Assessing Progress in SAT Solvers Through the Lens of Incremental SAT

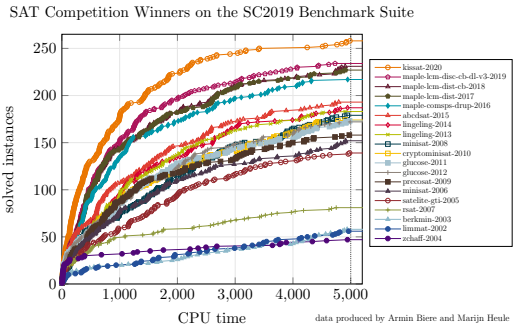
Stepan Kochemazov[1] Alexey Ignatiev[2] Joao Marques-Silva[3]

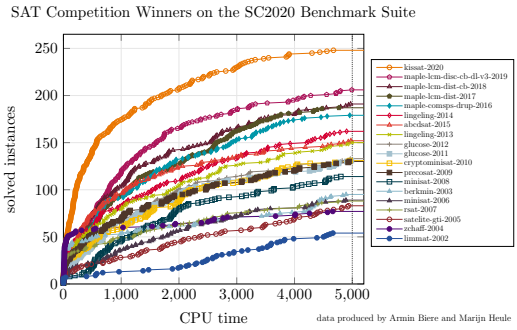
ITMO University, St. Petersburg, Russia
veinamond@gmail.com

Monash University, Melbourne, Australia
alexey.ignatiev@monash.edu

IRIT, CNRS, Toulouse, France
joao.marques-silva@irit.fr

SAT2021



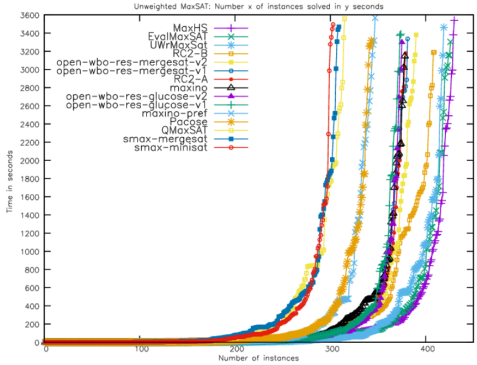


- The winners of recent SAT Competitions achieved a great progress during the last several years[1].

[1] The plots are courtesy of Armin Biere (`http://fmv.jku.at/kissat/`)

- Nevertheless many applications which employ incremental SAT solvers (e.g. MaxSAT solvers) prefer to use the good old MiniSAT 2.2 (2008) or Glucose 3 (2013)/ Glucose 4 (2014).
  - Example: MaxSAT Evaluation 2020 participants [2]:

Algorithm	Backend
UWrMaxSAT	CoMiniSATPS
MaxHS smax	MiniSat 2.2
open-wbo-v1,v2 smax	MergeSAT
RC2-B,A open-wbo-v1,v2 maxino, maxino-pref Pacose QMaxSAT	Glucose



[2] https://maxsat-evaluations.github.io/2020/

Goal: to understand how significant the recent progress in SAT solvers is to practical incremental SAT solving.

Goal: to understand how significant the recent progress in SAT solvers is to practical incremental SAT solving.

How will we do it?

- Take a CDCL solver that won at the *Main Track* of a recent SAT competition.

Goal: to understand how significant the recent progress in SAT solvers is to practical incremental SAT solving.

How will we do it?

- Take a CDCL solver that won at the *Main Track* of a recent SAT competition.
- Test it in incremental SAT applications.
  - MaxSAT solving
    - Satisfiability-based.
    - Unsatisfiability-based.

Goal: to understand how significant the recent progress in SAT solvers is to practical incremental SAT solving.

How will we do it?

- Take a CDCL solver that won at the *Main Track* of a recent SAT competition.
- Test it in incremental SAT applications.
  - MaxSAT solving
    - Satisfiability-based.
    - Unsatisfiability-based.
  - Minimum unsatisfiable subset (MUS) extraction.

Goal: to understand how significant the recent progress in SAT solvers is to practical incremental SAT solving.

How will we do it?

- Take a CDCL solver that won at the *Main Track* of a recent SAT competition.
- Test it in incremental SAT applications.
  - MaxSAT solving
    - Satisfiability-based.
    - Unsatisfiability-based.
  - Minimum unsatisfiable subset (MUS) extraction.
- Compare its performance with that of MiniSAT / Glucose.

