Chasing Target Phases

http://fmv.jku.at/chasing-target-phases

Armin Biere and Mathias Fleury

Pragmatics of SAT 2020, 2020/07/03

CDCL

CDCL decision heuristics:

- select and focus on interesting variables — (E)VSIDS, VMTF, LRB
- set phase (polarity) of decision variable — false, occurrences, phase saving

This talk is about:

- rephasing saved phases — experiments in RISS, “flipping” in STRANGENIGHT
- maximizing the trail — similar in spirit to GLUCOSE style restart blocking

which helped CADICAL to solve the largest number of instances at the SAT Race 2019

SAT as Optimization

Most important heuristics for SAT instances: phase heuristics

Pick any variable and set it to the “right” phase Hans van Maaren

New view for CDCL: maximize the trail

trail = current partial assigment

Objective is to maximize the size of the trail without conflict

Save *maximum consistent trail* as target phases

Prioritize target phases for decisions over saved phases

Intensification: target phases and best phases

Diversification: rephasing

Phase Saving

Phase Saving

first in RSAT by Pipatsrisawat & Darwiche

Saving phases	as soon variable is assigned save its phase
Phase heuristic	set decision phase to saved phase
Initialization	use arbitrary initial value (*false* in MINISAT)
Components	saves assignment of satisfied components
Rapid restarts	works well with (allows) rapid restarts

Rephasing Saved Phases

Rephasing

Reset saved and target phases in increasing conflict intervals:

Original O:	set phases to original value	(*false* or *true*)
Inverted I:	set phases to opposite of the original value	
Best B:	restore best assignment	
Walk W:	let local search maximize satisfied clauses	
Random #:	set phase to random value	
Flipped F:	flip current phase	

Used policy: $\texttt{OI}\,(\texttt{BWO}\;\texttt{BWI}\;\texttt{BW\#}\;\texttt{BWF})^{\omega}$ KISSAT

emphasize best phases and local search phases

Rephasing

Reset saved and target phases in increasing conflict intervals:

Original O:	set phases to original value	(*false* or *true*)
Inverted I:	set phases to opposite of the original value	
Best B:	restore best assignment	
Walk W:	let local search maximize satisfied clauses	
Random #:	set phase to random value	
Flipped F:	flip current phase	

Used policy: $\texttt{OI}\,(\texttt{BWO}\ \texttt{BWI}\ \texttt{BW\#}\ \texttt{BWF})^{\omega}$ Kissat

emphasize best phases and local search phases

Rephasing

Reset saved and target phases in increasing conflict intervals:

Original O:	set phases to original value (*false* or *true*)
Inverted I:	set phases to opposite of the original value
Best B:	restore best assignment
Walk W:	let local search maximize satisfied clauses
Random #:	set phase to random value
Flipped F:	flip current phase

Used policy: OI (BWO BWI BW# BWF)$^\omega$ KISSAT

emphasize best phases and local search phases

Rephasing

Reset saved and target phases in increasing conflict intervals:

Original O:	set phases to original value	(*false* or *true*)
Inverted I:	set phases to opposite of the original value	
Best B:	restore best assignment	
Walk W:	let local search maximize satisfied clauses	
Random #:	set phase to random value	
Flipped F:	flip current phase	

Used policy: OI (BWO BWI BW# BWF)$^\omega$ KISSAT

emphasize best phases and local search phases

Rephasing

Reset saved and target phases in increasing conflict intervals:

Original O:	set phases to original value	(*false* or *true*)
Inverted I:	set phases to opposite of the original value	
Best B:	restore best assignment	
Walk W:	let local search maximize satisfied clauses	
Random #:	set phase to random value	
Flipped F:	flip current phase	

Used policy: OI (BWO BWI BW# BWF)$^{\omega}$ Kissat

emphasize best phases and local search phases

Rephasing

Reset saved and target phases in increasing conflict intervals:

Original O:	set phases to original value	(*false* or *true*)
Inverted I:	set phases to opposite of the original value	
Best B:	restore best assignment	
Walk W:	let local search maximize satisfied clauses	
Random #:	set phase to random value	
Flipped F:	flip current phase	

Used policy: OI (BWO BWI BW# BWF)$^{\omega}$ Kissat

emphasize best phases and local search phases

Rephasing

Reset saved and target phases in increasing conflict intervals:

Original O:	set phases to original value	(*false* or *true*)
Inverted I:	set phases to opposite of the original value	
Best B:	restore best assignment	
Walk W:	let local search maximize satisfied clauses	
Random #:	set phase to random value	
Flipped F:	flip current phase	

Used policy: OI (BWO BWI BW# BWF)$^{\omega}$ KISSAT

emphasize best phases and local search phases

Really Rephase All?

