
3	PyRAT	0	8	0	0	80	19.0%
4	NeuralSAT	0	0	0	35	-5250	0%

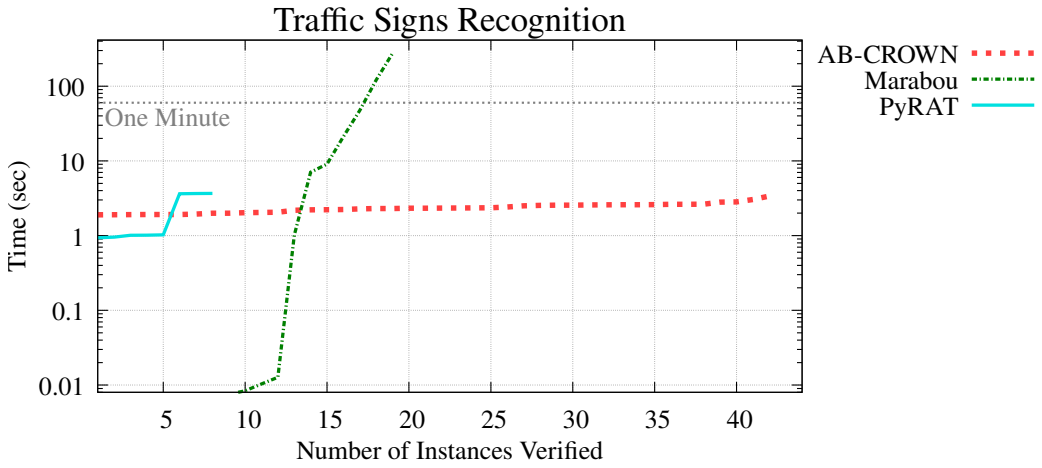


Figure 13: Cactus Plot for Traffic Signs Recognition.

Table 21: Benchmark `2023-vggnet16`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	15	0	0	0	150	100.0%
2	nnenum	14	0	0	0	140	93.3%
3	PyRAT	13	1	0	0	140	93.3%
4	NeuralSAT	5	1	0	0	60	40.0%
5	Marabou	2	1	0	0	30	20.0%

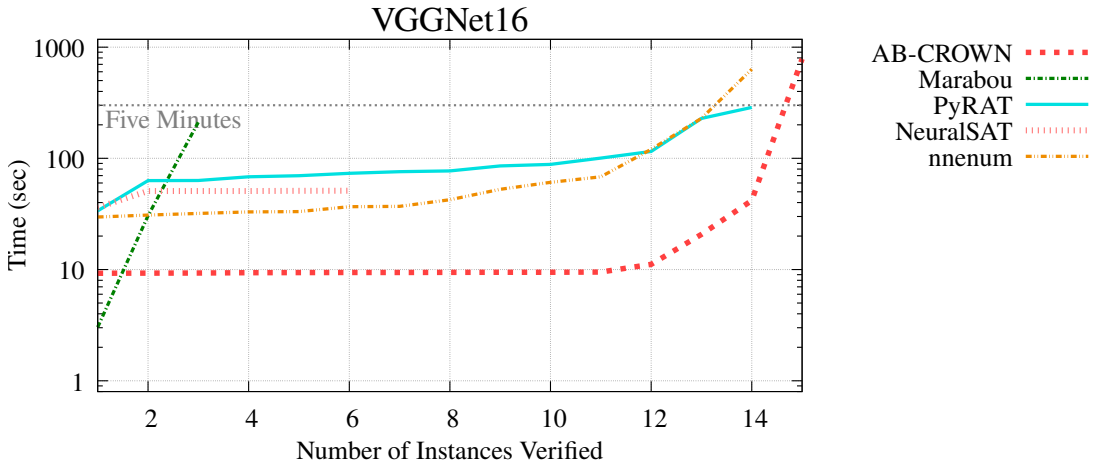


Figure 14: Cactus Plot for VGGNet16.

Table 22: Benchmark `2023-vit`. Note: A soundness issue was found after the competition in Marabou's results, due to an issue in Gurobi; these results may be updated in a future version of the report.

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	Marabou	200	0	0	0	2000	100.0%
2	α-β-CROWN	79	0	0	0	790	39.5%

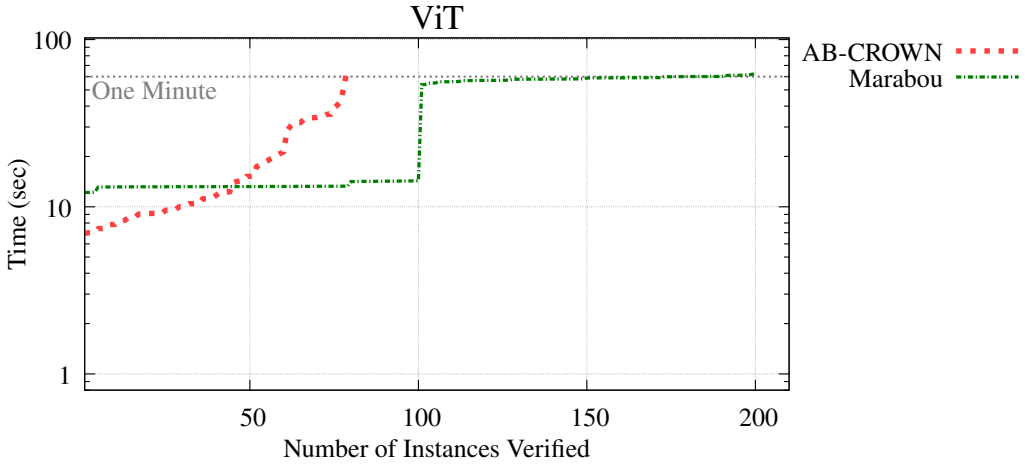


Figure 15: Cactus Plot for ViT.

A.2 Unscored Benchmarks

Table 23: Benchmark `2022-carvana-unet-2022`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	39	0	0	0	390	100.0%

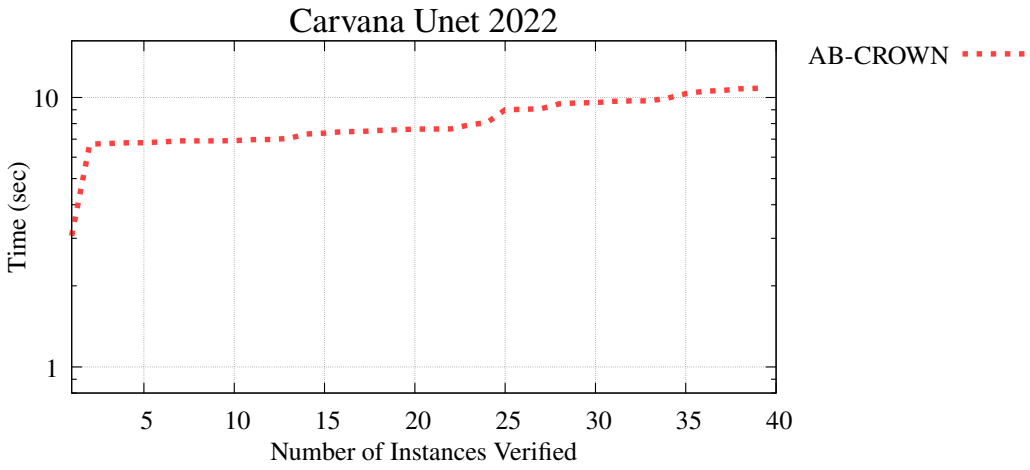


Figure 16: Cactus Plot for Carvana Unet 2022.

Table 24: Benchmark `2022-cifar100-tinyimagenet-resnet`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	69	3	0	0	720	100.0%
2	NeuralSAT	56	3	0	0	590	81.9%

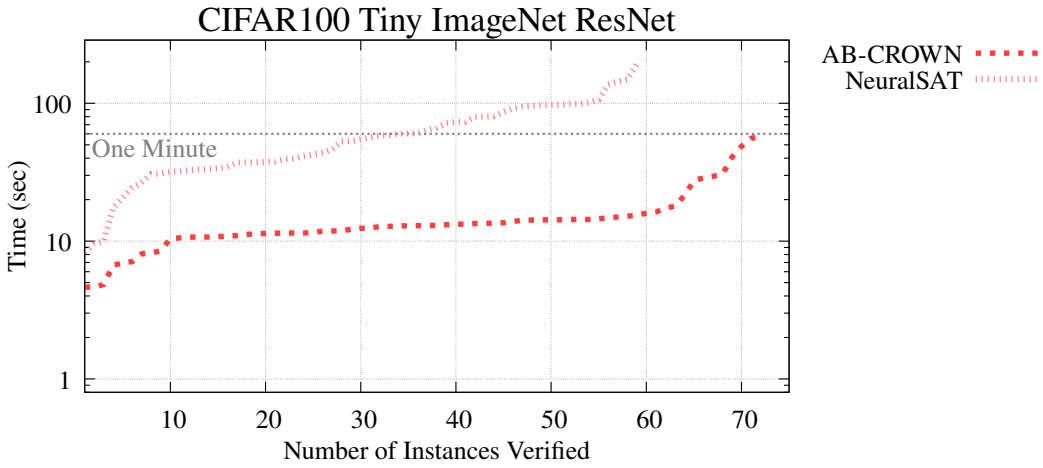


Figure 17: Cactus Plot for CIFAR100 Tiny ImageNet ResNet.

Table 25: Benchmark `2022-cifar2020`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	NeuralSAT	148	42	0	0	1900	100.0%
2	α-β-CROWN	148	41	0	0	1890	99.5%
3	PyRAT	96	39	0	0	1350	71.1%
4	nnenum	66	19	0	0	850	44.7%
5	NNV	32	0	0	6	-580	0%

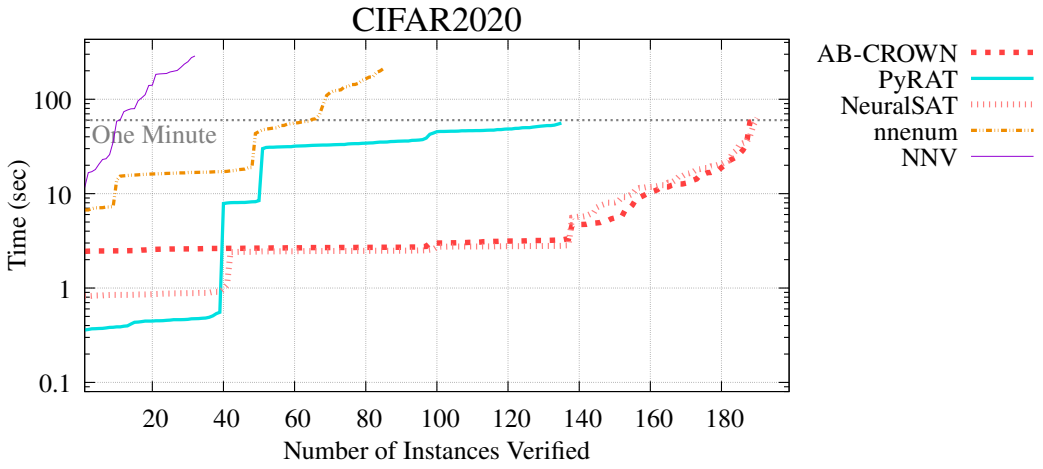


Figure 18: Cactus Plot for CIFAR2020.

Table 26: Benchmark `2022-cifar-biasfield`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	68	1	0	0	690	100.0%
2	NeuralSAT	67	1	0	0	680	98.6%
3	PyRAT	51	1	0	0	520	75.4%
4	nnenum	18	0	0	0	180	26.1%

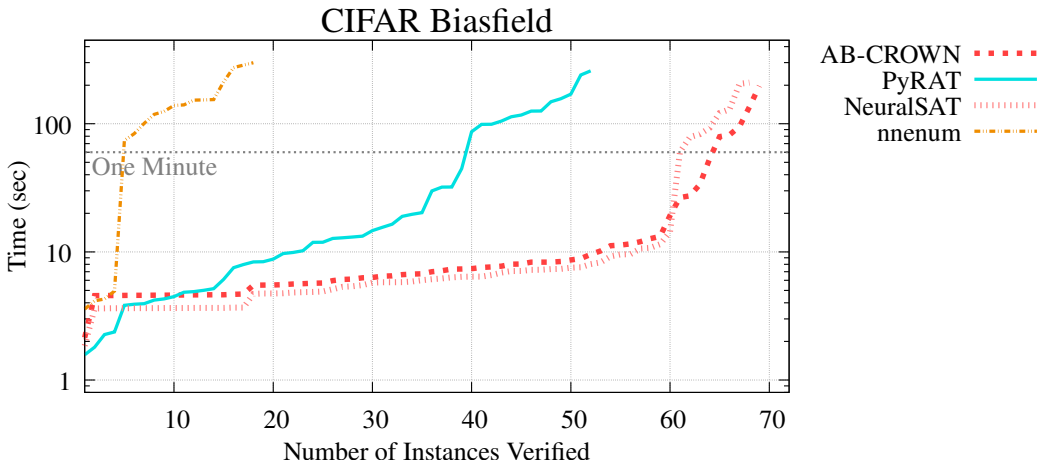


Figure 19: Cactus Plot for CIFAR Biasfield.

Table 27: Benchmark `2022-mnist-fc`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	NeuralSAT	46	18	0	0	640	100.0%
2	nnenum	48	11	0	0	590	92.2%
3	α-β-CROWN	68	16	0	2	540	84.4%
4	PyRAT	35	16	0	0	510	79.7%
5	NNV	30	4	0	0	340	53.1%

Table 28: Benchmark `2022-nn4sys`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	152	0	0	0	1520	100.0%
2	NeuralSAT	78	0	0	0	780	51.3%
3	nnenum	22	0	0	0	220	14.5%
4	PyRAT	14	0	0	0	140	9.2%
5	NNV	0	2	0	8	-1180	0%

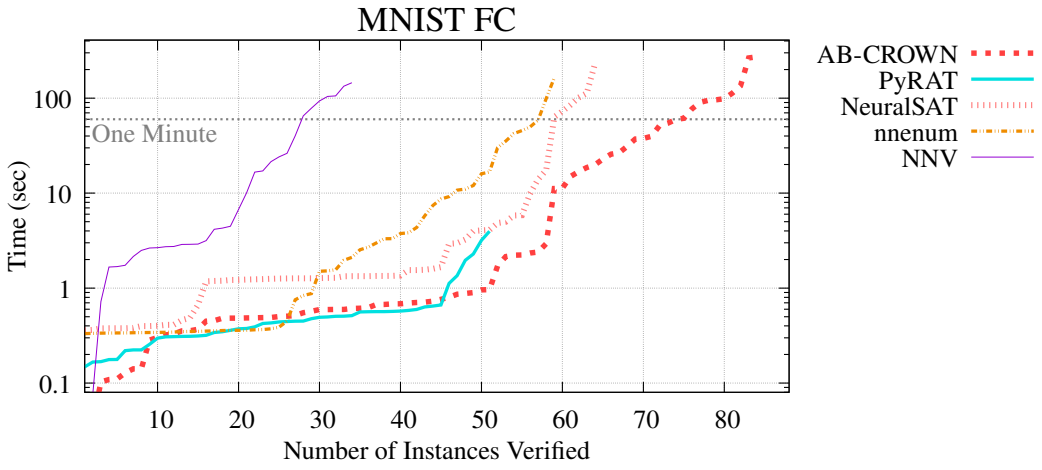


Figure 20: Cactus Plot for MNIST FC.