- **2015.** COMiniSatPS combined *luby* and *Glucose*-style restarts with separate VSIDS activity scores for each mode and introduced three-tiered approach to storing learnt clauses.

- **2015.** COMiniSatPS combined *luby* and *Glucose*-style restarts with separate VSIDS activity scores for each mode and introduced three-tiered approach to storing learnt clauses.
- **2016.** MapleCOMSPS replaced VSIDS activity scores in the mode with luby restarts in COMiniSatPS by *Learning Rate Branching* (LRB) scores and changed the scheme for switching between modes.

- **2015.** COMiniSatPS combined *luby* and *Glucose*-style restarts with separate VSIDS activity scores for each mode and introduced three-tiered approach to storing learnt clauses.
- **2016.** MapleCOMSPS replaced VSIDS activity scores in the mode with luby restarts in COMiniSatPS by *Learning Rate Branching* (LRB) scores and changed the scheme for switching between modes.
- **2017. LCM & DIST**. MapleLCMDist extended MapleCOMSPS by *Learnt Clause Minimization* technique and a new *Distance* metric to evaluate variables' worth at the start of the search.

- **2015.** COMiniSatPS combined *luby* and *Glucose*-style restarts with separate VSIDS activity scores for each mode and introduced three-tiered approach to storing learnt clauses.
- **2016.** MapleCOMSPS replaced VSIDS activity scores in the mode with luby restarts in COMiniSatPS by *Learning Rate Branching* (LRB) scores and changed the scheme for switching between modes.
- **2017. LCM & DIST**. MapleLCMDist extended MapleCOMSPS by *Learnt Clause Minimization* technique and a new *Distance* metric to evaluate variables' worth at the start of the search.
- **2018. CB**. MapleLCMDistChronoBT added to MapleLCMDist the *chronological backtracking* heuristic.

- **2015.** COMiniSatPS combined *luby* and *Glucose*-style restarts with separate VSIDS activity scores for each mode and introduced three-tiered approach to storing learnt clauses.
- **2016.** MapleCOMSPS replaced VSIDS activity scores in the mode with luby restarts in COMiniSatPS by *Learning Rate Branching* (LRB) scores and changed the scheme for switching between modes.
- **2017. LCM & DIST**. MapleLCMDist extended MapleCOMSPS by *Learnt Clause Minimization* technique and a new *Distance* metric to evaluate variables' worth at the start of the search.
- **2018. CB**. MapleLCMDistChronoBT added to MapleLCMDist the *chronological backtracking* heuristic.
- **2019. DL**. MapleLCMDistChronoBT-DL-v3 implemented the *duplicate learnts* heuristic in MapleLCMDistChronoBT and retuned the scheme for switching between modes that remained untouched since MapleCOMSPS.

- **2015.** COMiniSatPS combined *luby* and *Glucose*-style restarts with separate VSIDS activity scores for each mode and introduced three-tiered approach to storing learnt clauses.
- **2016.** MapleCOMSPS replaced VSIDS activity scores in the mode with luby restarts in COMiniSatPS by *Learning Rate Branching* (LRB) scores and changed the scheme for switching between modes.
- **2017. LCM & DIST**. MapleLCMDist extended MapleCOMSPS by *Learnt Clause Minimization* technique and a new *Distance* metric to evaluate variables' worth at the start of the search.
- **2018. CB**. MapleLCMDistChronoBT added to MapleLCMDist the *chronological backtracking* heuristic.
- **2019. DL**. MapleLCMDistChronoBT-DL-v3 implemented the *duplicate learnts* heuristic in MapleLCMDistChronoBT and retuned the scheme for switching between modes that remained untouched since MapleCOMSPS.
- **2020. SLS**. Relaxed_LCMDCBDL_newTech incorporated into the solver a stochastic local search component, complemented with *rephasing* technique and a novel approach to bumping activity values in CDCL component based on some of the statistics accumulated by the SLS component.

What solver(s) to choose?

What solver(s) to choose?

- The solver chosen for evaluation must implement the heuristics that signified the progress in recent SAT Competitions and support incremental solving.

What solver(s) to choose?