If formula falls apart into several disconnected components:

- focus on one component at a time — bumping heuristic
- solve components one by one — unless one component is UNSAT
- phase saving also saves models of satisfied components

Rephasing forgets satisfying assignments of components!

So KISSAT makes sure not to loose them: — this is not in CADICAL

- largest **autarky** of saved phases — fixpoint algorithm by Kullmann
- clauses satisfied by autarky eliminated
- pushed on the reconstruction stack

Really Rephase All?

If formula falls apart into several disconnected components:

- focus on one component at a time — bumping heuristic
- solve components one by one — unless one component is UNSAT
- phase saving also saves models of satisfied components

Rephasing forgets satisfying assignments of components!

So KISSAT makes sure not to loose them: — this is not in CADICAL

- largest **autarky** of saved phases — fixpoint algorithm by Kullmann
- clauses satisfied by autarky eliminated
- pushed on the reconstruction stack

Really Rephase All?

If formula falls apart into several disconnected components:

- focus on one component at a time — bumping heuristic
- solve components one by one — unless one component is UNSAT
- phase saving also saves models of satisfied components

Rephasing forgets satisfying assignments of components!

So KISSAT makes sure not to loose them: — this is not in CADICAL

- largest **autarky** of saved phases — fixpoint algorithm by Kullmann
- clauses satisfied by autarky eliminated
- pushed on the reconstruction stack

Target Phases

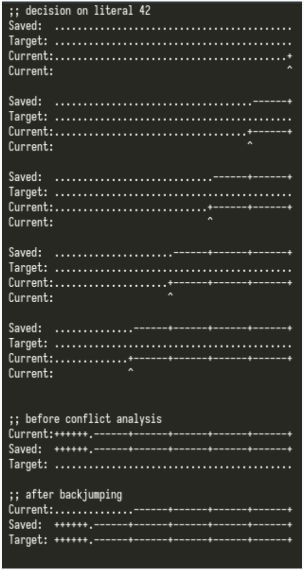
Maximize the Trail

Passive optimization in GLUCOSE:
Block restarts if trail shows steady size increase

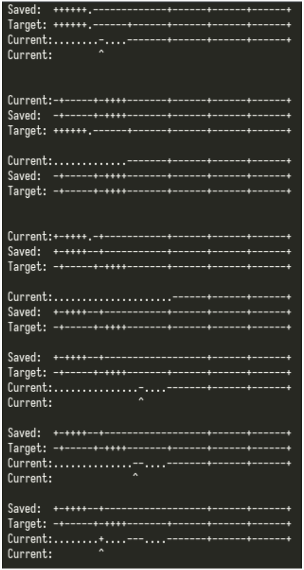
Using moving averages of trail size

Active optimization using target phases:
Use maximum consistent trail assignment for future decisions