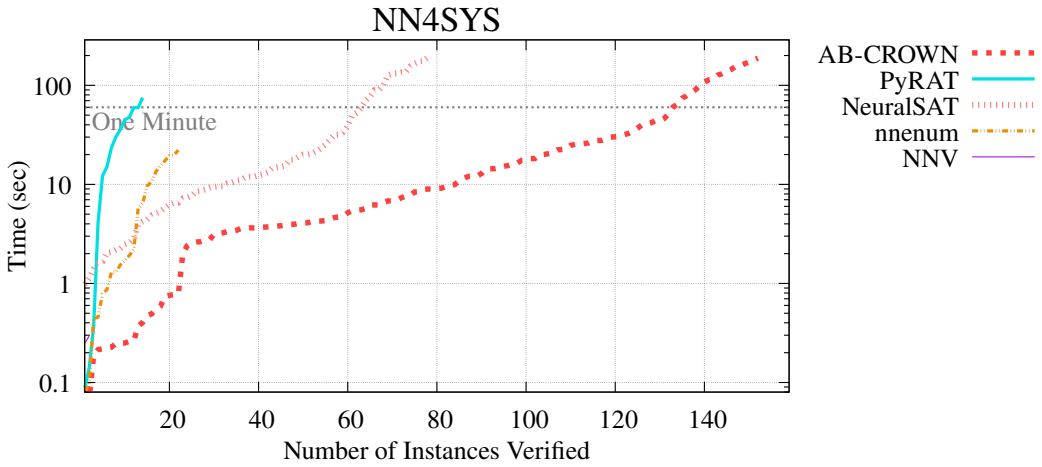


Figure 21: Cactus Plot for NN4SYS.

Table 29: Benchmark `2022-oval21`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	25	1	0	0	260	100.0%
2	NeuralSAT	19	1	0	0	200	76.9%
3	nnenum	3	1	0	0	40	15.4%
4	NNV	4	0	0	0	40	15.4%
5	PyRAT	1	1	0	0	20	7.7%

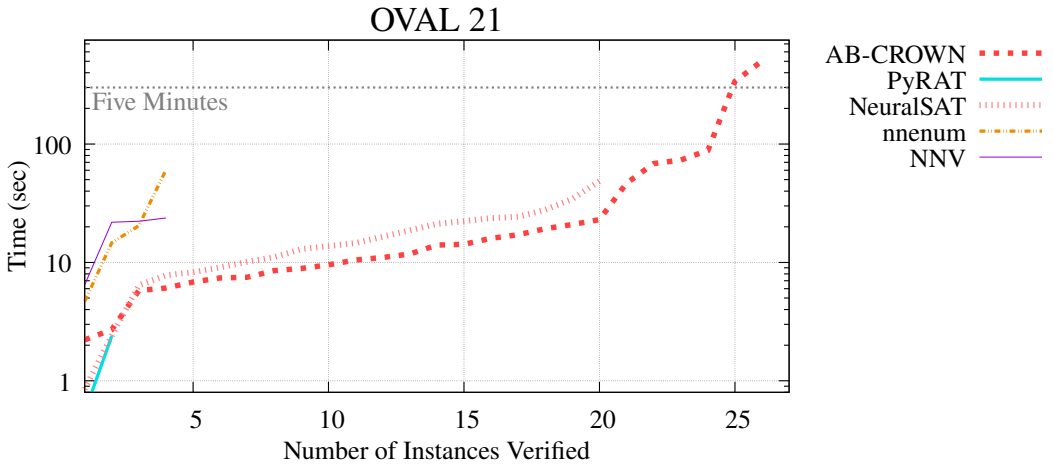


Figure 22: Cactus Plot for OVAL 21.

Table 30: Benchmark `2022-reach-prob-density`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	nnenum	22	14	0	0	360	100.0%
2	α-β-CROWN	22	14	0	0	360	100.0%
3	NeuralSAT	22	12	0	0	340	94.4%
4	PyRAT	17	7	0	0	240	66.7%
5	NNV	9	0	0	0	90	25.0%

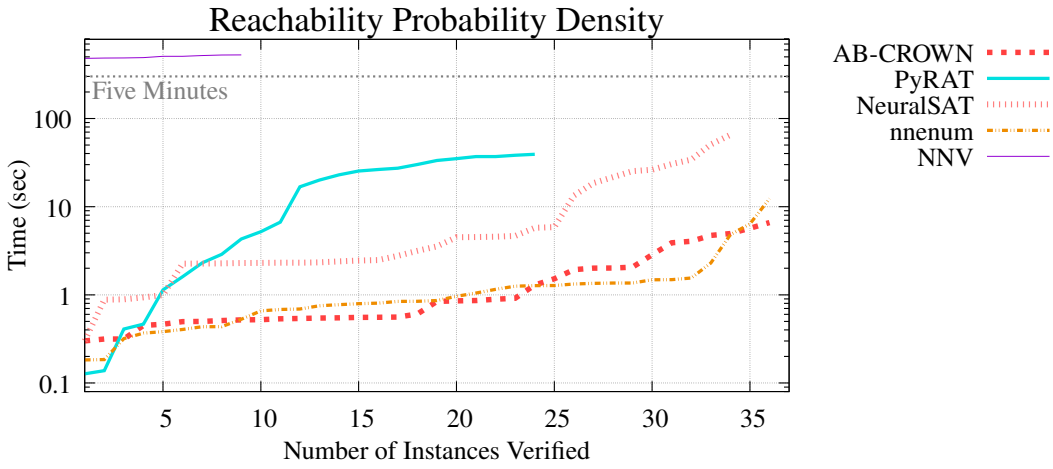


Figure 23: Cactus Plot for Reachability Probability Density.

Table 31: Benchmark `2022-rl-benchmarks`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	NeuralSAT	193	103	0	0	2960	100.0%
2	α-β-CROWN	193	103	0	0	2960	100.0%
3	nnenum	191	103	0	0	2940	99.3%
4	PyRAT	194	96	0	0	2900	98.0%
5	NNV	177	90	0	0	2670	90.2%

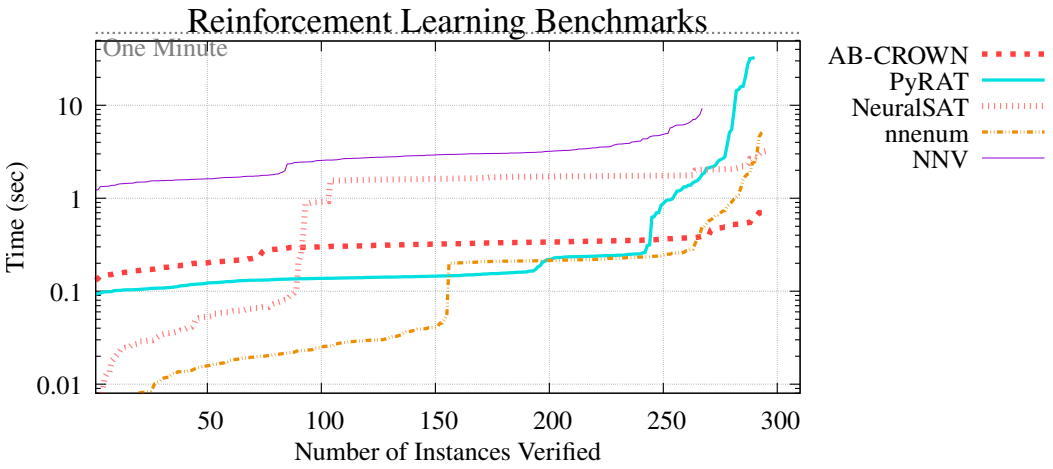


Figure 24: Cactus Plot for Reinforcement Learning Benchmarks.

Table 32: Benchmark `2022-sri-resnet-a`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	20	12	0	0	320	100.0%
2	NeuralSAT	18	12	0	0	300	93.8%
3	PyRAT	5	11	0	0	160	50.0%

Table 33: Benchmark `2022-sri-resnet-b`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	NeuralSAT	28	11	0	0	390	100.0%
2	α-β-CROWN	28	11	0	0	390	100.0%
3	PyRAT	13	11	0	0	240	61.5%
4	NNV	2	0	0	0	20	5.1%

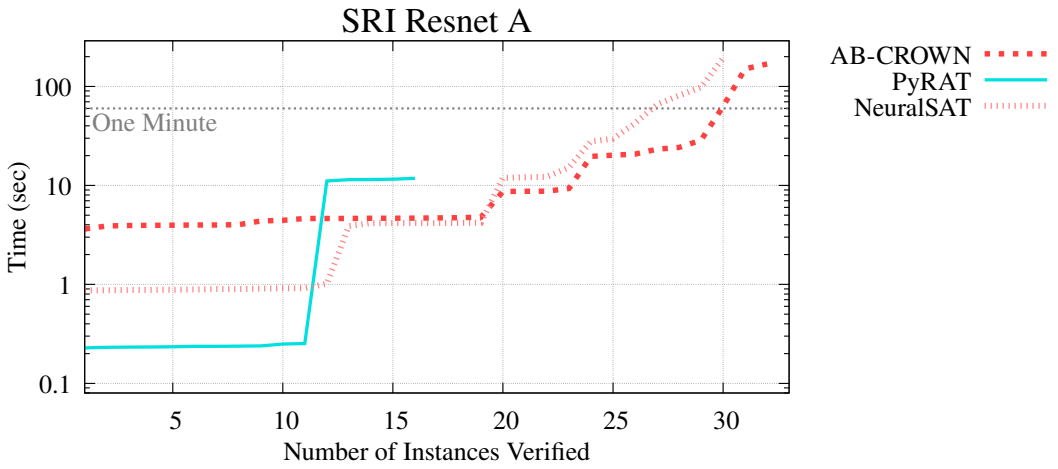


Figure 25: Cactus Plot for SRI Resnet A.

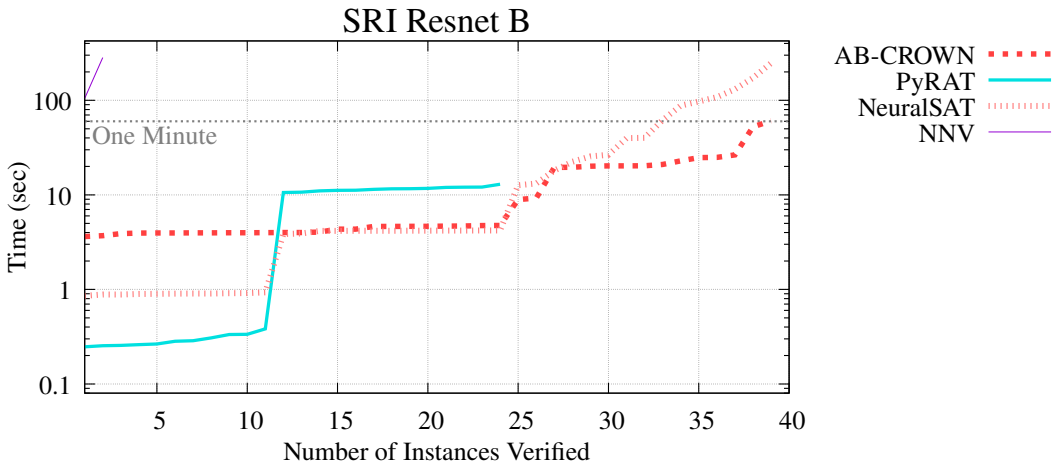


Figure 26: Cactus Plot for SRI Resnet B.

Table 34: Benchmark `2022-tllverifybench`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	NeuralSAT	11	21	0	0	320	100.0%
2	FastBATLLNN	11	21	0	0	320	100.0%
3	α-β-CROWN	11	21	0	0	320	100.0%
4	PyRAT	9	17	0	0	260	81.2%
5	nnenum	1	21	0	0	220	68.8%
6	NNV	0	21	0	0	210	65.6%

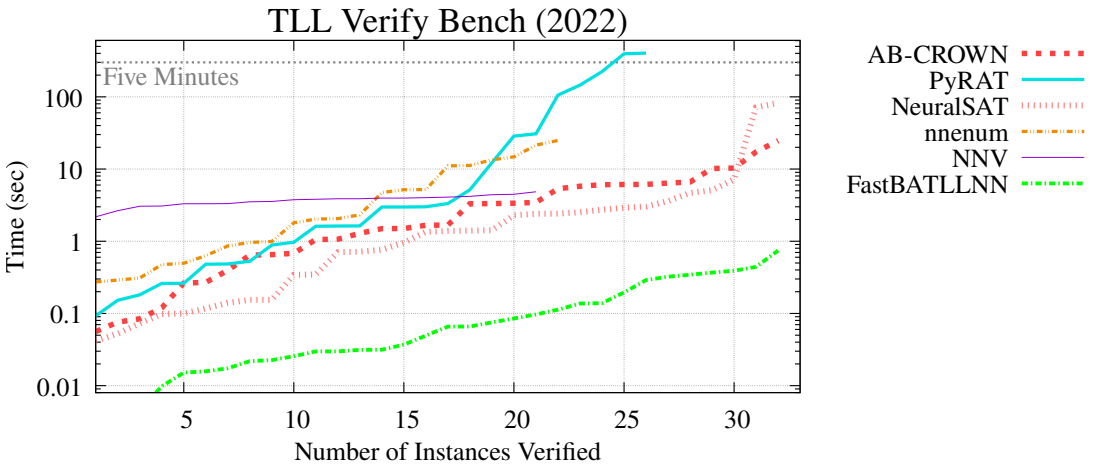


Figure 27: Cactus Plot for TLL Verify Bench (2022).