- The solver chosen for evaluation must implement the heuristics that signified the progress in recent SAT Competitions and support incremental solving.
- Neither Kissat (2020), nor any of the winners of SAT Competitions 2016-2020 support incremental mode out of the box.

What solver(s) to choose?

- The solver chosen for evaluation must implement the heuristics that signified the progress in recent SAT Competitions and support incremental solving.
- Neither Kissat (2020), nor any of the winners of SAT Competitions 2016-2020 support incremental mode out of the box.
- But the winners of SAT competitions 2016-2019, as well as 2nd place at SAT Competition 2020 use the MiniSAT codebase and can be modified without major effort.

What solver(s) to choose?

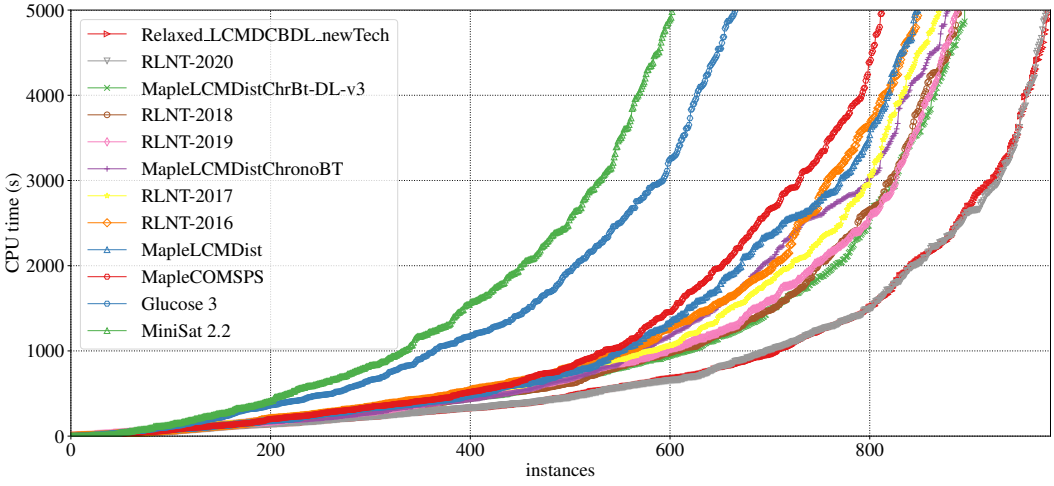
- The solver chosen for evaluation must implement the heuristics that signified the progress in recent SAT Competitions and support incremental solving.
- Neither Kissat (2020), nor any of the winners of SAT Competitions 2016-2020 support incremental mode out of the box.
- But the winners of SAT competitions 2016-2019, as well as 2nd place at SAT Competition 2020 use the MiniSAT codebase and can be modified without major effort.

RLNT

It was decided to develop a single solver based on Relaxed_LCMDCBDL_newTech, such that the modern heuristics can be separately disabled and enabled. The solver was called RLNT.

With all heuristics enabled it differs from Relaxed_LCMDCBDL_newTech in the support of the incremental mode and in several small fixes.

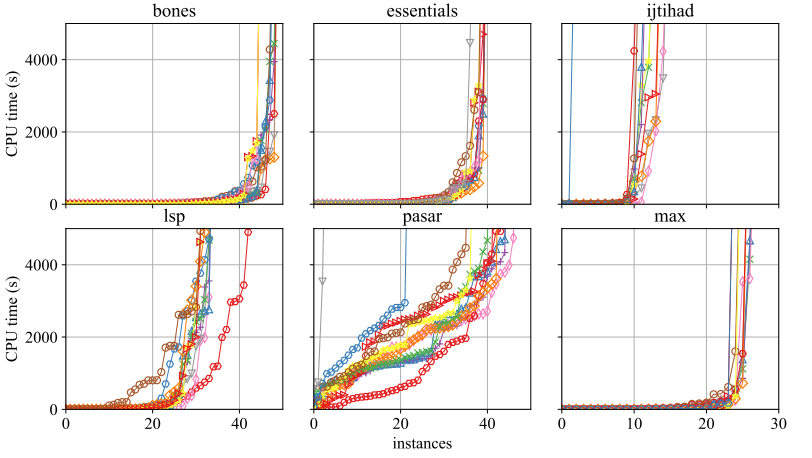
- RLNT-2020 – RLNT with DIST, LCM, CB, DL, SLS.
- RLNT-2019 – RLNT with DIST, LCM, CB, DL.
- RLNT-2018 – RLNT with DIST, LCM, CB.
- RLNT-2017 – RLNT with DIST, LCM.
- RLNT-2016 – RLNT without any of heuristics enabled.
- Relaxed_LCMDCBDL_newTech – SAT Competition 2020 2nd place.
- MapleLCMDistChronoBT-DL-v3 – SAT Race 2019 winner.
- MapleLCMDistChronoBT – SAT Competition 2018 winner.
- MapleLCMDist – SAT Competition 2017 winner.
- MapleCOMSPS – SAT Competition 2016 winner.
- Glucose 3.0.
- MiniSat 2.2.