Save target/best trail during backtracking only



conflict at the bottom



mismatch saved/target at the bottom

Demo: KISSAT stable / focused mode

```
c  seconds  reductions        redundant     trail        variables
c          MB      restarts            best       glue          remaining
c           level        conflicts        target    irredundant
c
c i 70.03 23 57 53 14758 431074 22014 57 532 40% 18 182398 28012 52%
c O 70.43 23 58 53 14840 435064 25762 57 539 40% 17 182389 28012 52%
c i 71.47 25 58 53 15483 440084 30607 57 552 40% 18 182389 28011 52%
c - 72.46 24 57 54 15869 445542 23296 57 554 40% 18 182372 28011 52%
c s 73.40 20 57 54 16377 450010 27608 57 554 40% 18 182367 28011 52%
c e 73.44 21 57 54 16377 450010 27608 57 554 40% 18 181317 27547 51%
c 2 73.44 21 57 54 16377 450010 27608 57 554 40% 18 181289 27547 51%
c s 73.44 21 57 54 16377 450010 27608 57 554 40% 18 181286 27547 51%
c e 73.45 21 57 54 16377 450010 27608 57 554 40% 18 181278 27538 51%
c } 74.60 25 58 54 16763 459745 31013 57 554 40% 18 181278 27538 51%
c [ 74.60 25 127 54 16763 459745 31013 57 554 43% 48 181278 27538 51%
c - 74.85 21 126 55 16765 462043 22072 57 554 43% 47 181278 27538 51%
c B 75.26 22 125 55 16767 465068 24885 57 555 43% 47 181278 27538 51%
c d 76.10 29 127 55 16772 475023 34237 57 556 44% 47 181278 27536 51%
c 3 76.13 26 127 55 16772 475023 34457 57 556 44% 47 181278 27536 51%
c t 76.26 26 127 55 16772 475023 34457 57 556 44% 47 181215 27536 51%
c f 76.52 27 127 55 16772 475023 100173 57 556 44% 47 181215 27532 51%
c u 76.78 28 127 55 16772 475023 99495 57 556 44% 47 181215 27532 51%
c v 76.86 28 127 55 16772 475023 99728 57 556 44% 47 181215 27532 51%
c w 77.00 29 127 55 16772 475023 100884 57 556 44% 47 181135 27532 51%
c
c  seconds  reductions        redundant     trail        variables
c          MB      restarts            best       glue          remaining
c           level        conflicts        target    irredundant
c
c d 77.02 29 127 55 16772 475023 100881 57 556 44% 47 181134 27530 51%
c i 77.05 29 127 55 16772 475023 100881 57 556 44% 47 181134 27530 51%
c - 77.34 23 127 56 16775 478843 22545 57 556 44% 47 181089 27530 51%
c - 79.12 22 127 57 16780 495943 20549 57 556 44% 48 181089 27530 51%
c W 79.22 16 127 57 16780 496069 20671 57 557 44% 48 181089 27530 51%
c ] 80.45 28 123 57 16786 506395 30307 57 587 44% 47 181089 27530 51%
c { 80.45 28 58 57 16786 506395 30307 57 587 40% 18 181089 27530 51%
c - 81.31 21 57 58 17030 513343 21270 57 591 40% 18 181089 27530 51%
```

Demo: KISSAT rephase scheduling

```
c O  0.21 13 3957  2 84 1002 589 13 14 31% 30 300264 51178 95%
c I  0.81 17 455  4 122 3003 6761 13 34 32% 9 294546 49206 91%
c B  1.14 17 1130  5 271 6006 6087 20 55 33% 23 293659 49035 91%
c W  2.17 18 175  7 359 10009 6761 20 80 34% 8 277679 47447 88%
c O  3.21 16 401  9 490 15010 61412 29 109 39% 24 237913 34101 63%
c B  4.02 16 337 11 1161 21011 4770 36 128 39% 23 232789 34002 63%
c W  5.45 18 112 13 1514 28012 7588 36 146 30% 9 220480 32919 61%
c I  7.01 17 228 15 2089 36014 6305 36 156 43% 47 220480 32919 61%
c B  9.37 19 212 16 3485 45026 11926 46 185 44% 47 218330 32526 60%
c W 10.95 18 184 18 3490 55027 66706 46 195 43% 62 215284 32387 60%
c B 14.91 18 180 22 4950 78032 10126 48 237 45% 62 208079 31234 58%
c W 16.65 20 87 24 5008 91033 9816 48 264 38% 16 207270 31222 58%
c F 18.66 26 155 25 5646 105035 16066 48 286 46% 50 207270 31222 58%
c B 21.57 21 148 27 6792 120038 17481 48 316 46% 47 205968 31123 58%
c W 23.88 26 160 29 6801 136039 16018 49 333 46% 54 205311 30808 57%
c O 27.59 19 70 31 8658 153040 16075 49 359 41% 18 204440 30732 57%
c B 30.56 17 139 33 8750 171042 15722 49 366 44% 50 204211 30724 57%
c W 33.40 17 66 35 9141 190043 14369 49 369 41% 17 203773 30515 57%
c I 36.87 21 138 36 10079 210045 25010 49 387 43% 52 202044 30394 56%
c B 40.02 23 135 38 10090 231046 73732 49 412 44% 50 201818 29876 55%
c W 42.86 20 60 40 10693 253048 22396 49 413 41% 17 195708 29703 55%
c B 50.25 18 61 44 11748 300049 73579 50 452 41% 17 193786 29417 55%
c W 54.02 22 136 46 13040 325052 21669 57 460 44% 49 193774 29415 55%
c F 58.25 27 139 47 13050 351055 82892 57 472 44% 52 191395 28781 53%
c B 62.78 21 58 49 14407 378056 28423 57 505 41% 18 191087 28743 53%
c W 67.07 25 135 51 14749 406062 29277 57 510 43% 51 182703 28024 52%
c O 70.69 23 58 53 14840 435064 25762 57 539 40% 17 182389 28012 52%
c B 75.49 23 125 55 16767 465068 24885 57 555 43% 47 181278 27538 51%
c W 79.43 17 127 57 16780 496069 20671 57 557 44% 48 181089 27530 51%
c I 84.86 27 57 58 17839 528074 106383 57 592 40% 18 180813 27512 51%
```