Table 35: Benchmark `2022-vggnet16-2022`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	nnenum	11	1	0	0	120	100.0%
2	PyRAT	9	1	0	0	100	83.3%
3	NeuralSAT	5	1	0	0	60	50.0%
4	α-β-CROWN	14	0	0	1	-10	0%

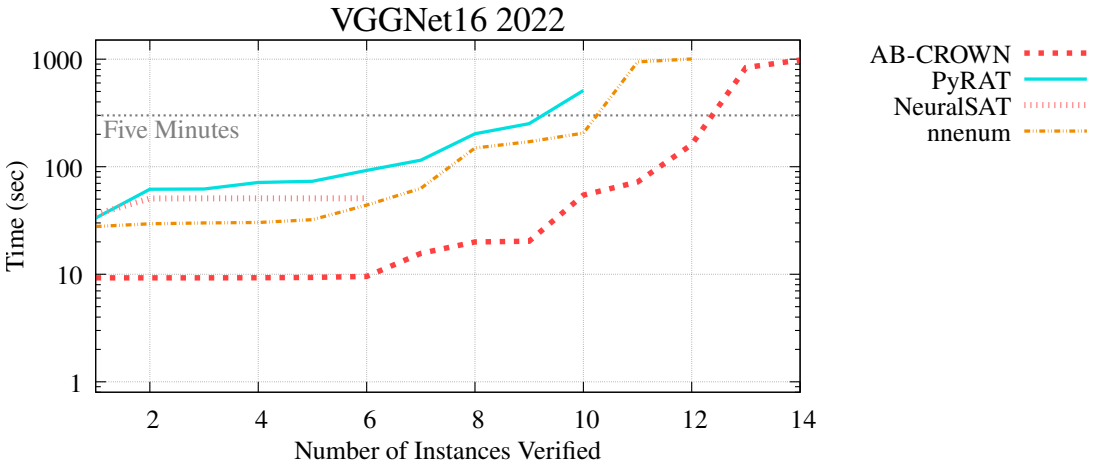


Figure 28: Cactus Plot for VGGNet16 2022.

Table 36: Benchmark `2023-cctsdb-yolo`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	11	28	0	0	390	100.0%

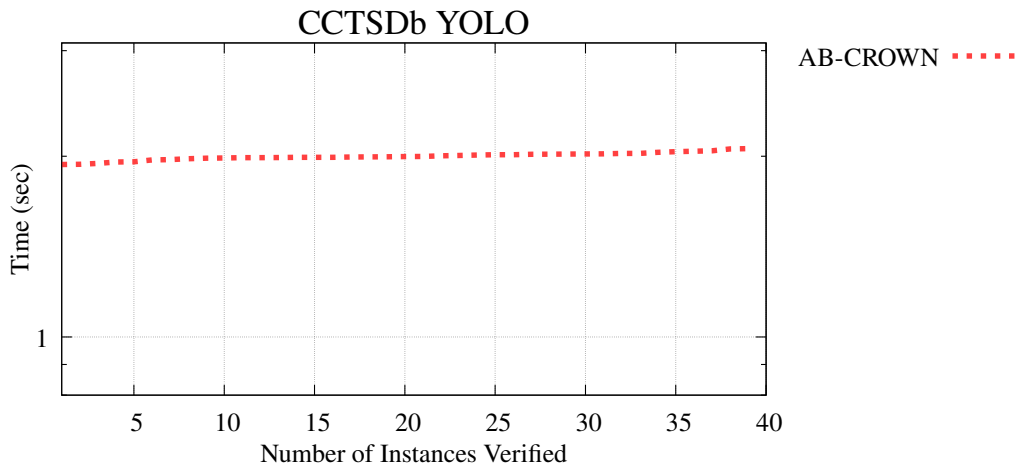


Figure 29: Cactus Plot for CCTSDb YOLO.

Table 37: Benchmark `2023-collins-yolo-robustness`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	PyRAT	0	6	0	0	60	100.0%
2	α-β-CROWN	0	6	0	0	60	100.0%

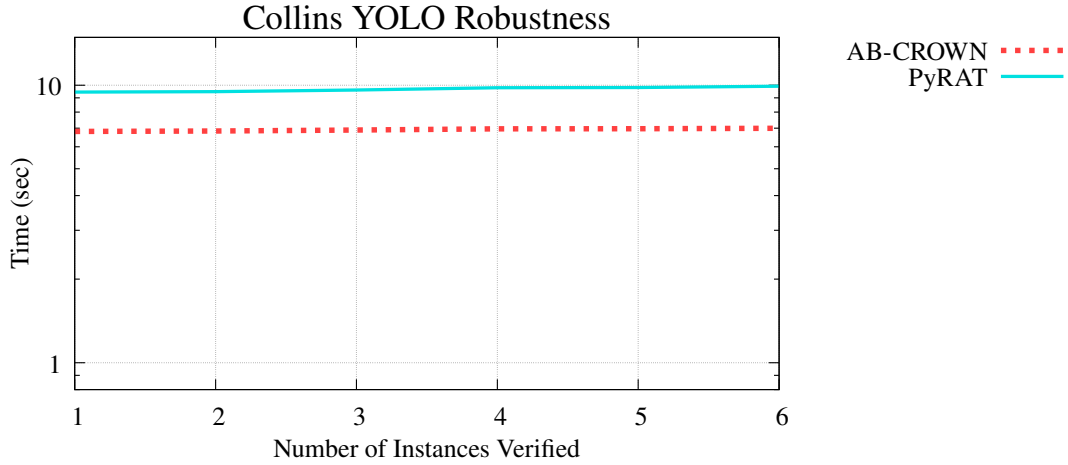


Figure 30: Cactus Plot for Collins YOLO Robustness.

Table 38: Benchmark `2023-metaroom`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	94	5	0	0	990	100.0%
2	NeuralSAT	92	5	0	0	970	98.0%
3	PyRAT	93	2	0	0	950	96.0%
4	nnenum	51	2	0	0	530	53.5%

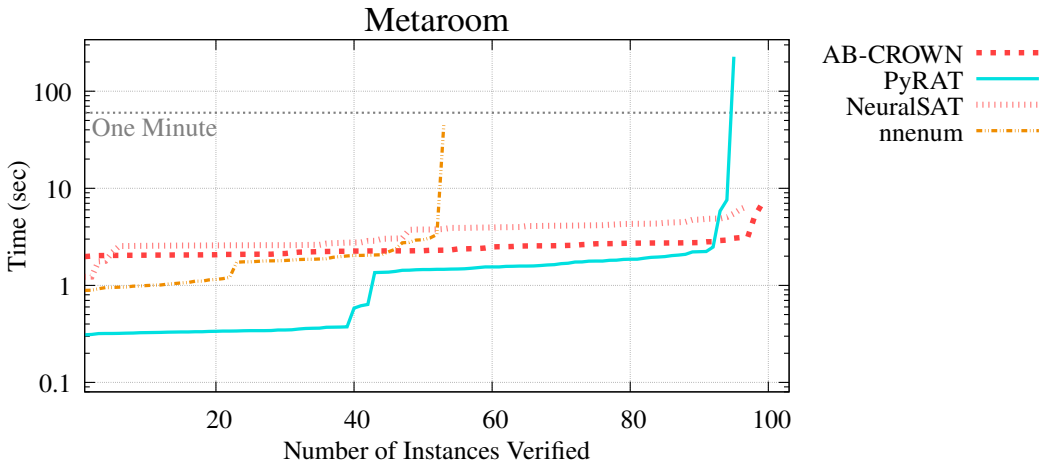


Figure 31: Cactus Plot for Metaroom.

Table 39: Benchmark `2023-yolo`

#	Tool	Verified	Falsified	Fastest	Penalty	Score	Percent
1	α-β-CROWN	63	0	0	0	630	100.0%
2	PyRAT	41	0	0	0	410	65.1%
3	NNV	0	0	0	41	-6150	0%

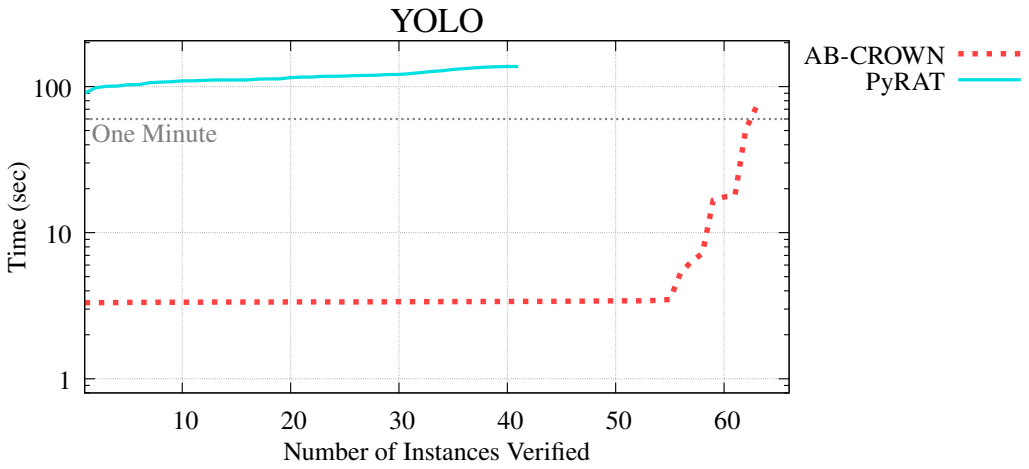


Figure 32: Cactus Plot for YOLO.

B Alternative Scoring Results

As described in Section 2, tools had to provide a concrete counterexample if they report SAT. These were supposed to contain both the network input and the corresponding output. However, as several tools provided either no outputs or incorrect outputs, we decided to discard all outputs in the provided counterexamples. Instead, the outputs were computed using the onnxruntime package.

Alternatively, tools could have been assigned a penalty if the network output for the counterexample was missing or incorrect. This would have lead to the following ranking:

Table 40: Overall Score

#	**Tool**	**Score**
1	α-β-CROWN	830.9
2	Marabou	531.6
3	NeuralSAT	469.6
4	nnenum	441.9
5	PyRAT	276.6
6	NNV	176.4
7	FastBATLLNN	100.0

The detailed list of results for this setting can be generated using the scripts in https://github.com/ChristopherBrix/vnncomp2023_results.