Relaxed_LCMDCBDL_newTech
RLNT-2020
MapleLCMDistChrBt-DL-v3
RLNT-2018
RLNT-2019
MapleLCMDistChronoBT
RLNT-2017
RLNT-2016
MapleLCMDist
MapleCOMSPS
Glucose 3
MiniSat 2.2
CPU time (s)
instances
0
1000
2000
3000
4000
5000
0
200
400
600
800

Incremental track 2020 included the following applications:

- *bones*: finding backbones of a CNF SAT formula
- *essentials*: finding variables essential for the satisfiability of a formula
- *lsp*: finding the longest simple path in a graph
- *max*: a simple MaxSAT solver
- *ijtihad*: QBF solver
- *pasar*: planning solver

For each application, there were 50 benchmarks (*bones* and *essentials* used the same set of benchmarks), i.e. 250 benchmarks in total.


bones
essentials
ijtihad
lsp
pasar
max
CPU time (s)
CPU time (s)
0
2000
4000
0
20
40
0
20
40
0
10
20
30
instances
RLNT-2020
RLNT-2019
RLNT-2018
RLNT-2017
RLNT-2016
Riss-7.1.2
abcdsat_i20
cadicalsc2020
cryptominisat5
Glucose 3
MiniSat 2.2

	bones		*essentials*		*lsp*		*max*		*ijtihad*		*pasar*	
	S	P2	S	P2	S	P2	S	P2	S	P2	S	P2
RLNT-2020	49	350	40	2219	**43**	**1987**	26	4869	11	6879	44	2325
RLNT-2019	48	580	40	2170	34	3546	27	4753	12	6845	45	2733
RLNT-2018	49	482	40	2168	34	3577	**27**	**4739**	13	6393	41	3254
RLNT-2017	49	441	40	2174	34	3542	27	4745	12	6845	45	2553
RLNT-2016	**49**	**322**	**40**	**2086**	33	3837	26	4826	14	6369	43	2902
Glucose 3	45	1108	40	2273	32	3845	24	5211	14	6895	42	3387
MiniSat 2.2	48	635	40	2180	34	3811	24	5212	2	9600	22	6380
Riss-7.1.2	45	1108	39	2388	32	3844	25	5013	13	7270	37	3907
abcdsat_i20	48	627	39	2450	32	4205	25	4696	11	7830	36	4295
cadicalsc2020	45	1085	39	2323	34	3381	27	4756	15	6729	**47**	**2400**
cryptoMiniSat5	49	333	37	2737	34	3495	26	4478	**15**	**5966**	3	9496

- It is clear that RLNT configurations are on par or better than competition (see *lsp* column).
- Contrary to what is observed on main track benchmarks, the difference between modern solvers and MiniSAT/ Glucose becomes much less pronounced.

Next we use the 5 developed RLNT configurations (2016-2020) in two concrete practical settings where incremental calls to a SAT oracle are of crucial importance: *Maximum Satisfiability (MaxSAT) solving* and *Minimal unsatisfiable subset (MUS) extraction*, and compare them with Glucose 3 and MiniSAT 2.2.

Next we use the 5 developed RLNT configurations (2016-2020) in two concrete practical settings where incremental calls to a SAT oracle are of crucial importance: *Maximum Satisfiability (MaxSAT) solving* and *Minimal unsatisfiable subset (MUS) extraction*, and compare them with Glucose 3 and MiniSAT 2.2.

All solvers were integrated into PySAT framework and used in *unified* fashion via the same API.