Implementation

Scheduling

Alternation between SAT/UNSAT mode: Chanseok Oh

Stable mode slow changes Luby restarts, smooth bumping, target phases

Focused mode agile GLUCOSE-style restarts, aggressive bumping, phase saving (only)

scheduled in geometrically increasing conflict intervals CADICAL

Rephasing scheduled in arithmetically increasing intervals

1000 conflicts base interval

Rephasing frequency in SAT/UNSAT interval steadily increasing

Implementation

Kissat

Kissat in Finnish   or Keep it simple (and clean) SAT solver

`http://fmv.jku.at/kissat`

to make it easier to find

If you liked CADICAL, you will love KISSAT

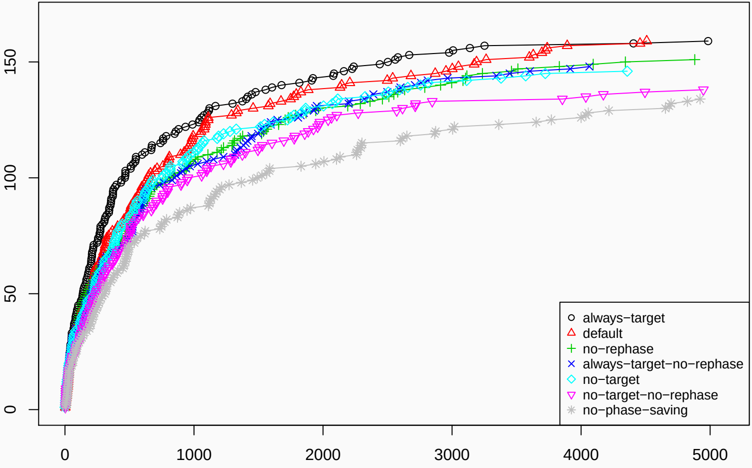
Port to C	removed redundant computation	
Less memory (I)	no binary clauses in the arena	
Less memory (II)	compact watcher data structures	LINGELING
Less memory (III)	support for $2^{28} - 1$ variables	

```
p cnf 268435455 0
```