B.1 Detailed Results

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Acasxu	0	UNSAT	2.47	7.02	1.39	3.23	0.43	-	-
2023 Acasxu	1	UNSAT	2.40	7.02	1.60	3.16	0.42	-	-
2023 Acasxu	2	UNSAT	2.46	7.02	2.32	3.23	0.60	-	-
2023 Acasxu	3	UNSAT	2.45	7.02	1.82	3.16	0.62	-	-
2023 Acasxu	4	UNSAT	2.36	7.02	1.29	3.15	0.49	-	-
2023 Acasxu	5	UNSAT	2.42	7.02	1.58	3.15	0.76	-	-
2023 Acasxu	6	UNSAT	2.29	7.02	1.38	3.18	0.50	-	-
2023 Acasxu	7	UNSAT	2.21	7.02	1.14	3.14	0.58	-	-
2023 Acasxu	8	UNSAT	2.14	7.02	0.92	3.13	0.50	-	-
2023 Acasxu	9	UNSAT	2.43	7.02	1.93	3.15	0.74	-	-
2023 Acasxu	10	UNSAT	2.55	8.03	2.56	3.29	0.94	-	-
2023 Acasxu	11	UNSAT	2.42	7.02	2.06	3.16	0.71	-	-
2023 Acasxu	12	UNSAT	2.45	7.02	1.85	3.20	0.63	-	-
2023 Acasxu	13	UNSAT	2.79	9.03	3.35	3.35	0.92	-	-
2023 Acasxu	14	UNSAT	2.61	8.03	2.95	3.37	0.81	-	-
2023 Acasxu	15	UNSAT	2.96	26.1	5.93	3.40	1.41	-	-
2023 Acasxu	16	UNSAT	3.03	18.1	5.88	3.43	1.22	-	-
2023 Acasxu	17	UNSAT	3.16	24.1	7.45	3.41	61.4	-	-
2023 Acasxu	18	UNSAT	2.43	7.02	2.07	3.16	0.64	-	-
2023 Acasxu	19	UNSAT	2.49	8.02	2.16	3.16	0.59	-	-
2023 Acasxu	20	UNSAT	2.39	7.02	2.06	3.21	0.72	-	-
2023 Acasxu	21	UNSAT	2.45	7.02	1.70	3.23	0.60	-	-
2023 Acasxu	22	UNSAT	2.59	8.03	2.75	3.35	0.84	-	-
2023 Acasxu	23	UNSAT	2.74	12.0	4.89	3.39	1.58	-	-
2023 Acasxu	24	UNSAT	2.83	19.1	5.11	3.40	1.30	-	-
2023 Acasxu	25	UNSAT	2.83	16.1	4.49	3.44	1.08	-	-
2023 Acasxu	26	UNSAT	3.59	39.2	6.42	3.56	0.98	-	-
2023 Acasxu	27	UNSAT	2.59	9.03	3.02	3.35	1.02	-	-
2023 Acasxu	28	UNSAT	2.37	7.02	2.20	3.17	0.76	-	-
2023 Acasxu	29	UNSAT	2.46	7.02	2.01	3.17	0.65	-	-
2023 Acasxu	30	UNSAT	2.43	8.03	1.98	3.23	0.59	-	-
2023 Acasxu	31	UNSAT	2.54	8.03	3.00	3.34	0.94	-	-
2023 Acasxu	32	UNSAT	2.99	23.1	6.47	3.48	1.86	-	-
2023 Acasxu	33	UNSAT	3.21	32.2	10.6	3.47	1.38	-	-
2023 Acasxu	34	UNSAT	3.04	23.1	5.55	3.43	1.16	-	-
2023 Acasxu	35	UNSAT	3.30	43.2	9.29	3.48	2.39	-	-
2023 Acasxu	36	UNSAT	2.43	7.02	1.92	3.17	0.60	-	-
2023 Acasxu	37	UNSAT	2.45	8.02	2.18	3.21	0.79	-	-
2023 Acasxu	38	UNSAT	2.46	7.02	1.64	3.18	0.67	-	-
2023 Acasxu	39	UNSAT	2.43	7.02	1.69	3.19	0.73	-	-
2023 Acasxu	40	UNSAT	2.61	8.02	2.59	3.29	0.82	-	-
2023 Acasxu	41	UNSAT	2.79	13.1	4.22	3.37	1.05	-	-
2023 Acasxu	42	UNSAT	2.81	11.0	4.28	3.37	1.03	-	-
2023 Acasxu	43	UNSAT	3.12	24.1	7.49	3.45	1.17	-	-
2023 Acasxu	44	UNSAT	2.86	16.1	6.23	3.36	1.37	-	-
2023 Acasxu	45	UNSAT	4.52	15.1	3.67	3.61	0.72	-	-
2023 Acasxu	46	UNSAT	0.06	<0.01	0.97	4.42	0.29	-	-
2023 Acasxu	47	UNSAT	1.96	3.02	10.1	4.71	0.67	-	-
2023 Acasxu	48	UNSAT	0.04	<0.01	0.44	4.14	0.34	-	-
2023 Acasxu	49	UNSAT	7.04	-	23.6	5.45	0.29	-	-
2023 Acasxu	50	UNSAT	4.15	<0.01	8.46	4.12	0.86	-	-
2023 Acasxu	51	UNSAT	2.86	18.1	4.35	3.36	0.79	-	-
2023 Acasxu	52	UNSAT	2.84	25.1	4.78	3.35	1.27	-	-
2023 Acasxu	53	UNSAT	3.03	21.1	4.32	3.36	0.90	-	-
2023 Acasxu	54	UNSAT	0.04	<0.01	0.53	0.07	0.24	-	-
2023 Acasxu	55	UNSAT	0.06	<0.01	0.45	0.05	0.22	3.16	-
2023 Acasxu	56	UNSAT	0.03	<0.01	0.54	0.05	0.22	1.98	-
2023 Acasxu	57	UNSAT	0.01	<0.01	0.19	0.10	0.22	-	-
2023 Acasxu	58	UNSAT	0.06	<0.01	0.49	0.08	0.22	3.13	-
2023 Acasxu	59	UNSAT	0.06	<0.01	0.37	0.07	0.22	1.89	-
2023 Acasxu	60	UNSAT	0.04	<0.01	0.49	0.06	0.21	1.90	-
2023 Acasxu	61	UNSAT	0.04	<0.01	0.43	0.06	0.22	2.13	-
2023 Acasxu	62	UNSAT	0.05	<0.01	0.55	0.08	0.36	-	-
2023 Acasxu	63	UNSAT	0.05	<0.01	0.45	0.06	0.22	2.58	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Acasxu	64	UNSAT	1.61	<0.01	2.25	4.13	0.23	-	-
2023 Acasxu	65	UNSAT	59.0	-	-	-	4.38	-	-
2023 Acasxu	66	UNSAT	0.05	<0.01	0.69	0.07	0.21	2.91	-
2023 Acasxu	67	UNSAT	0.07	<0.01	0.47	0.08	0.22	1.99	-
2023 Acasxu	68	UNSAT	0.07	<0.01	0.67	0.04	0.23	1.71	-
2023 Acasxu	69	UNSAT	0.06	<0.01	1.41	0.05	0.35	-	-
2023 Acasxu	70	UNSAT	0.02	<0.01	0.47	0.17	0.23	-	-
2023 Acasxu	71	UNSAT	0.06	<0.01	0.46	0.05	0.22	3.25	-
2023 Acasxu	72	UNSAT	0.06	<0.01	0.52	0.16	0.23	-	-
2023 Acasxu	73	UNSAT	52.8	-	-	47.4	4.73	-	-
2023 Acasxu	74	UNSAT	0.01	<0.01	0.48	0.05	0.22	3.33	-
2023 Acasxu	75	UNSAT	0.07	<0.01	0.64	0.05	0.27	2.01	-
2023 Acasxu	76	UNSAT	0.06	<0.01	0.19	0.08	0.22	-	-
2023 Acasxu	77	UNSAT	0.06	<0.01	0.51	0.05	0.23	3.33	-
2023 Acasxu	78	UNSAT	0.02	<0.01	0.41	0.04	0.22	1.66	-
2023 Acasxu	79	UNSAT	0.09	<0.01	0.45	0.04	0.22	2.08	-
2023 Acasxu	80	UNSAT	0.07	<0.01	0.66	4.12	0.23	-	-
2023 Acasxu	81	UNSAT	0.03	<0.01	0.43	0.06	0.22	-	-
2023 Acasxu	82	UNSAT	0.06	<0.01	0.61	0.03	0.22	-	-
2023 Acasxu	83	UNSAT	24.2	83.5	-	-	0.83	-	-
2023 Acasxu	84	UNSAT	<0.01	<0.01	0.56	0.10	0.22	-	-
2023 Acasxu	85	UNSAT	0.06	<0.01	0.47	0.05	0.21	3.08	-
2023 Acasxu	86	UNSAT	0.06	<0.01	0.44	0.03	0.23	1.83	-
2023 Acasxu	87	UNSAT	0.04	<0.01	0.44	0.05	0.22	1.38	-
2023 Acasxu	88	UNSAT	0.06	<0.01	0.54	0.16	0.22	1.78	-
2023 Acasxu	89	UNSAT	0.07	<0.01	0.27	0.04	0.22	1.50	-
2023 Acasxu	90	UNSAT	13.7	17.3	75.5	7.88	0.57	-	-
2023 Acasxu	91	UNSAT	2.18	9.03	3.28	3.26	0.55	-	-
2023 Acasxu	92	UNSAT	3.03	14.1	5.10	3.50	0.44	-	-
2023 Acasxu	93	UNSAT	1.49	7.02	0.71	3.01	0.24	6.70	-
2023 Acasxu	94	UNSAT	1.26	7.02	1.05	2.29	0.28	5.18	-
2023 Acasxu	95	UNSAT	0.69	6.01	0.21	1.73	0.20	4.81	-
2023 Acasxu	96	UNSAT	0.06	<0.01	0.15	0.06	0.21	1.82	-
2023 Acasxu	97	UNSAT	0.05	<0.01	0.17	0.04	0.22	1.93	-
2023 Acasxu	98	UNSAT	0.08	<0.01	0.16	<0.01	0.21	2.04	-
2023 Acasxu	99	UNSAT	2.40	8.03	2.46	3.25	0.43	-	-
2023 Acasxu	100	UNSAT	2.29	8.02	2.88	3.23	0.36	-	-
2023 Acasxu	101	UNSAT	2.12	8.02	2.45	3.15	0.42	-	-
2023 Acasxu	102	UNSAT	0.62	6.01	0.17	1.75	0.02	6.07	-
2023 Acasxu	103	UNSAT	1.53	7.02	0.61	2.51	0.22	5.37	-
2023 Acasxu	104	UNSAT	0.65	6.02	0.17	1.74	0.22	5.20	-
2023 Acasxu	105	UNSAT	0.64	6.01	0.23	1.80	0.22	5.58	-
2023 Acasxu	106	UNSAT	0.65	6.02	0.38	1.75	0.24	4.47	-
2023 Acasxu	107	UNSAT	0.66	6.01	0.18	1.78	0.01	3.87	-
2023 Acasxu	108	UNSAT	1.52	7.02	0.86	2.99	0.22	5.29	-
2023 Acasxu	109	UNSAT	1.63	8.03	1.45	3.12	0.51	-	-
2023 Acasxu	110	UNSAT	1.53	7.02	1.36	3.08	0.38	6.90	-
2023 Acasxu	111	UNSAT	1.99	7.02	0.73	3.00	0.26	99.1	-
2023 Acasxu	112	UNSAT	1.24	7.02	0.56	2.26	0.23	5.87	-
2023 Acasxu	113	UNSAT	1.57	7.02	0.82	2.99	0.24	91.7	-
2023 Acasxu	114	UNSAT	0.63	7.02	0.19	1.76	0.20	4.99	-
2023 Acasxu	115	UNSAT	1.19	7.02	0.53	2.29	0.22	5.23	-
2023 Acasxu	116	UNSAT	1.32	7.02	0.81	2.30	0.21	4.62	-
2023 Acasxu	117	UNSAT	2.10	8.04	2.09	2.79	0.24	94.6	-
2023 Acasxu	118	UNSAT	2.17	10.0	3.00	3.23	0.51	-	-
2023 Acasxu	119	UNSAT	2.55	8.03	2.14	3.22	0.59	-	-
2023 Acasxu	120	UNSAT	1.51	7.02	0.65	2.96	0.22	6.72	-
2023 Acasxu	121	UNSAT	0.96	7.02	0.19	2.06	0.21	4.54	-
2023 Acasxu	122	UNSAT	1.53	7.02	0.77	3.03	0.22	7.12	-
2023 Acasxu	123	UNSAT	1.49	7.02	0.62	3.00	0.22	4.94	-
2023 Acasxu	124	UNSAT	0.63	7.02	0.21	1.75	0.27	5.02	-
2023 Acasxu	125	UNSAT	1.95	7.02	0.76	2.99	0.23	5.06	-
2023 Acasxu	126	UNSAT	2.58	9.04	3.53	3.30	0.42	-	-
2023 Acasxu	127	UNSAT	2.01	7.02	1.33	2.79	0.25	5.65	-
2023 Acasxu	128	UNSAT	1.49	7.03	0.89	2.56	0.25	4.99	-
2023 Acasxu	129	UNSAT	1.45	7.02	0.50	2.53	0.21	4.68	-
2023 Acasxu	130	UNSAT	1.54	7.02	0.65	2.98	0.21	5.16	-
2023 Acasxu	131	UNSAT	1.25	7.02	0.74	2.24	0.23	5.39	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Acasxu	132	UNSAT	0.94	7.02	0.19	2.05	<0.01	3.89	-
2023 Acasxu	133	UNSAT	1.22	7.02	0.65	2.22	0.23	4.52	-
2023 Acasxu	134	UNSAT	1.54	7.02	0.29	2.97	<0.01	3.36	-
2023 Acasxu	135	UNSAT	3.32	8.03	10.7	3.54	0.44	-	-
2023 Acasxu	136	UNSAT	2.60	8.03	2.74	3.60	0.43	-	-
2023 Acasxu	137	UNSAT	2.08	8.03	3.27	3.34	0.36	-	-
2023 Acasxu	138	UNSAT	1.54	7.02	0.92	3.09	0.22	76.5	-
2023 Acasxu	139	UNSAT	1.32	7.02	1.14	2.37	0.35	-	-
2023 Acasxu	140	UNSAT	1.23	7.03	0.95	2.32	0.29	5.76	-
2023 Acasxu	141	UNSAT	0.06	<0.01	0.15	0.06	0.22	1.85	-
2023 Acasxu	142	UNSAT	0.05	<0.01	0.19	0.06	0.20	1.29	-
2023 Acasxu	143	UNSAT	0.08	<0.01	0.22	0.01	0.20	1.42	-
2023 Acasxu	144	UNSAT	1.54	7.02	0.89	3.01	0.27	-	-
2023 Acasxu	145	UNSAT	1.51	7.02	0.91	2.60	0.30	-	-
2023 Acasxu	146	UNSAT	1.23	7.02	0.90	2.28	0.22	5.40	-
2023 Acasxu	147	UNSAT	1.53	7.02	0.61	2.96	0.21	4.75	-
2023 Acasxu	148	UNSAT	1.55	7.02	0.66	2.59	0.24	5.17	-
2023 Acasxu	149	UNSAT	1.54	7.02	0.69	2.55	0.22	6.11	-
2023 Acasxu	150	UNSAT	1.47	7.02	0.50	2.50	0.22	4.97	-
2023 Acasxu	151	UNSAT	1.96	7.02	0.75	3.04	0.27	95.3	-
2023 Acasxu	152	UNSAT	0.69	6.01	0.20	1.78	0.01	4.31	-
2023 Acasxu	153	UNSAT	1.50	7.02	1.13	3.03	0.24	5.16	-
2023 Acasxu	154	UNSAT	1.54	7.03	0.91	2.61	0.29	4.77	-
2023 Acasxu	155	UNSAT	0.94	7.02	0.20	2.05	0.21	4.16	-
2023 Acasxu	156	UNSAT	1.22	7.03	0.57	2.21	0.21	4.53	-
2023 Acasxu	157	UNSAT	1.53	8.03	0.67	3.00	0.25	5.47	-
2023 Acasxu	158	UNSAT	1.95	7.03	0.67	3.01	0.22	6.12	-
2023 Acasxu	159	UNSAT	1.50	7.02	0.54	2.50	0.23	4.45	-
2023 Acasxu	160	UNSAT	1.55	7.02	0.86	2.59	0.29	72.1	-
2023 Acasxu	161	UNSAT	1.53	7.02	0.68	2.99	0.33	5.89	-
2023 Acasxu	162	UNSAT	0.69	7.02	0.21	1.78	0.21	4.17	-
2023 Acasxu	163	UNSAT	1.95	7.03	0.63	2.98	0.23	5.28	-
2023 Acasxu	164	UNSAT	1.50	7.02	0.77	2.55	0.23	4.97	-
2023 Acasxu	165	UNSAT	1.51	7.02	0.64	3.00	0.24	80.9	-
2023 Acasxu	166	UNSAT	1.52	7.02	0.51	2.54	0.22	6.81	-
2023 Acasxu	167	UNSAT	1.52	7.02	0.50	2.52	0.21	5.02	-
2023 Acasxu	168	UNSAT	1.26	7.02	0.75	2.28	0.21	5.45	-
2023 Acasxu	169	UNSAT	1.52	7.03	0.63	2.99	0.22	5.56	-
2023 Acasxu	170	UNSAT	1.50	7.03	0.54	2.54	0.22	4.82	-
2023 Acasxu	171	UNSAT	1.49	7.02	0.54	3.01	0.22	4.99	-
2023 Acasxu	172	UNSAT	1.53	7.02	0.50	2.92	0.21	4.49	-
2023 Acasxu	173	UNSAT	1.52	7.02	0.65	2.96	0.23	4.63	-
2023 Acasxu	174	UNSAT	1.17	7.02	0.51	2.24	0.23	4.52	-
2023 Acasxu	175	UNSAT	1.60	7.02	0.59	2.98	0.22	5.34	-
2023 Acasxu	176	UNSAT	0.65	7.02	0.22	1.75	0.23	4.66	-
2023 Acasxu	177	UNSAT	0.95	7.02	0.21	2.06	0.22	5.01	-
2023 Acasxu	178	UNSAT	1.50	7.02	0.68	2.54	0.22	5.73	-
2023 Acasxu	179	UNSAT	1.23	7.02	0.56	2.25	0.22	5.00	-
2023 Acasxu	180	UNSAT	22.1	-	81.4	9.82	0.95	-	-
2023 Acasxu	181	UNSAT	14.3	-	30.2	3.34	3.74	-	-
2023 Acasxu	182	UNSAT	✗	62.7	-	5.16	30.6	-	-
2023 Acasxu	183	UNSAT	0.06	<0.01	3.30	4.46	0.27	-	-
2023 Acasxu	184	UNSAT	6.63	106	16.6	4.55	2.05	-	-