Next we use the 5 developed RLNT configurations (2016-2020) in two concrete practical settings where incremental calls to a SAT oracle are of crucial importance: *Maximum Satisfiability (MaxSAT) solving* and *Minimal unsatisfiable subset (MUS) extraction*, and compare them with Glucose 3 and MiniSAT 2.2.

All solvers were integrated into PySAT framework and used in *unified* fashion via the same API. In the experiments we tested three practical problem solvers:

- Award-winning core-guided MaxSAT solver RC2 (configurations *RC2-A* and *RC2-B*)
  - *mostly unsatisfiable oracle calls*

Next we use the 5 developed RLNT configurations (2016-2020) in two concrete practical settings where incremental calls to a SAT oracle are of crucial importance: *Maximum Satisfiability (MaxSAT) solving* and *Minimal unsatisfiable subset (MUS) extraction*, and compare them with Glucose 3 and MiniSAT 2.2.

All solvers were integrated into PySAT framework and used in *unified* fashion via the same API. In the experiments we tested three practical problem solvers:

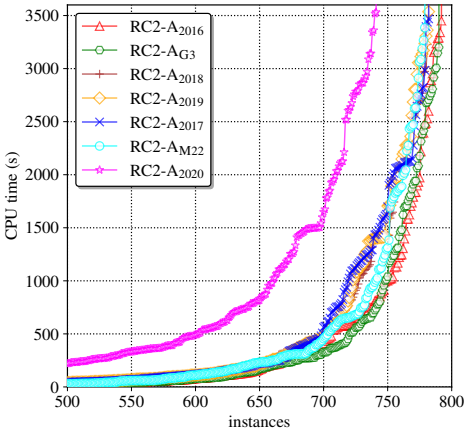
- Award-winning core-guided MaxSAT solver RC2 (configurations *RC2-A* and *RC2-B*)
  - *mostly unsatisfiable oracle calls*
- Linear search SAT-UNSAT algorithm for MaxSAT (*LSU*)
  - *mostly satisfiable oracle calls*

Next we use the 5 developed RLNT configurations (2016-2020) in two concrete practical settings where incremental calls to a SAT oracle are of crucial importance: *Maximum Satisfiability (MaxSAT) solving* and *Minimal unsatisfiable subset (MUS) extraction*, and compare them with Glucose 3 and MiniSAT 2.2.

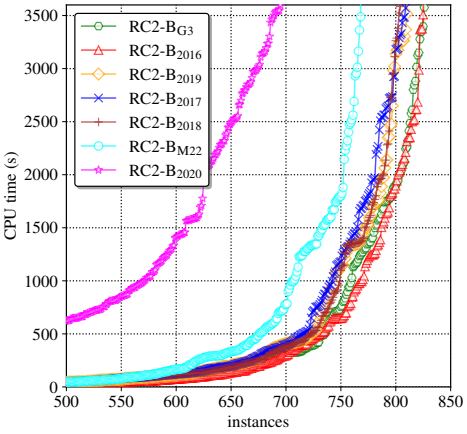
All solvers were integrated into PySAT framework and used in *unified* fashion via the same API. In the experiments we tested three practical problem solvers:

- Award-winning core-guided MaxSAT solver RC2 (configurations *RC2-A* and *RC2-B*)
  - *mostly unsatisfiable oracle calls*
- Linear search SAT-UNSAT algorithm for MaxSAT (*LSU*)
  - *mostly satisfiable oracle calls*
- A simple deletion-based MUS extractor (*MUSx*).
  - *mixed (satisfiable and unsatisfiable) oracle calls*

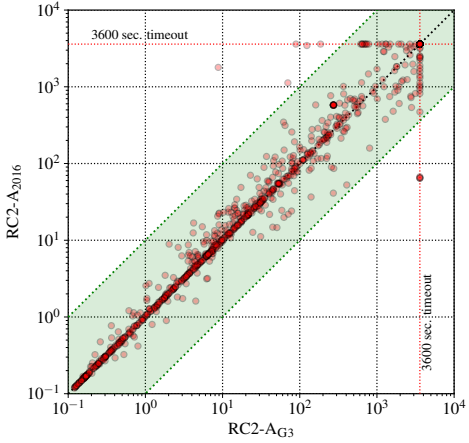
- The RC2 MaxSAT solver belongs to the large family of core-guided MaxSAT solvers and provides an efficient implementation of the OLL/RC2 algorithm.
- Each iteration performed by the solver involves calling a SAT oracle incrementally given an unsatisfiable formula that is slightly modified at each iteration.
- The solver proceeds until the final iteration, which determines the working formula to be satisfiable.
- The solver can also be instructed to apply a few additional heuristics, some of which may increase the number of satisfiable oracle calls; however, unsatisfiable oracle calls still prevail.
- The competition configurations RC2-A and RC2-B make use of the Glucose 3 SAT solver.
- This part of the experiment tested RC2 on the complete set of benchmarks (both unweighted and weighted) from the MSE'20.