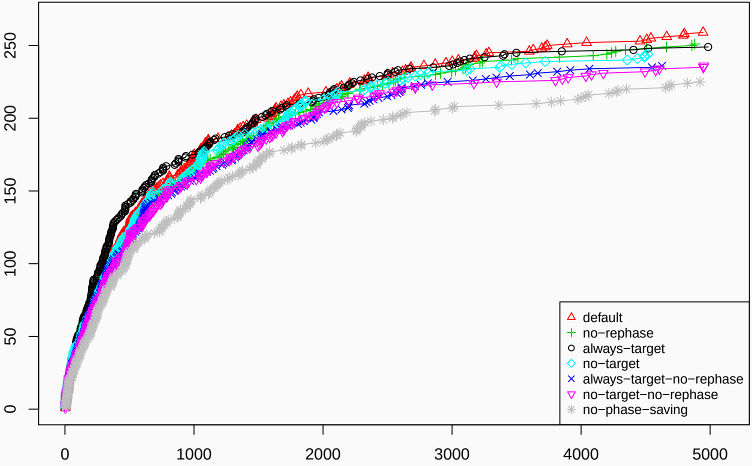
nearly no memory usage

Compact code	faster compile time and no comments	☺

Kissat, SAT Race 2019, satisfiable only



KISSAT, SAT Race 2019, all



Implementation

CaDiCaL

More variables — `INT_MAX` variables — requires a lot of memory though

Simpler rephasing — no autarky calculation when rephasing

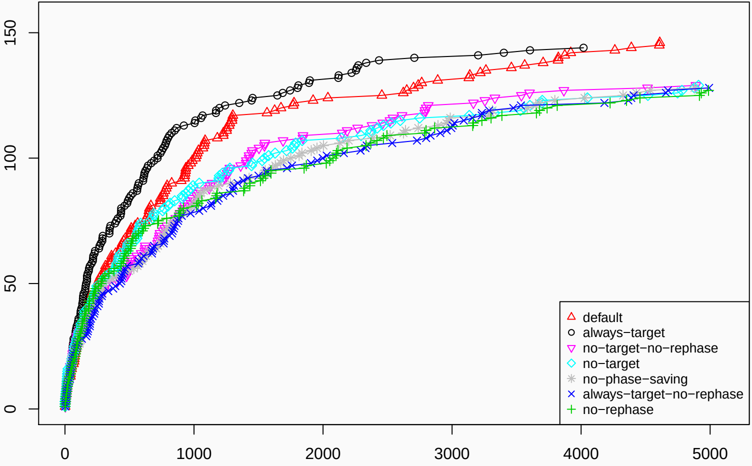
Fewer mode switches — $\mathcal{O}(2^n)$ vs. $\mathcal{O}(n \cdot \log^3 n)$ conflict intervals

Kissat conflict intervals:

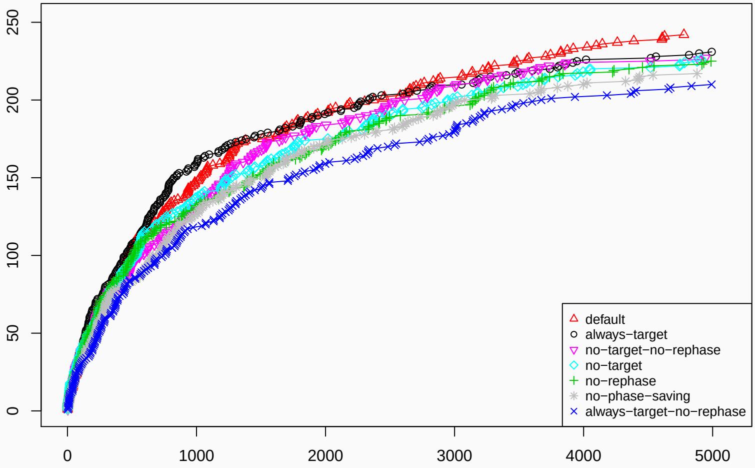
$\underbrace{\mathcal{O}(\log n)}_{\text{restart}} < \underbrace{\mathcal{O}(n/\log n)}_{\text{reduce}} < \underbrace{\mathcal{O}(n)}_{\text{rephase}} < \underbrace{\mathcal{O}(n \cdot \log n)}_{\text{probing}} < \underbrace{\mathcal{O}(n \cdot \log^2 n)}_{\text{elimination}} < \underbrace{\mathcal{O}(n \cdot \log^3 n)}_{\text{SAT/UNSAT mode}}$