2023 Acasxu	185	UNSAT	5.68	24.1	3.43	3.57	0.64	-	-
2023 Cgan	0	UNSAT	1.73	<0.01	2.75	0.52	9.98	2.37	-
2023 Cgan	1	UNSAT	1.70	<0.01	0.35	0.47	31.1	2.42	-
2023 Cgan	2	UNSAT	1.72	<0.01	2.28	0.46	13.7	2.13	-
2023 Cgan	3	UNSAT	5.68	-	10.6	16.4	24.9	-	-
2023 Cgan	4	UNSAT	1.72	<0.01	1.76	0.45	10.2	2.14	-
2023 Cgan	5	UNSAT	8.02	-	13.1	810	53.8	-	-
2023 Cgan	6	UNSAT	1.70	<0.01	0.21	0.42	0.94	2.46	-
2023 Cgan	7	UNSAT	1.75	<0.01	0.30	0.45	14.7	2.08	-
2023 Cgan	8	UNSAT	13.6	-	71.6	-	159	-	-
2023 Cgan	9	UNSAT	8.80	-	21.4	-	39.4	98.1	-
2023 Cgan	10	UNSAT	1.69	0.01	0.26	0.46	0.95	2.47	-
2023 Cgan	11	UNSAT	1.74	<0.01	0.23	0.44	0.87	2.64	-
2023 Cgan	12	UNSAT	9.95	-	19.5	-	30.0	28.0	-
2023 Cgan	13	UNSAT	9.56	-	48.2	-	57.1	137	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Cgan	14	UNSAT	1.72	<0.01	0.24	0.47	1.30	2.15	-
2023 Cgan	15	UNSAT	1.75	<0.01	0.26	0.48	55.6	2.33	-
2023 Cgan	16	UNSAT	5.89	-	10.5	22.3	-	-	-
2023 Cgan	17	UNSAT	1.73	<0.01	0.24	0.43	7.89	2.15	-
2023 Cgan	18	UNSAT	5.07	-	9.25	-	-	-	-
2023 Cgan	19	UNSAT	9.44	1.01	-	-	-	-	-
2023 Cgan	20	UNSAT	9.39	1.01	-	-	-	-	-
2023 Collins Rul Cnn	0	UNSAT	1.50	0.01	0.43	0.47	0.71	✗	-
2023 Collins Rul Cnn	1	UNSAT	1.53	<0.01	0.46	3.34	0.75	✗	-
2023 Collins Rul Cnn	2	UNSAT	3.13	8.02	0.81	1.78	0.68	2.25	-
2023 Collins Rul Cnn	3	UNSAT	3.10	7.02	0.81	1.75	0.76	1.78	-
2023 Collins Rul Cnn	4	UNSAT	1.54	<0.01	0.46	3.32	0.68	✗	-
2023 Collins Rul Cnn	5	UNSAT	3.14	9.13	0.80	1.96	0.75	2.47	-
2023 Collins Rul Cnn	6	UNSAT	1.50	<0.01	0.44	3.36	0.87	✗	-
2023 Collins Rul Cnn	7	UNSAT	3.25	10.1	0.82	1.93	0.70	2.45	-
2023 Collins Rul Cnn	8	UNSAT	3.09	8.03	0.82	1.76	0.77	2.18	-
2023 Collins Rul Cnn	9	UNSAT	3.11	7.02	0.80	1.76	0.75	2.12	-
2023 Collins Rul Cnn	10	UNSAT	1.52	<0.01	0.44	0.44	0.69	✗	-
2023 Collins Rul Cnn	11	UNSAT	1.54	<0.01	0.47	0.42	0.73	✗	-
2023 Collins Rul Cnn	12	UNSAT	3.08	10.1	0.83	1.75	0.74	2.82	-
2023 Collins Rul Cnn	13	UNSAT	3.12	9.04	0.81	1.78	0.76	3.22	-
2023 Collins Rul Cnn	14	UNSAT	3.27	10.2	0.84	1.91	0.70	3.36	-
2023 Collins Rul Cnn	15	UNSAT	3.15	9.11	0.84	1.95	0.74	3.35	-
2023 Collins Rul Cnn	16	UNSAT	3.15	7.02	0.90	1.76	0.73	3.29	-
2023 Collins Rul Cnn	17	UNSAT	1.52	<0.01	0.51	4.55	0.71	✗	-
2023 Collins Rul Cnn	18	UNSAT	3.09	7.02	0.81	1.77	0.70	2.00	-
2023 Collins Rul Cnn	19	UNSAT	1.57	<0.01	0.45	0.42	7.39	✗	-
2023 Collins Rul Cnn	20	UNSAT	1.53	<0.01	0.44	0.44	0.77	✗	-
2023 Collins Rul Cnn	21	UNSAT	1.53	<0.01	0.45	0.48	0.76	✗	-
2023 Collins Rul Cnn	22	UNSAT	1.56	<0.01	0.44	0.45	0.73	✗	-
2023 Collins Rul Cnn	23	UNSAT	1.55	<0.01	0.46	4.20	0.79	✗	-
2023 Collins Rul Cnn	24	UNSAT	3.09	10.0	0.83	1.81	0.72	2.35	-
2023 Collins Rul Cnn	25	UNSAT	3.22	10.0	0.84	1.77	0.68	2.56	-
2023 Collins Rul Cnn	26	UNSAT	3.25	14.1	0.85	1.96	0.78	2.44	-
2023 Collins Rul Cnn	27	UNSAT	3.16	10.0	0.94	1.94	0.75	2.63	-
2023 Collins Rul Cnn	28	UNSAT	1.53	<0.01	0.44	3.73	0.73	✗	-
2023 Collins Rul Cnn	29	UNSAT	1.55	<0.01	0.46	0.42	0.72	✗	-
2023 Collins Rul Cnn	30	UNSAT	3.24	10.0	0.86	1.78	0.77	2.88	-
2023 Collins Rul Cnn	31	UNSAT	3.09	10.0	0.86	1.82	0.79	2.83	-
2023 Collins Rul Cnn	32	UNSAT	1.54	<0.01	0.43	0.42	0.83	✗	-
2023 Collins Rul Cnn	33	UNSAT	1.57	<0.01	0.44	0.43	0.78	✗	-
2023 Collins Rul Cnn	34	UNSAT	1.57	<0.01	0.53	3.32	0.73	✗	-
2023 Collins Rul Cnn	35	UNSAT	3.12	14.3	0.85	1.82	0.70	2.88	-
2023 Collins Rul Cnn	36	UNSAT	3.28	15.2	0.88	1.92	0.93	2.81	-
2023 Collins Rul Cnn	37	UNSAT	3.14	15.2	0.85	1.96	0.73	3.08	-
2023 Collins Rul Cnn	38	UNSAT	3.07	10.0	0.92	1.77	0.68	2.30	-
2023 Collins Rul Cnn	39	UNSAT	1.57	<0.01	0.46	4.53	0.78	✗	-
2023 Collins Rul Cnn	40	UNSAT	3.14	10.0	0.88	1.77	0.70	2.55	-
2023 Collins Rul Cnn	41	UNSAT	1.54	<0.01	0.44	0.42	0.78	✗	-
2023 Collins Rul Cnn	42	UNSAT	1.55	<0.01	0.44	0.41	0.70	✗	-
2023 Collins Rul Cnn	43	UNSAT	3.44	48.8	1.52	1.97	0.82	-	-
2023 Collins Rul Cnn	44	UNSAT	1.56	<0.01	0.44	0.42	1.09	✗	-
2023 Collins Rul Cnn	45	UNSAT	3.47	51.9	1.51	1.79	0.74	-	-
2023 Collins Rul Cnn	46	UNSAT	1.57	<0.01	0.49	0.44	0.79	✗	-
2023 Collins Rul Cnn	47	UNSAT	3.48	47.0	1.50	1.94	0.95	-	-
2023 Collins Rul Cnn	48	UNSAT	3.62	48.9	1.53	1.92	0.85	-	-
2023 Collins Rul Cnn	49	UNSAT	3.48	49.8	1.48	1.98	0.72	-	-
2023 Collins Rul Cnn	50	UNSAT	1.52	<0.01	0.45	0.47	0.75	✗	-
2023 Collins Rul Cnn	51	UNSAT	1.55	<0.01	0.45	0.45	0.74	✗	-
2023 Collins Rul Cnn	52	UNSAT	3.50	49.9	1.54	1.84	0.76	-	-
2023 Collins Rul Cnn	53	UNSAT	3.66	46.9	1.58	1.95	0.93	-	-
2023 Collins Rul Cnn	54	UNSAT	1.61	<0.01	0.46	0.45	3.51	✗	-
2023 Collins Rul Cnn	55	UNSAT	3.50	52.9	1.57	1.93	0.81	-	-
2023 Collins Rul Cnn	56	UNSAT	3.55	50.8	1.57	1.96	1.03	-	-
2023 Collins Rul Cnn	57	UNSAT	1.54	<0.01	0.47	0.42	0.74	✗	-
2023 Collins Rul Cnn	58	UNSAT	1.58	<0.01	0.46	0.44	0.88	✗	-
2023 Collins Rul Cnn	59	UNSAT	3.49	47.8	-	1.93	0.90	-	-
2023 Collins Rul Cnn	60	UNSAT	3.50	50.0	-	1.96	1.14	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Collins Rul Cnn	61	UNSAT	3.52	46.7	1.62	1.95	0.79	-	-
2023 Dist Shift	0	UNSAT	0.93	10.0	0.72	1.85	-	-	-
2023 Dist Shift	1	UNSAT	1.50	-	0.74	2.25	-	-	-
2023 Dist Shift	2	UNSAT	0.97	13.1	0.73	1.61	-	-	-
2023 Dist Shift	3	UNSAT	1.01	63.4	6.85	1.68	-	-	-
2023 Dist Shift	4	UNSAT	3.07	78.6	3.48	2.34	-	-	-
2023 Dist Shift	5	UNSAT	0.98	8.03	0.71	1.67	-	-	-
2023 Dist Shift	6	UNSAT	4.73	-	9.33	2.47	-	-	-
2023 Dist Shift	7	UNSAT	0.06	<0.01	0.50	0.02	-	1.09	-
2023 Dist Shift	8	UNSAT	3.03	-	1.65	2.27	-	-	-
2023 Dist Shift	9	UNSAT	3.03	-	4.98	2.31	-	-	-
2023 Dist Shift	10	UNSAT	1.02	13.1	0.70	1.66	-	-	-
2023 Dist Shift	11	UNSAT	0.97	8.03	0.71	1.63	-	-	-
2023 Dist Shift	12	UNSAT	1.58	102	0.70	2.29	-	-	-
2023 Dist Shift	13	UNSAT	0.96	8.03	0.67	1.69	-	-	-
2023 Dist Shift	14	UNSAT	3.05	8.03	1.26	2.33	-	-	-
2023 Dist Shift	15	UNSAT	1.47	9.04	1.68	2.30	-	-	-
2023 Dist Shift	16	UNSAT	0.96	8.02	0.71	1.64	-	-	-
2023 Dist Shift	17	UNSAT	0.97	8.03	0.71	1.61	-	-	-
2023 Dist Shift	18	UNSAT	0.99	8.02	0.73	1.64	-	-	-
2023 Dist Shift	19	UNSAT	0.98	8.03	0.72	1.66	-	-	-
2023 Dist Shift	20	UNSAT	1.59	117	1.30	2.26	-	-	-
2023 Dist Shift	21	UNSAT	0.94	8.03	0.70	1.63	-	-	-
2023 Dist Shift	22	UNSAT	0.01	<0.01	1.21	<0.01	-	1.25	-
2023 Dist Shift	23	UNSAT	0.96	8.02	0.78	1.63	-	-	-
2023 Dist Shift	24	UNSAT	1.00	8.03	0.69	1.65	-	-	-
2023 Dist Shift	25	UNSAT	0.96	8.03	0.73	1.66	-	-	-
2023 Dist Shift	26	UNSAT	3.93	-	1.63	2.32	-	-	-
2023 Dist Shift	27	UNSAT	12.4	-	-	-	-	-	-
2023 Dist Shift	28	UNSAT	0.96	8.03	0.69	1.61	-	-	-
2023 Dist Shift	29	UNSAT	1.71	105	0.72	2.25	-	-	-
2023 Dist Shift	30	UNSAT	3.03	-	7.07	2.33	-	-	-
2023 Dist Shift	31	UNSAT	1.61	42.3	2.07	2.29	-	-	-
2023 Dist Shift	32	UNSAT	2.97	-	1.37	2.28	-	-	-
2023 Dist Shift	33	UNSAT	3.05	121	1.69	2.30	-	-	-
2023 Dist Shift	34	UNSAT	0.96	10.0	0.70	1.67	-	-	-
2023 Dist Shift	35	UNSAT	1.50	80.5	0.73	2.30	-	-	-
2023 Dist Shift	36	UNSAT	3.03	134	2.73	2.31	-	-	-
2023 Dist Shift	37	UNSAT	3.05	9.03	1.39	2.28	-	-	-
2023 Dist Shift	38	UNSAT	0.98	8.03	0.69	1.64	-	-	-
2023 Dist Shift	39	UNSAT	0.97	9.04	0.74	1.66	-	-	-
2023 Dist Shift	40	UNSAT	0.96	9.03	1.87	1.67	-	-	-
2023 Dist Shift	41	UNSAT	0.99	8.03	0.77	1.63	-	-	-
2023 Dist Shift	42	UNSAT	0.97	8.03	0.71	1.68	-	-	-
2023 Dist Shift	43	UNSAT	0.03	<0.01	0.51	0.04	-	0.94	-
2023 Dist Shift	44	UNSAT	0.09	<0.01	0.50	0.03	-	1.06	-
2023 Dist Shift	45	UNSAT	3.06	-	0.72	2.28	-	-	-
2023 Dist Shift	46	UNSAT	0.94	8.03	0.72	1.66	-	-	-
2023 Dist Shift	47	UNSAT	1.72	8.03	0.71	2.29	-	-	-
2023 Dist Shift	48	UNSAT	0.94	8.03	0.69	1.65	-	-	-
2023 Dist Shift	49	UNSAT	1.74	159	0.70	2.26	-	-	-
2023 Dist Shift	50	UNSAT	1.00	8.03	0.77	1.67	-	-	-
2023 Dist Shift	51	UNSAT	0.98	8.03	0.70	1.62	-	-	-
2023 Dist Shift	52	UNSAT	0.95	8.03	0.71	1.64	-	-	-
2023 Dist Shift	53	UNSAT	0.99	8.03	0.69	1.64	-	-	-
2023 Dist Shift	54	UNSAT	0.08	2.02	142	0.39	-	-	-
2023 Dist Shift	55	UNSAT	0.97	8.03	0.71	1.66	-	-	-
2023 Dist Shift	56	UNSAT	4.05	-	4.41	2.36	-	-	-
2023 Dist Shift	57	UNSAT	1.48	14.1	1.34	2.27	-	-	-
2023 Dist Shift	58	UNSAT	0.95	8.03	0.70	1.64	-	-	-
2023 Dist Shift	59	UNSAT	0.99	8.03	0.71	1.64	-	-	-
2023 Dist Shift	60	UNSAT	0.97	8.03	0.71	1.68	-	-	-
2023 Dist Shift	61	UNSAT	0.99	8.03	0.69	1.65	-	-	-
2023 Dist Shift	62	UNSAT	1.45	46.3	0.70	2.31	-	-	-
2023 Dist Shift	63	UNSAT	1.51	16.1	0.72	2.28	-	-	-
2023 Dist Shift	64	UNSAT	1.60	16.1	10.2	2.31	-	-	-
2023 Dist Shift	65	UNSAT	1.86	13.1	2.82	3.19	-	-	-
2023 Dist Shift	66	UNSAT	9.61	-	23.5	3.46	-	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Dist Shift	67	UNSAT	0.97	8.03	0.71	1.65	-	-	-
2023 Dist Shift	68	UNSAT	0.95	11.0	0.68	1.68	-	-	-
2023 Dist Shift	69	UNSAT	5.35	-	38.1	3.41	-	-	-
2023 Dist Shift	70	UNSAT	0.92	8.02	0.70	1.64	-	-	-
2023 Dist Shift	71	UNSAT	1.42	12.1	0.72	2.29	-	-	-
2023 Ml4acopf	0	UNSAT	29.0	-	-	-	-	-	-
2023 Ml4acopf	1	?	-	-	-	-	-	-	-
2023 Ml4acopf	2	?	-	-	-	-	-	-	-
2023 Ml4acopf	3	UNSAT	2.22	-	-	-	-	-	-
2023 Ml4acopf	4	UNSAT	2.30	-	-	-	-	-	-
2023 Ml4acopf	5	UNSAT	2.15	-	-	-	-	-	-
2023 Ml4acopf	6	UNSAT	2.17	-	-	-	-	-	-
2023 Ml4acopf	7	UNSAT	2.60	-	-	-	-	-	-
2023 Ml4acopf	8	UNSAT	2.12	-	-	-	-	-	-
2023 Ml4acopf	9	UNSAT	2.18	-	-	-	-	-	-
2023 Ml4acopf	10	UNSAT	2.12	-	-	-	-	-	-
2023 Ml4acopf	11	UNSAT	2.21	-	-	-	-	-	-
2023 Ml4acopf	12	UNSAT	2.18	-	-	-	-	-	-
2023 Ml4acopf	13	UNSAT	2.11	-	-	-	-	-	-
2023 Ml4acopf	14	UNSAT	2.19	-	-	-	-	-	-
2023 Ml4acopf	15	UNSAT	7.92	-	-	-	-	-	-
2023 Ml4acopf	16	UNSAT	2.19	-	-	-	-	-	-
2023 Ml4acopf	17	UNSAT	2.14	-	-	-	-	-	-
2023 Ml4acopf	18	UNSAT	2.10	-	-	-	-	-	-
2023 Ml4acopf	19	?	-	-	-	-	-	-	-
2023 Ml4acopf	20	UNSAT	2.48	-	-	-	-	-	-
2023 Ml4acopf	21	UNSAT	0.66	-	-	-	-	-	-
2023 Ml4acopf	22	?	-	-	-	-	-	-	-
2023 Nn4sys	0	UNSAT	2.36	7.02	0.18	4.36	-	-	-
2023 Nn4sys	1	UNSAT	2.34	7.02	0.35	3.55	-	-	-
2023 Nn4sys	2	UNSAT	0.81	7.02	0.28	3.10	-	-	-
2023 Nn4sys	3	UNSAT	2.42	7.02	0.23	3.55	-	-	-
2023 Nn4sys	4	UNSAT	0.82	7.02	0.19	3.10	-	-	-
2023 Nn4sys	5	UNSAT	1.90	-	-	4.83	-	-	-
2023 Nn4sys	6	UNSAT	3.88	-	-	-	-	-	-
2023 Nn4sys	7	UNSAT	2.37	7.02	0.27	3.62	-	-	-
2023 Nn4sys	8	UNSAT	3.92	-	-	-	-	-	-