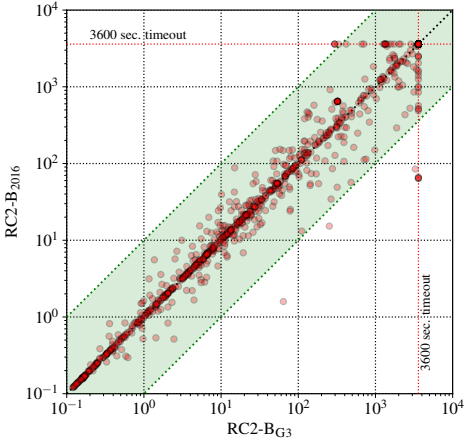
Performance of RC2-A



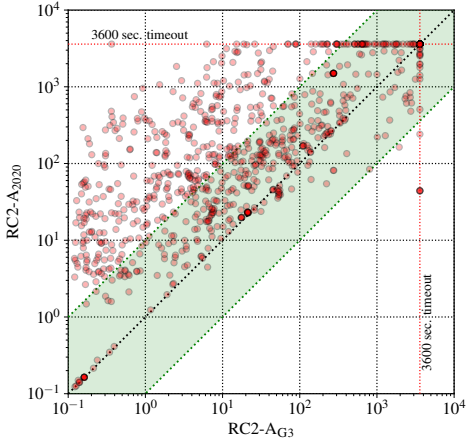
Performance of RC2-B



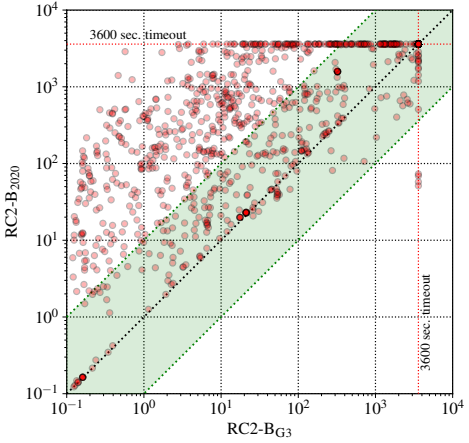
RC2-A_{G3} vs RC2-A_{2016} (best)



RC2-B_{G3} vs RC2-B_{2016} (best)



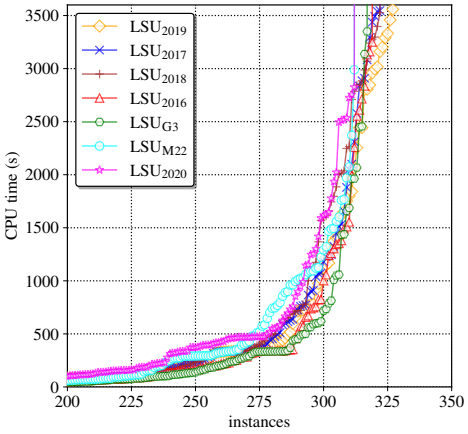
RC2-A_{G3} vs RC2-A_{2020} (worst)



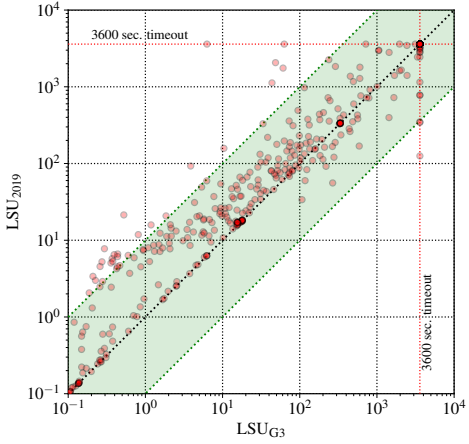
RC2-B_{G3} vs RC2-B_{2020} (worst)

- The LSU MaxSAT algorithm performs a linear search iterating over the possible numbers of satisfied soft clauses.
- It decreases this number as long as the underlying solver reports the current formula to be satisfiable.
- Thus, all but one iterations of the algorithm involve satisfiable oracle calls.