release/competition version of Kissat $\mathcal{O}(n^2)$

CaDiCaL, SAT Race 2019, satisfiable only



CaDiCaL, SAT Race 2019, all



Implementation

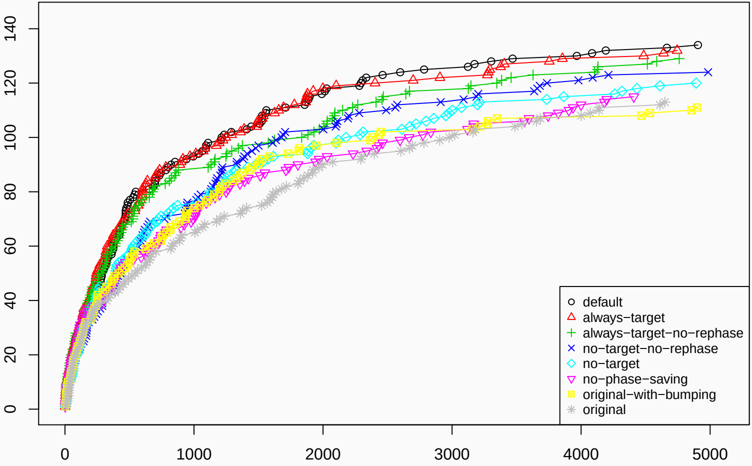
Glucose

Glucose

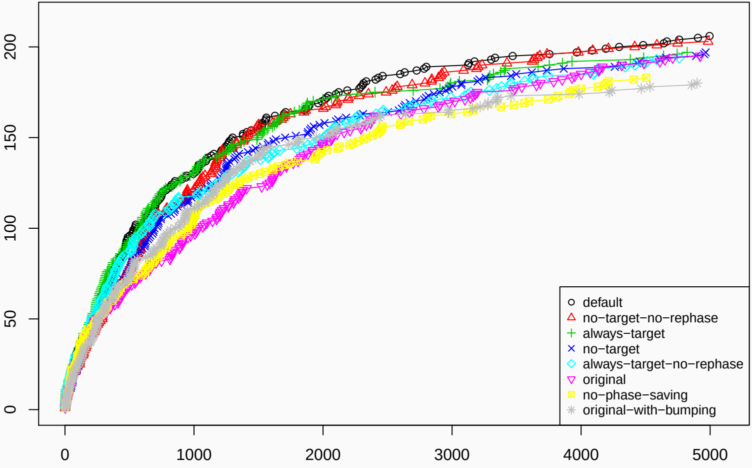
Stable mode	low variable decay	no chronological backtracking
Focus mode	high variable decay	VSIDS = poor man's VMTF

Bumping of reasons turned out to be important   not for normal Glucose

GLUCOSE, SAT Race 2019, satisfiable only



Glucose, SAT Race 2019, all



Implementation

Spass-Satt

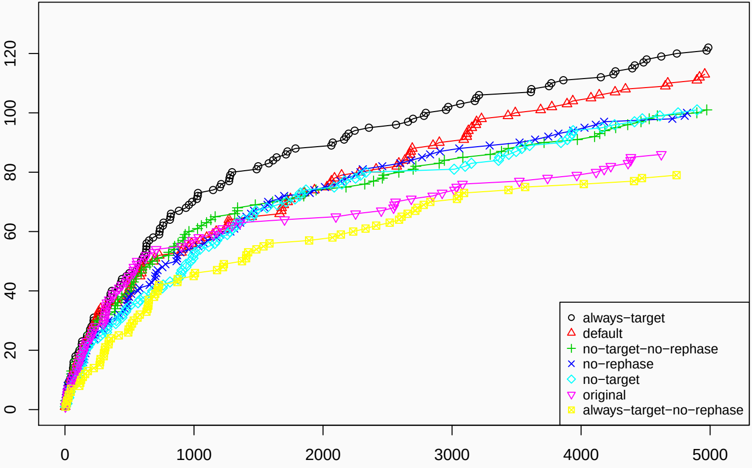
Core of the CDCL($\mathcal{T}$) solver SPASS-SATT

Based on the ideas of GLUCOSE 2

Has inprocessing (subsumption-resolution until fixpoint)

But: no BCE, no BVE

SPASS-SATT, SAT Race 2019, satisfiable only



Conclusion

Rephasing alone helps Kissat on SAT Comp. 2018, not 2019

somewhat *fragile*: bad with the wrong strategy

Target phasing with rephasing helps for satisfiable instances

key idea: Maximize trail length

Autarky and random walk have an unclear effect.