2023 Nn4sys	9	UNSAT	0.82	7.02	0.23	3.10	-	-	-
2023 Nn4sys	10	UNSAT	3.92	-	-	-	-	-	-
2023 Nn4sys	11	UNSAT	2.37	7.02	0.25	3.59	-	-	-
2023 Nn4sys	12	UNSAT	0.83	7.02	0.14	3.10	-	-	-
2023 Nn4sys	13	UNSAT	0.84	7.02	0.16	3.05	-	-	-
2023 Nn4sys	14	UNSAT	3.87	-	-	5.54	-	-	-
2023 Nn4sys	15	UNSAT	0.83	7.02	0.15	3.08	-	-	-
2023 Nn4sys	16	UNSAT	3.91	-	-	-	-	-	-
2023 Nn4sys	17	UNSAT	2.34	7.02	0.27	3.60	-	-	-
2023 Nn4sys	18	UNSAT	0.84	7.02	0.20	3.08	-	-	-
2023 Nn4sys	19	UNSAT	0.82	7.02	0.23	3.09	-	-	-
2023 Nn4sys	20	UNSAT	0.83	7.02	0.23	3.06	-	-	-
2023 Nn4sys	21	UNSAT	2.36	8.03	0.24	3.83	-	-	-
2023 Nn4sys	22	UNSAT	0.85	7.02	0.14	3.09	-	-	-
2023 Nn4sys	23	UNSAT	3.92	-	-	-	-	-	-
2023 Nn4sys	24	UNSAT	1.91	-	-	4.81	-	-	-
2023 Nn4sys	25	UNSAT	2.37	7.02	0.17	3.57	-	-	-
2023 Nn4sys	26	UNSAT	3.93	-	-	5.57	-	-	-
2023 Nn4sys	27	UNSAT	2.33	7.02	0.17	3.55	-	-	-
2023 Nn4sys	28	UNSAT	1.89	-	-	4.85	-	-	-
2023 Nn4sys	29	UNSAT	0.82	7.02	0.17	3.09	-	-	-
2023 Nn4sys	30	UNSAT	0.86	7.02	0.16	3.07	-	-	-
2023 Nn4sys	31	UNSAT	2.45	7.02	0.25	3.58	-	-	-
2023 Nn4sys	32	UNSAT	2.40	8.03	0.24	3.88	-	-	-
2023 Nn4sys	33	UNSAT	2.44	8.03	0.32	3.83	-	-	-
2023 Nn4sys	34	UNSAT	2.40	8.03	0.21	3.83	-	-	-
2023 Nn4sys	35	UNSAT	6.28	-	-	-	-	-	-
2023 Nn4sys	36	UNSAT	6.11	-	-	-	-	-	-
2023 Nn4sys	37	UNSAT	6.34	-	-	-	-	-	-
2023 Nn4sys	38	UNSAT	6.27	-	-	-	-	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Nn4sys	39	UNSAT	6.70	-	-	-	-	-	-
2023 Nn4sys	40	UNSAT	6.46	-	-	-	-	-	-
2023 Nn4sys	41	UNSAT	5.99	-	-	-	-	-	-
2023 Nn4sys	42	UNSAT	6.40	-	-	-	-	-	-
2023 Nn4sys	43	UNSAT	6.74	-	-	-	-	-	-
2023 Nn4sys	44	UNSAT	6.32	-	-	-	-	-	-
2023 Nn4sys	45	UNSAT	6.35	-	-	-	-	-	-
2023 Nn4sys	46	UNSAT	6.28	-	-	-	-	-	-
2023 Nn4sys	47	UNSAT	6.53	-	-	-	-	-	-
2023 Nn4sys	48	UNSAT	6.25	-	-	-	-	-	-
2023 Nn4sys	49	UNSAT	6.36	-	-	-	-	-	-
2023 Nn4sys	50	UNSAT	6.53	-	-	-	-	-	-
2023 Nn4sys	51	UNSAT	6.67	-	-	-	-	-	-
2023 Nn4sys	52	UNSAT	6.22	-	-	-	-	-	-
2023 Nn4sys	53	UNSAT	6.44	-	-	-	-	-	-
2023 Nn4sys	54	UNSAT	6.68	-	-	-	-	-	-
2023 Nn4sys	55	UNSAT	6.25	-	-	-	-	-	-
2023 Nn4sys	56	UNSAT	7.01	-	-	-	-	-	-
2023 Nn4sys	57	UNSAT	6.16	-	-	-	-	-	-
2023 Nn4sys	58	UNSAT	6.18	-	-	-	-	-	-
2023 Nn4sys	59	UNSAT	6.25	-	-	-	-	-	-
2023 Nn4sys	60	UNSAT	6.19	-	-	-	-	-	-
2023 Nn4sys	61	UNSAT	6.45	-	-	-	-	-	-
2023 Nn4sys	62	UNSAT	6.47	-	-	-	-	-	-
2023 Nn4sys	63	UNSAT	6.59	-	-	-	-	-	-
2023 Nn4sys	64	UNSAT	6.06	-	-	-	-	-	-
2023 Nn4sys	65	UNSAT	6.43	-	-	-	-	-	-
2023 Nn4sys	66	UNSAT	6.12	-	-	-	-	-	-
2023 Nn4sys	67	UNSAT	6.45	-	-	-	-	-	-
2023 Nn4sys	68	UNSAT	6.43	-	-	-	-	-	-
2023 Nn4sys	69	UNSAT	6.40	-	-	-	-	-	-
2023 Nn4sys	70	UNSAT	6.47	-	-	-	-	-	-
2023 Nn4sys	71	UNSAT	6.77	-	-	-	-	-	-
2023 Nn4sys	72	UNSAT	6.78	-	-	-	-	-	-
2023 Nn4sys	73	UNSAT	6.79	-	-	-	-	-	-
2023 Nn4sys	74	UNSAT	6.48	-	-	-	-	-	-
2023 Nn4sys	75	UNSAT	6.69	-	-	-	-	-	-
2023 Nn4sys	76	UNSAT	6.97	-	-	-	-	-	-
2023 Nn4sys	77	UNSAT	6.72	-	-	-	-	-	-
2023 Nn4sys	78	UNSAT	6.73	-	-	-	-	-	-
2023 Nn4sys	79	UNSAT	7.31	-	-	-	-	-	-
2023 Nn4sys	80	UNSAT	6.69	-	-	-	-	-	-
2023 Nn4sys	81	UNSAT	6.72	-	-	-	-	-	-
2023 Nn4sys	82	UNSAT	6.47	-	-	-	-	-	-
2023 Nn4sys	83	UNSAT	6.94	-	-	-	-	-	-
2023 Nn4sys	84	UNSAT	6.75	-	-	-	-	-	-
2023 Nn4sys	85	UNSAT	6.77	-	-	-	-	-	-
2023 Nn4sys	86	UNSAT	6.71	-	-	-	-	-	-
2023 Nn4sys	87	UNSAT	6.65	-	-	-	-	-	-
2023 Nn4sys	88	UNSAT	6.37	-	-	-	-	-	-
2023 Nn4sys	89	UNSAT	6.64	-	-	-	-	-	-
2023 Nn4sys	90	UNSAT	6.61	-	-	-	-	-	-
2023 Nn4sys	91	UNSAT	7.23	-	-	-	-	-	-
2023 Nn4sys	92	UNSAT	6.41	-	-	-	-	-	-
2023 Nn4sys	93	UNSAT	6.64	-	-	-	-	-	-
2023 Nn4sys	94	UNSAT	6.37	-	-	-	-	-	-
2023 Nn4sys	95	UNSAT	6.42	-	-	-	-	-	-
2023 Nn4sys	96	UNSAT	6.60	-	-	-	-	-	-
2023 Nn4sys	97	UNSAT	6.38	-	-	-	-	-	-
2023 Nn4sys	98	UNSAT	6.49	-	-	-	-	-	-
2023 Nn4sys	99	UNSAT	6.54	-	-	-	-	-	-
2023 Nn4sys	100	UNSAT	6.63	-	-	-	-	-	-
2023 Nn4sys	101	UNSAT	6.39	-	-	-	-	-	-
2023 Nn4sys	102	UNSAT	6.77	-	-	-	-	-	-
2023 Nn4sys	103	UNSAT	6.57	-	-	-	-	-	-
2023 Nn4sys	104	UNSAT	6.76	-	-	-	-	-	-
2023 Nn4sys	105	UNSAT	0.09	6.02	0.36	1.55	0.02	-	-
2023 Nn4sys	106	UNSAT	0.09	7.02	0.35	1.58	0.03	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Nn4sys	107	UNSAT	0.29	21.1	13.3	1.07	0.43	✗	-
2023 Nn4sys	108	UNSAT	0.22	44.3	16.2	1.56	0.52	✗	-
2023 Nn4sys	109	UNSAT	0.24	36.2	24.8	1.47	0.83	✗	-
2023 Nn4sys	110	UNSAT	0.24	81.5	31.1	1.69	0.93	✗	-
2023 Nn4sys	111	UNSAT	0.22	50.3	39.1	2.42	1.22	✗	-
2023 Nn4sys	112	UNSAT	0.28	-	48.3	2.34	1.37	✗	-
2023 Nn4sys	113	UNSAT	0.22	65.4	51.4	1.78	1.61	✗	-
2023 Nn4sys	114	UNSAT	0.29	-	63.0	2.67	1.78	✗	-
2023 Nn4sys	115	UNSAT	0.25	79.5	64.9	1.97	2.00	4.08	-
2023 Nn4sys	116	UNSAT	0.29	-	79.0	2.88	2.23	4.14	-
2023 Nn4sys	117	UNSAT	0.43	-	-	4.27	5.96	-	-
2023 Nn4sys	118	UNSAT	0.43	-	-	4.52	6.93	-	-
2023 Nn4sys	119	UNSAT	0.50	-	-	6.64	9.88	-	-
2023 Nn4sys	120	UNSAT	0.52	-	-	7.23	11.0	-	-
2023 Nn4sys	121	UNSAT	0.62	-	-	8.74	13.9	-	-
2023 Nn4sys	122	UNSAT	0.59	-	-	9.58	15.4	-	-
2023 Nn4sys	123	UNSAT	0.77	-	-	10.8	17.7	-	-
2023 Nn4sys	124	UNSAT	0.81	-	-	12.2	20.0	-	-
2023 Nn4sys	125	UNSAT	0.75	-	-	12.1	20.3	-	-
2023 Nn4sys	126	UNSAT	0.80	-	-	13.3	21.8	-	-
2023 Nn4sys	127	UNSAT	3.93	-	-	-	-	-	-
2023 Nn4sys	128	UNSAT	4.08	-	-	-	-	-	-
2023 Nn4sys	129	UNSAT	1.98	-	0.42	3.77	-	-	-
2023 Nn4sys	130	UNSAT	2.46	-	-	7.43	-	-	-
2023 Nn4sys	131	UNSAT	3.01	-	-	9.98	-	-	-
2023 Nn4sys	132	UNSAT	3.46	-	-	11.8	-	-	-
2023 Nn4sys	133	UNSAT	5.35	-	-	22.1	-	-	-
2023 Nn4sys	134	UNSAT	6.07	-	-	27.4	-	-	-
2023 Nn4sys	135	UNSAT	6.89	-	-	32.5	-	-	-
2023 Nn4sys	136	UNSAT	7.49	-	-	37.4	-	-	-
2023 Nn4sys	137	UNSAT	2.70	-	3.57	5.57	-	-	-
2023 Nn4sys	138	UNSAT	3.96	-	-	12.0	-	-	-
2023 Nn4sys	139	UNSAT	4.03	-	-	12.7	-	-	-
2023 Nn4sys	140	UNSAT	3.82	-	-	14.9	-	-	-
2023 Nn4sys	141	UNSAT	4.08	-	-	17.2	-	-	-
2023 Nn4sys	142	UNSAT	4.21	-	-	19.0	-	-	-
2023 Nn4sys	143	UNSAT	4.28	-	-	20.1	-	-	-
2023 Nn4sys	144	UNSAT	4.26	-	-	23.7	-	-	-
2023 Nn4sys	145	UNSAT	8.56	-	-	47.5	-	-	-
2023 Nn4sys	146	UNSAT	9.49	-	-	59.3	-	-	-
2023 Nn4sys	147	UNSAT	12.0	-	-	74.0	-	-	-
2023 Nn4sys	148	UNSAT	14.0	-	-	97.3	-	-	-
2023 Nn4sys	149	UNSAT	15.3	-	-	96.8	-	-	-
2023 Nn4sys	150	UNSAT	16.9	-	-	110	-	-	-
2023 Nn4sys	151	UNSAT	19.3	-	-	127	-	-	-
2023 Nn4sys	152	UNSAT	20.2	-	-	157	-	-	-
2023 Nn4sys	153	UNSAT	21.8	-	-	142	-	-	-
2023 Nn4sys	154	UNSAT	22.4	-	-	154	-	-	-
2023 Nn4sys	155	UNSAT	25.2	-	-	171	-	-	-
2023 Nn4sys	156	UNSAT	27.1	-	-	-	-	-	-
2023 Nn4sys	157	UNSAT	28.5	-	-	198	-	-	-
2023 Nn4sys	158	UNSAT	30.3	-	-	-	-	-	-
2023 Nn4sys	159	UNSAT	33.1	-	-	259	-	-	-
2023 Nn4sys	160	UNSAT	3.75	-	35.9	6.32	-	-	-
2023 Nn4sys	161	UNSAT	5.98	-	-	-	-	-	-
2023 Nn4sys	162	UNSAT	8.72	-	-	-	-	-	-
2023 Nn4sys	163	UNSAT	9.94	-	-	-	-	-	-
2023 Nn4sys	164	UNSAT	19.8	-	-	-	-	-	-
2023 Nn4sys	165	UNSAT	24.5	-	-	-	-	-	-
2023 Nn4sys	166	UNSAT	28.9	-	-	-	-	-	-
2023 Nn4sys	167	UNSAT	33.3	-	-	-	-	-	-
2023 Nn4sys	168	UNSAT	38.0	-	-	-	-	-	-
2023 Nn4sys	169	UNSAT	42.6	-	-	-	-	-	-
2023 Nn4sys	170	UNSAT	46.8	-	-	-	-	-	-
2023 Nn4sys	171	UNSAT	3.98	-	-	8.69	-	-	-
2023 Nn4sys	172	UNSAT	7.58	-	-	-	-	-	-
2023 Nn4sys	173	UNSAT	8.89	-	-	-	-	-	-
2023 Nn4sys	174	UNSAT	10.8	-	-	-	-	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Nn4sys	175	UNSAT	12.2	-	-	-	-	-	-
2023 Nn4sys	176	UNSAT	15.3	-	-	-	-	-	-
2023 Nn4sys	177	UNSAT	16.9	-	-	-	-	-	-
2023 Nn4sys	178	UNSAT	17.9	-	-	-	-	-	-
2023 Nn4sys	179	UNSAT	40.2	-	-	-	-	-	-
2023 Nn4sys	180	UNSAT	50.0	-	-	-	-	-	-
2023 Nn4sys	181	UNSAT	59.0	-	-	-	-	-	-
2023 Nn4sys	182	UNSAT	71.7	-	-	-	-	-	-
2023 Nn4sys	183	UNSAT	80.6	-	-	-	-	-	-
2023 Nn4sys	184	UNSAT	93.7	-	-	-	-	-	-
2023 Nn4sys	185	UNSAT	105	-	-	-	-	-	-
2023 Nn4sys	186	UNSAT	115	-	-	-	-	-	-
2023 Nn4sys	187	UNSAT	127	-	-	-	-	-	-
2023 Nn4sys	188	UNSAT	134	-	-	-	-	-	-
2023 Nn4sys	189	UNSAT	144	-	-	-	-	-	-
2023 Nn4sys	190	UNSAT	157	-	-	-	-	-	-
2023 Nn4sys	191	UNSAT	166	-	-	-	-	-	-
2023 Nn4sys	192	UNSAT	176	-	-	-	-	-	-
2023 Nn4sys	193	UNSAT	188	-	-	-	-	-	-
2023 Tllverifybench	0	UNSAT	0.97	8.03	13.5	2.43	-	399	0.01
2023 Tllverifybench	1	UNSAT	1.04	7.03	7.68	2.46	1.51	-	0.15
2023 Tllverifybench	2	UNSAT	0.06	<0.01	0.26	0.02	0.49	1.23	<0.01
2023 Tllverifybench	3	UNSAT	0.97	7.02	6.49	2.43	1.52	-	0.11
2023 Tllverifybench	4	UNSAT	0.11	<0.01	0.32	0.06	0.71	0.71	0.04
2023 Tllverifybench	5	UNSAT	0.37	<0.01	-	0.19	0.96	0.21	0.19
2023 Tllverifybench	6	UNSAT	0.07	<0.01	0.35	0.05	0.64	<0.01	0.08
2023 Tllverifybench	7	UNSAT	0.09	<0.01	0.32	0.03	0.80	0.09	0.07
2023 Tllverifybench	8	UNSAT	0.26	<0.01	0.40	0.09	0.53	0.40	0.01
2023 Tllverifybench	9	UNSAT	1.82	-	26.7	2.93	-	-	0.08
2023 Tllverifybench	10	UNSAT	1.47	29.3	9.50	2.96	-	-	0.06
2023 Tllverifybench	11	UNSAT	3.24	-	181	3.14	-	-	0.01
2023 Tllverifybench	12	UNSAT	0.66	<0.01	-	0.23	1.71	1.50	0.22
2023 Tllverifybench	13	UNSAT	2.34	-	149	3.16	-	-	0.01
2023 Tllverifybench	14	UNSAT	1.87	-	48.0	3.09	-	-	0.16
2023 Tllverifybench	15	UNSAT	0.66	<0.01	0.63	0.12	1.00	0.51	0.07
2023 Tllverifybench	16	UNSAT	1.64	<0.01	1.10	0.36	3.34	0.93	0.09
2023 Tllverifybench	17	UNSAT	4.87	185	94.4	5.11	-	-	0.08
2023 Tllverifybench	18	UNSAT	10.9	-	-	5.50	-	-	0.02
2023 Tllverifybench	19	UNSAT	5.62	-	250	5.32	-	-	0.41
2023 Tllverifybench	20	UNSAT	3.28	<0.01	-	0.73	7.47	0.74	0.45
2023 Tllverifybench	21	UNSAT	3.26	<0.01	2.60	0.72	7.49	0.66	0.20
2023 Tllverifybench	22	UNSAT	9.03	-	-	7.03	-	-	<0.01
2023 Tllverifybench	23	UNSAT	3.24	<0.01	1.81	0.74	5.21	1.11	0.02
2023 Tllverifybench	24	UNSAT	18.9	-	-	44.9	-	-	0.03
2023 Tllverifybench	25	UNSAT	6.07	<0.01	4.07	1.40	11.1	1.16	0.03
2023 Tllverifybench	26	UNSAT	6.22	<0.01	3.67	1.40	11.3	0.95	0.03
2023 Tllverifybench	27	UNSAT	6.05	0.01	3.63	1.43	12.3	1.31	0.10
2023 Tllverifybench	28	UNSAT	16.6	-	-	70.0	-	-	0.27
2023 Tllverifybench	29	UNSAT	10.6	1.01	-	2.38	-	-	0.01
2023 Tllverifybench	30	UNSAT	10.2	1.02	-	2.39	25.1	1.99	0.54
2023 Tllverifybench	31	UNSAT	16.9	-	-	72.8	-	-	0.63
2023 Traffic Signs Recognition	0	UNSAT	1.90	0.01	1.01	-	-	-	-
2023 Traffic Signs Recognition	1	UNSAT	1.91	<0.01	1.01	-	-	-	-
2023 Traffic Signs Recognition	2	UNSAT	1.91	<0.01	0.95	-	-	-	-