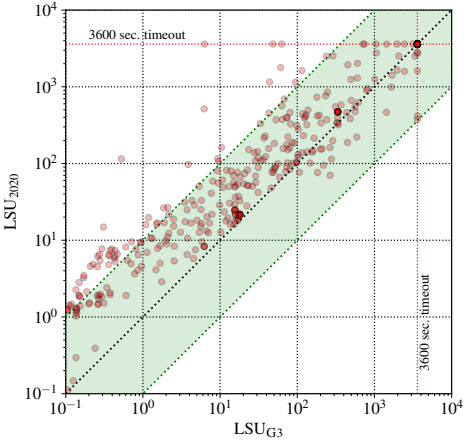
Note. Evaluation was made only using unweighted formulas.



Performance of LSU

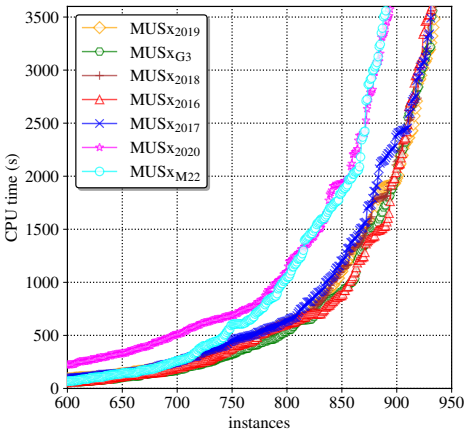


$\mathrm{LSU_{G3}}$ vs $\mathrm{LSU_{2019}}$ (best)

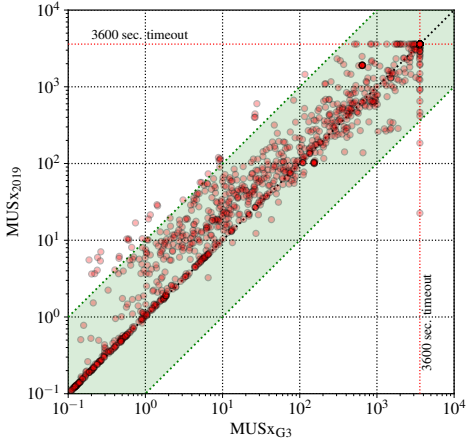


$\mathrm{LSU_{G3}}$ vs $\mathrm{LSU_{2020}}$ (worst)

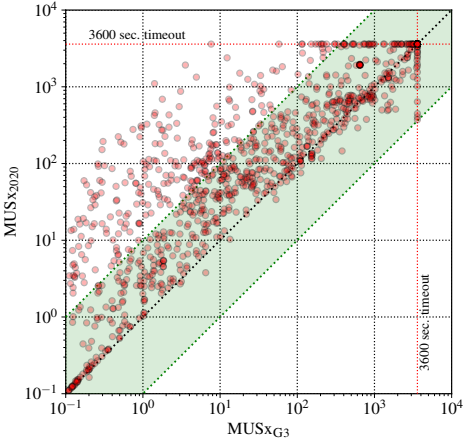
- MUSx implements the simple deletion-based algorithm, which is bootstrapped with an unsatisfiable core of a formula.
- It iterates over all clauses of the core trying to incrementally get rid of them one-by-one to get an MUS, so the outcomes of incremental SAT solver calls are mixed.
- We generated a large collection of new MUS benchmarks based on the MSE'20 benchmark set (1103 formulas in total).
- To ensure that the reduction phase computes exactly the same MUS, for each benchmark MUSx was bootstrapped with an initial unsatisfiable core, which was always obtained by Glucose 3.