Alternation is a good compromise.

`http://fmv.jku.at/chasing-target-phases`

Appendix

Appendix

Detailed Results

Performance of the SAT solvers KISSAT and CADICAL

	All instances				Satisfiable instances			
	KISSAT		CADICAL		KISSAT		CADICAL	
Configuration	Solved	PAR2	Solved	PAR2	Solved	PAR2	Solved	PAR2
`default`	259	1662994	242	1857009	159	263235	146	323506
`alw.-target`	249	1714647	231	1921490	159	234467	144	294832
`no-targetno-rephase`	236	1859306	227	1979507	138	450986	129	473741
`no-target`	244	1775317	226	2002909	146	366564	129	479070
`no-rephase`	251	1736491	225	2020048	151	341314	127	509767
`no-phasesaving`	225	1986220	217	2084103	134	522721	128	498526
`alw.-targetno-rephase`	236	1851424	210	2165365	148	359317	128	508870

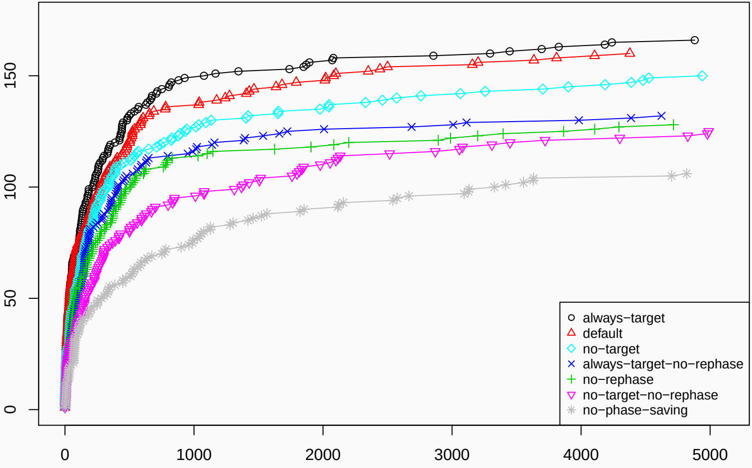
Performance of the SAT solvers Glucose and Spass-Satt

	All instances				Satisfiable instances			
	Glucose		Spass-Satt		Glucose		Spass-Satt	
Configuration	Solved	PAR2	Solved	PAR2	Solved	PAR2	Solved	PAR2
`default`	206	2154525	159	2671578	134	368068	113	532083
`alw.-target`	197	2222227	168	2582440	132	383353	122	436410
`no-targetno-rephase`	203	2192177	148	2741503	124	476546	101	616097
`no-target`	197	2259101	151	2736161	120	516055	101	626393
`alw.-targetno-rephase`	194	2282175	127	2909751	129	424991	79	801110
`original`	195	2312300	137	2829600	112	359317	86	729944

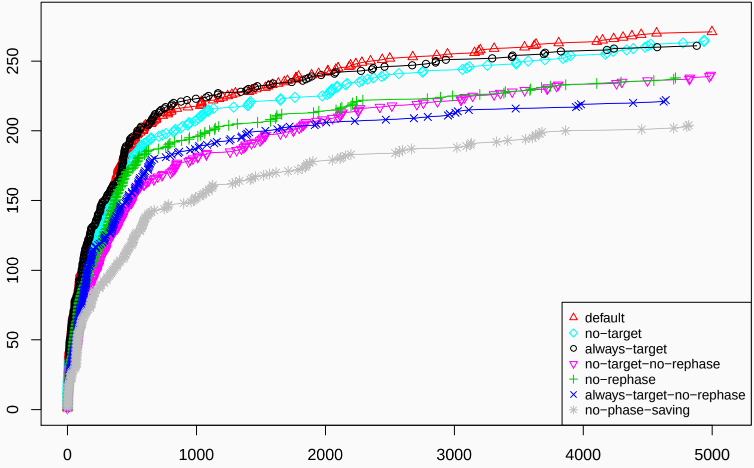
Appendix

More Results

Kissat, SAT Competition 2018, satisfiable only



Kissat, SAT Competition 2018, all



Spass-Satt, SAT Competition 2018