2023 Traffic Signs Recognition	3	UNSAT	1.90	<0.01	1.02	-	-	-	-
2023 Traffic Signs Recognition	4	UNSAT	1.91	0.01	0.94	-	-	-	-
2023 Traffic Signs Recognition	5	?	-	-	-	-	-	-	-
2023 Traffic Signs Recognition	6	?	-	-	-	-	-	-	-
2023 Traffic Signs Recognition	7	UNSAT	2.55	-	-	-	-	-	-
2023 Traffic Signs Recognition	8	UNSAT	2.28	-	-	-	-	-	-
2023 Traffic Signs Recognition	9	UNSAT	2.18	-	-	-	-	-	-
2023 Traffic Signs Recognition	10	UNSAT	2.24	-	-	✗	-	-	-
2023 Traffic Signs Recognition	11	UNSAT	2.31	47.3	-	✗	-	-	-
2023 Traffic Signs Recognition	12	UNSAT	2.22	7.06	-	✗	-	-	-
2023 Traffic Signs Recognition	13	UNSAT	1.92	<0.01	-	✗	-	-	-
2023 Traffic Signs Recognition	14	UNSAT	1.94	<0.01	-	✗	-	-	-
2023 Traffic Signs Recognition	15	UNSAT	2.83	-	-	✗	-	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Traffic Signs Recognition	16	UNSAT	2.42	-	-	✗	-	-	-
2023 Traffic Signs Recognition	17	UNSAT	2.34	-	-	✗	-	-	-
2023 Traffic Signs Recognition	18	UNSAT	2.35	-	-	✗	-	-	-
2023 Traffic Signs Recognition	19	UNSAT	2.36	276	-	✗	-	-	-
2023 Traffic Signs Recognition	20	UNSAT	2.50	-	-	✗	-	-	-
2023 Traffic Signs Recognition	21	UNSAT	2.33	-	-	✗	-	-	-
2023 Traffic Signs Recognition	22	UNSAT	2.30	-	-	✗	-	-	-
2023 Traffic Signs Recognition	23	UNSAT	2.33	-	-	✗	-	-	-
2023 Traffic Signs Recognition	24	UNSAT	2.33	-	-	✗	-	-	-
2023 Traffic Signs Recognition	25	UNSAT	2.03	<0.01	-	✗	-	-	-
2023 Traffic Signs Recognition	26	UNSAT	2.04	<0.01	3.67	✗	-	-	-
2023 Traffic Signs Recognition	27	UNSAT	2.05	<0.01	-	✗	-	-	-
2023 Traffic Signs Recognition	28	UNSAT	1.99	<0.01	3.66	✗	-	-	-
2023 Traffic Signs Recognition	29	UNSAT	1.99	<0.01	3.63	✗	-	-	-
2023 Traffic Signs Recognition	30	UNSAT	2.63	-	-	✗	-	-	-
2023 Traffic Signs Recognition	31	UNSAT	2.60	-	-	✗	-	-	-
2023 Traffic Signs Recognition	32	UNSAT	2.59	-	-	✗	-	-	-
2023 Traffic Signs Recognition	33	UNSAT	2.54	-	-	✗	-	-	-
2023 Traffic Signs Recognition	34	UNSAT	2.56	9.07	-	✗	-	-	-
2023 Traffic Signs Recognition	35	UNSAT	-	-	-	✗	-	-	-
2023 Traffic Signs Recognition	36	UNSAT	3.39	-	-	✗	-	-	-
2023 Traffic Signs Recognition	37	UNSAT	3.04	-	-	✗	-	-	-
2023 Traffic Signs Recognition	38	UNSAT	2.80	-	-	✗	-	-	-
2023 Traffic Signs Recognition	39	UNSAT	2.63	-	-	✗	-	-	-
2023 Traffic Signs Recognition	40	UNSAT	2.57	-	-	✗	-	-	-
2023 Traffic Signs Recognition	41	UNSAT	2.62	-	-	✗	-	-	-
2023 Traffic Signs Recognition	42	UNSAT	2.59	121	-	✗	-	-	-
2023 Traffic Signs Recognition	43	UNSAT	2.58	21.1	-	✗	-	-	-
2023 Traffic Signs Recognition	44	UNSAT	2.21	1.01	-	✗	-	-	-
2023 Vggnet16	0	UNSAT	9.46	30.5	63.1	50.9	33.2	-	-
2023 Vggnet16	1	UNSAT	9.41	211	63.1	51.0	29.7	-	-
2023 Vggnet16	2	UNSAT	9.40	-	68.2	-	33.0	-	-
2023 Vggnet16	3	UNSAT	9.27	-	69.7	50.9	30.9	-	-
2023 Vggnet16	4	UNSAT	9.28	-	73.3	-	37.0	-	-
2023 Vggnet16	5	UNSAT	9.36	-	85.5	-	52.7	-	-
2023 Vggnet16	6	UNSAT	9.30	-	75.8	50.8	32.0	-	-
2023 Vggnet16	7	UNSAT	9.41	-	77.1	-	42.4	-	-
2023 Vggnet16	8	UNSAT	11.1	-	228	-	121	-	-
2023 Vggnet16	9	UNSAT	9.45	-	88.2	51.2	36.8	-	-
2023 Vggnet16	10	UNSAT	9.50	-	101	-	60.8	-	-
2023 Vggnet16	11	UNSAT	9.43	-	115	-	68.3	-	-
2023 Vggnet16	12	UNSAT	783	-	-	-	-	-	-
2023 Vggnet16	13	UNSAT	20.7	-	287	-	229	-	-
2023 Vggnet16	14	UNSAT	41.8	-	-	-	632	-	-
2023 Vggnet16	15	UNSAT	-	3.03	33.7	36.1	-	-	-
2023 Vggnet16	16	?	-	-	-	-	-	-	-
2023 Vggnet16	17	?	-	-	-	-	-	-	-
2023 Vit	0	UNSAT	-	12.2	-	-	-	-	-
2023 Vit	1	UNSAT	-	12.2	-	-	-	-	-
2023 Vit	2	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	3	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	4	UNSAT	-	13.1	-	-	-	-	-
2023 Vit	5	UNSAT	9.08	14.2	-	-	-	-	-
2023 Vit	6	UNSAT	7.86	13.1	-	-	-	-	-
2023 Vit	7	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	8	UNSAT	-	14.3	-	-	-	-	-
2023 Vit	9	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	10	UNSAT	8.23	13.3	-	-	-	-	-
2023 Vit	11	UNSAT	7.80	13.1	-	-	-	-	-
2023 Vit	12	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	13	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	14	UNSAT	-	14.1	-	-	-	-	-
2023 Vit	15	UNSAT	11.3	13.3	-	-	-	-	-
2023 Vit	16	UNSAT	7.40	13.3	-	-	-	-	-
2023 Vit	17	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	18	UNSAT	17.8	13.2	-	-	-	-	-
2023 Vit	19	UNSAT	-	13.2	-	-	-	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Vit	20	UNSAT	-	13.3	-	-	-	-	-
2023 Vit	21	UNSAT	-	13.1	-	-	-	-	-
2023 Vit	22	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	23	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	24	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	25	UNSAT	9.17	13.2	-	-	-	-	-
2023 Vit	26	UNSAT	19.7	13.2	-	-	-	-	-
2023 Vit	27	UNSAT	-	14.3	-	-	-	-	-
2023 Vit	28	UNSAT	-	13.3	-	-	-	-	-
2023 Vit	29	UNSAT	-	14.3	-	-	-	-	-
2023 Vit	30	UNSAT	-	14.3	-	-	-	-	-
2023 Vit	31	UNSAT	-	12.2	-	-	-	-	-
2023 Vit	32	UNSAT	7.42	14.2	-	-	-	-	-
2023 Vit	33	UNSAT	9.54	13.2	-	-	-	-	-
2023 Vit	34	UNSAT	-	14.2	-	-	-	-	-
2023 Vit	35	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	36	UNSAT	9.99	13.2	-	-	-	-	-
2023 Vit	37	UNSAT	-	13.3	-	-	-	-	-
2023 Vit	38	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	39	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	40	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	41	UNSAT	-	14.2	-	-	-	-	-
2023 Vit	42	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	43	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	44	UNSAT	-	14.2	-	-	-	-	-
2023 Vit	45	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	46	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	47	UNSAT	-	13.3	-	-	-	-	-
2023 Vit	48	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	49	UNSAT	6.94	13.2	-	-	-	-	-
2023 Vit	50	UNSAT	20.8	13.2	-	-	-	-	-
2023 Vit	51	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	52	UNSAT	11.7	13.3	-	-	-	-	-
2023 Vit	53	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	54	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	55	UNSAT	14.0	14.2	-	-	-	-	-
2023 Vit	56	UNSAT	8.19	13.2	-	-	-	-	-
2023 Vit	57	UNSAT	11.3	14.2	-	-	-	-	-
2023 Vit	58	UNSAT	-	14.2	-	-	-	-	-
2023 Vit	59	UNSAT	9.13	13.2	-	-	-	-	-
2023 Vit	60	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	61	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	62	UNSAT	18.7	13.1	-	-	-	-	-
2023 Vit	63	UNSAT	9.56	14.2	-	-	-	-	-
2023 Vit	64	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	65	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	66	UNSAT	9.13	14.2	-	-	-	-	-
2023 Vit	67	UNSAT	10.4	13.2	-	-	-	-	-
2023 Vit	68	UNSAT	-	13.3	-	-	-	-	-
2023 Vit	69	UNSAT	28.4	13.2	-	-	-	-	-
2023 Vit	70	UNSAT	7.04	14.2	-	-	-	-	-
2023 Vit	71	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	72	UNSAT	10.5	13.2	-	-	-	-	-
2023 Vit	73	UNSAT	7.00	13.2	-	-	-	-	-
2023 Vit	74	UNSAT	-	13.3	-	-	-	-	-
2023 Vit	75	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	76	UNSAT	19.3	13.1	-	-	-	-	-
2023 Vit	77	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	78	UNSAT	8.31	13.2	-	-	-	-	-
2023 Vit	79	UNSAT	7.46	14.2	-	-	-	-	-
2023 Vit	80	UNSAT	10.4	13.2	-	-	-	-	-
2023 Vit	81	UNSAT	9.62	13.3	-	-	-	-	-
2023 Vit	82	UNSAT	10.5	12.2	-	-	-	-	-
2023 Vit	83	UNSAT	8.64	13.3	-	-	-	-	-
2023 Vit	84	UNSAT	6.98	13.1	-	-	-	-	-
2023 Vit	85	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	86	UNSAT	9.52	13.2	-	-	-	-	-
2023 Vit	87	UNSAT	7.83	13.2	-	-	-	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Vit	88	UNSAT	9.07	14.2	-	-	-	-	-
2023 Vit	89	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	90	UNSAT	9.12	13.2	-	-	-	-	-
2023 Vit	91	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	92	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	93	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	94	UNSAT	-	14.3	-	-	-	-	-
2023 Vit	95	UNSAT	-	14.2	-	-	-	-	-
2023 Vit	96	UNSAT	20.3	14.2	-	-	-	-	-
2023 Vit	97	UNSAT	8.69	13.2	-	-	-	-	-
2023 Vit	98	UNSAT	-	13.2	-	-	-	-	-
2023 Vit	99	UNSAT	8.68	13.2	-	-	-	-	-
2023 Vit	100	UNSAT	-	60.2	-	-	-	-	-
2023 Vit	101	UNSAT	-	60.2	-	-	-	-	-
2023 Vit	102	UNSAT	-	58.1	-	-	-	-	-
2023 Vit	103	UNSAT	43.0	60.2	-	-	-	-	-
2023 Vit	104	UNSAT	35.7	58.1	-	-	-	-	-
2023 Vit	105	UNSAT	33.2	59.9	-	-	-	-	-
2023 Vit	106	UNSAT	-	58.0	-	-	-	-	-
2023 Vit	107	UNSAT	15.2	59.1	-	-	-	-	-
2023 Vit	108	UNSAT	-	61.1	-	-	-	-	-
2023 Vit	109	UNSAT	39.1	58.0	-	-	-	-	-
2023 Vit	110	UNSAT	-	53.9	-	-	-	-	-
2023 Vit	111	UNSAT	31.8	60.1	-	-	-	-	-
2023 Vit	112	UNSAT	12.2	59.2	-	-	-	-	-
2023 Vit	113	UNSAT	-	57.1	-	-	-	-	-
2023 Vit	114	UNSAT	-	58.0	-	-	-	-	-
2023 Vit	115	UNSAT	-	60.0	-	-	-	-	-
2023 Vit	116	UNSAT	-	58.0	-	-	-	-	-
2023 Vit	117	UNSAT	-	59.0	-	-	-	-	-
2023 Vit	118	UNSAT	-	61.1	-	-	-	-	-
2023 Vit	119	UNSAT	10.1	61.1	-	-	-	-	-
2023 Vit	120	UNSAT	68.2	59.1	-	-	-	-	-
2023 Vit	121	UNSAT	-	60.0	-	-	-	-	-
2023 Vit	122	UNSAT	30.3	59.0	-	-	-	-	-
2023 Vit	123	UNSAT	31.7	58.0	-	-	-	-	-
2023 Vit	124	UNSAT	10.0	58.2	-	-	-	-	-
2023 Vit	125	UNSAT	-	58.3	-	-	-	-	-
2023 Vit	126	UNSAT	-	54.9	-	-	-	-	-
2023 Vit	127	UNSAT	21.3	59.2	-	-	-	-	-
2023 Vit	128	UNSAT	-	58.1	-	-	-	-	-
2023 Vit	129	UNSAT	-	57.1	-	-	-	-	-
2023 Vit	130	UNSAT	-	59.0	-	-	-	-	-
2023 Vit	131	UNSAT	-	54.9	-	-	-	-	-
2023 Vit	132	UNSAT	-	59.2	-	-	-	-	-
2023 Vit	133	UNSAT	35.8	58.2	-	-	-	-	-
2023 Vit	134	UNSAT	-	59.0	-	-	-	-	-
2023 Vit	135	UNSAT	53.9	56.9	-	-	-	-	-
2023 Vit	136	UNSAT	9.04	56.0	-	-	-	-	-
2023 Vit	137	UNSAT	-	60.0	-	-	-	-	-
2023 Vit	138	UNSAT	-	59.2	-	-	-	-	-
2023 Vit	139	UNSAT	-	57.9	-	-	-	-	-
2023 Vit	140	UNSAT	34.7	58.0	-	-	-	-	-
2023 Vit	141	UNSAT	-	57.0	-	-	-	-	-
2023 Vit	142	UNSAT	15.2	58.2	-	-	-	-	-
2023 Vit	143	UNSAT	-	57.9	-	-	-	-	-
2023 Vit	144	UNSAT	-	59.1	-	-	-	-	-
2023 Vit	145	UNSAT	-	57.2	-	-	-	-	-
2023 Vit	146	UNSAT	11.1	58.0	-	-	-	-	-
2023 Vit	147	UNSAT	-	54.9	-	-	-	-	-
2023 Vit	148	UNSAT	-	61.3	-	-	-	-	-
2023 Vit	149	UNSAT	34.0	59.0	-	-	-	-	-
2023 Vit	150	UNSAT	-	55.9	-	-	-	-	-
2023 Vit	151	UNSAT	-	59.1	-	-	-	-	-
2023 Vit	152	UNSAT	-	56.1	-	-	-	-	-
2023 Vit	153	UNSAT	14.2	57.9	-	-	-	-	-
2023 Vit	154	UNSAT	-	62.0	-	-	-	-	-
2023 Vit	155	UNSAT	12.2	61.3	-	-	-	-	-