Performance of MUSx



MUSx$_{G3}$ vs MUSx$_{2019}$ (best)



MUSx$_{G3}$ vs MUSx$_{2020}$ (worst)

	MiniSat 2.2	Glucose 3	RLNT-2016	RLNT-2017	RLNT-2018	RLNT-2019	RLNT-2020
RC2-A	2536.8	2466.3	**2462.5**	2544.6	2536.7	2543.9	2875.7
RC2-B	2628.1	2287.7	**2281.3**	2406.0	2406.1	2387.5	3272.8
LSU	3391.1	3307.4	3302.7	3306.3	3318.7	**3259.7**	3411.7
MUSx	1621.4	**1333.9**	1345.7	1386.3	1361.8	1353.8	1665.7
Overall	2303.2	2061.9	**2056.3**	2151.8	2153.2	2134.3	2754.0

	MiniSat 2.2	Glucose 3	RLNT-2016	RLNT-2017	RLNT-2018	RLNT-2019	RLNT-2020
RC2-A	2536.8	2466.3	**2462.5**	2544.6	2536.7	2543.9	2875.7
RC2-B	2628.1	2287.7	**2281.3**	2406.0	2406.1	2387.5	3272.8
LSU	3391.1	3307.4	3302.7	3306.3	3318.7	**3259.7**	3411.7
MUSx	1621.4	**1333.9**	1345.7	1386.3	1361.8	1353.8	1665.7
Overall	2303.2	2061.9	**2056.3**	2151.8	2153.2	2134.3	2754.0

- None of the tested configurations of RLNT brings any consistent (and significant) performance improvements to the considered problem solvers.

	MiniSat 2.2	Glucose 3	RLNT-2016	RLNT-2017	RLNT-2018	RLNT-2019	RLNT-2020
RC2-A	2536.8	2466.3	**2462.5**	2544.6	2536.7	2543.9	2875.7
RC2-B	2628.1	2287.7	**2281.3**	2406.0	2406.1	2387.5	3272.8
LSU	3391.1	3307.4	3302.7	3306.3	3318.7	**3259.7**	3411.7
MUSx	1621.4	**1333.9**	1345.7	1386.3	1361.8	1353.8	1665.7
Overall	2303.2	2061.9	**2056.3**	2151.8	2153.2	2134.3	2754.0

- None of the tested configurations of RLNT brings any consistent (and significant) performance improvements to the considered problem solvers.
- Most of the heuristics recently proposed for SAT solvers have no significant positive impact on the performance of practical problem solvers in the incremental setting, at least in the way they are implemented in the original *main track* solvers.

	MiniSat 2.2	Glucose 3	RLNT-2016	RLNT-2017	RLNT-2018	RLNT-2019	RLNT-2020
RC2-A	2536.8	2466.3	**2462.5**	2544.6	2536.7	2543.9	2875.7
RC2-B	2628.1	2287.7	**2281.3**	2406.0	2406.1	2387.5	3272.8
LSU	3391.1	3307.4	3302.7	3306.3	3318.7	**3259.7**	3411.7
MUSx	1621.4	**1333.9**	1345.7	1386.3	1361.8	1353.8	1665.7
Overall	2303.2	2061.9	**2056.3**	2151.8	2153.2	2134.3	2754.0

- None of the tested configurations of RLNT brings any consistent (and significant) performance improvements to the considered problem solvers.
- Most of the heuristics recently proposed for SAT solvers have no significant positive impact on the performance of practical problem solvers in the incremental setting, at least in the way they are implemented in the original *main track* solvers.
- The conclusions can be challenged if other uses of incremental SAT are considered.

	MiniSat 2.2	Glucose 3	RLNT-2016	RLNT-2017	RLNT-2018	RLNT-2019	RLNT-2020
RC2-A	2536.8	2466.3	**2462.5**	2544.6	2536.7	2543.9	2875.7
RC2-B	2628.1	2287.7	**2281.3**	2406.0	2406.1	2387.5	3272.8
LSU	3391.1	3307.4	3302.7	3306.3	3318.7	**3259.7**	3411.7
MUSx	1621.4	**1333.9**	1345.7	1386.3	1361.8	1353.8	1665.7
Overall	2303.2	2061.9	**2056.3**	2151.8	2153.2	2134.3	2754.0

- None of the tested configurations of RLNT brings any consistent (and significant) performance improvements to the considered problem solvers.
- Most of the heuristics recently proposed for SAT solvers have no significant positive impact on the performance of practical problem solvers in the incremental setting, at least in the way they are implemented in the original *main track* solvers.
- The conclusions can be challenged if other uses of incremental SAT are considered.
- There is a need for discussion in the SAT community to bridge the gap between practical applications of SAT solvers and the SAT Competition.

Thank you for your attention!