Table 41: Instance Runtimes. Fastest times are blue. Second fastest are green. Penalties are red crosses (✗).

Category	Id	Result	α-β-C	Marab	PyRat	NSAT	NNen	NNV	FastBaT
2023 Vit	156	UNSAT	34.1	59.0	-	-	-	-	-
2023 Vit	157	UNSAT	34.8	59.9	-	-	-	-	-
2023 Vit	158	UNSAT	-	58.8	-	-	-	-	-
2023 Vit	159	UNSAT	-	57.0	-	-	-	-	-
2023 Vit	160	UNSAT	40.6	58.0	-	-	-	-	-
2023 Vit	161	UNSAT	-	59.2	-	-	-	-	-
2023 Vit	162	UNSAT	12.3	60.1	-	-	-	-	-
2023 Vit	163	UNSAT	-	58.1	-	-	-	-	-
2023 Vit	164	UNSAT	-	60.1	-	-	-	-	-
2023 Vit	165	UNSAT	31.8	55.9	-	-	-	-	-
2023 Vit	166	UNSAT	-	57.1	-	-	-	-	-
2023 Vit	167	UNSAT	-	60.0	-	-	-	-	-
2023 Vit	168	UNSAT	-	61.0	-	-	-	-	-
2023 Vit	169	UNSAT	-	61.1	-	-	-	-	-
2023 Vit	170	UNSAT	-	57.0	-	-	-	-	-
2023 Vit	171	UNSAT	-	60.1	-	-	-	-	-
2023 Vit	172	UNSAT	18.3	59.1	-	-	-	-	-
2023 Vit	173	UNSAT	-	60.2	-	-	-	-	-
2023 Vit	174	UNSAT	-	57.9	-	-	-	-	-
2023 Vit	175	UNSAT	14.3	56.8	-	-	-	-	-
2023 Vit	176	UNSAT	14.2	57.8	-	-	-	-	-
2023 Vit	177	UNSAT	-	57.1	-	-	-	-	-
2023 Vit	178	UNSAT	9.13	59.1	-	-	-	-	-
2023 Vit	179	UNSAT	-	60.0	-	-	-	-	-
2023 Vit	180	UNSAT	-	60.1	-	-	-	-	-
2023 Vit	181	UNSAT	-	57.0	-	-	-	-	-
2023 Vit	182	UNSAT	-	62.1	-	-	-	-	-
2023 Vit	183	UNSAT	12.2	56.1	-	-	-	-	-
2023 Vit	184	UNSAT	33.3	59.0	-	-	-	-	-
2023 Vit	185	UNSAT	17.4	56.0	-	-	-	-	-
2023 Vit	186	UNSAT	-	59.1	-	-	-	-	-
2023 Vit	187	UNSAT	-	60.1	-	-	-	-	-
2023 Vit	188	UNSAT	-	58.0	-	-	-	-	-
2023 Vit	189	UNSAT	-	59.2	-	-	-	-	-
2023 Vit	190	UNSAT	15.3	56.9	-	-	-	-	-
2023 Vit	191	UNSAT	-	59.1	-	-	-	-	-
2023 Vit	192	UNSAT	-	56.9	-	-	-	-	-
2023 Vit	193	UNSAT	11.2	61.3	-	-	-	-	-
2023 Vit	194	UNSAT	-	57.0	-	-	-	-	-
2023 Vit	195	UNSAT	-	56.9	-	-	-	-	-
2023 Vit	196	UNSAT	-	60.1	-	-	-	-	-
2023 Vit	197	UNSAT	-	59.2	-	-	-	-	-
2023 Vit	198	UNSAT	-	53.8	-	-	-	-	-
2023 Vit	199	UNSAT	34.2	56.0	-	-	-	-	-