Improving Implementation of SAT Competitions 2017-2019 Winners

Stepan Kochemazov
veinamond@gmail.com

Matrosov Institute for System Dynamics and Control Theory SB RAS
Irkutsk, Russia

SAT2020

The solvers that enter recent SAT competitions are biased towards the winner of the previous competition.

The solvers that enter recent SAT competitions are biased towards the winner of the previous competition.

- Each of the winners of SAT competitions 2017-2019 is based on the winner of the previous competition.
  - SC2016 main track winner: `MapleCOMSPS`.
  - In 2017, 2 solvers out of 29 participants were based on `MapleCOMSPS`.
    - SC2017 main track winner: `Maple_LCM_Dist`.
  - In 2018, about 13 out of 41 participants were based on `Maple_LCM_Dist`.
    - SC2018 main track winner: `MapleLCMDistChronoBT`.
  - In 2019, about 24 out of 55 participants were based on `Maple_LCM_Dist`, 15 of them – on `MapleLCMDistChronoBT`.
    - SR2019 winner: `MapleLCMDistChronoBT-DL-v3`.

The solvers that enter recent SAT competitions are biased towards the winner of the previous competition.

- Each of the winners of SAT competitions 2017-2019 is based on the winner of the previous competition.
  - SC2016 main track winner: `MapleCOMSPS`.
  - In 2017, 2 solvers out of 29 participants were based on `MapleCOMSPS`.
    - SC2017 main track winner: `Maple_LCM_Dist`.
  - In 2018, about 13 out of 41 participants were based on `Maple_LCM_Dist`.
    - SC2018 main track winner: `MapleLCMDistChronoBT`.
  - In 2019, about 24 out of 55 participants were based on `Maple_LCM_Dist`, 15 of them – on `MapleLCMDistChronoBT`.
    - SR2019 winner: `MapleLCMDistChronoBT-DL-v3`.
- On the one hand, when implementing a new heuristic it is easier to work with the existing implementation as is.
- On the other hand, improving common implementation aspects of winners of several years may result in improving the effectiveness of a major part of potential SAT competition participants.

The goal of the study was to make deterministic implementations of SAT Competitions 2017-2019 winners with performance comparable or better to the original solvers.

The goal of the study was to make deterministic implementations of SAT Competitions 2017-2019 winners with performance comparable or better to the original solvers.

- `MapleCOMSPS`, `Maple_LCM_Dist` and `MapleLCMDistChronoBT` switch between branching heuristics after 2500 seconds.
  - On different systems / launches this switch happens at different stages in terms of number of conflicts / number of propagations.

The goal of the study was to make deterministic implementations of SAT Competitions 2017-2019 winners with performance comparable or better to the original solvers.

- `MapleCOMSPS`, `Maple_LCM_Dist` and `MapleLCMDistChronoBT` switch between branching heuristics after 2500 seconds.
  - On different systems / launches this switch happens at different stages in terms of number of conflicts / number of propagations.
    - The solving of all problems with runtime $>$ 2500 seconds is very hard to explicitly reproduce.
    - Hinders the development of SAT solver modifications since the same solver in different launches over the same benchmark set may solve +/- 3 instances.

The goal of the study was to make deterministic implementations of SAT Competitions 2017-2019 winners with performance comparable or better to the original solvers.

- `MapleCOMSPS`, `Maple_LCM_Dist` and `MapleLCMDistChronoBT` switch between branching heuristics after 2500 seconds.
  - On different systems / launches this switch happens at different stages in terms of number of conflicts / number of propagations.
    - The solving of all problems with runtime $>$ 2500 seconds is very hard to explicitly reproduce.
    - Hinders the development of SAT solver modifications since the same solver in different launches over the same benchmark set may solve +/- 3 instances.
- The authors of `MapleCOMSPS` realized their oversight and submitted to later competitions only deterministic variants of their solver.

All experiments were performed using a single node of the "Academician V.M. Matrosov" cluster.[1]

- Two 18-core Intel Xeon E5-2695 (Broadwell) CPUs with 128 GB DDR4 RAM.
- The solvers were launched in 36 simultaneous threads with time limit of 5000 seconds.
- All logs of experiments are available online.[2]

To test the proposed implementation changes the modifications of `MapleLCMDistChronoBT` were launched on the instances from SAT Race 2019.

[1] http://hpc.icc.ru/en/

[2] https://github.com/veinamond/SAT2020

All experiments were performed using a single node of the "Academician V.M. Matrosov" cluster.[1]

- Two 18-core Intel Xeon E5-2695 (Broadwell) CPUs with 128 GB DDR4 RAM.
- The solvers were launched in 36 simultaneous threads with time limit of 5000 seconds.
- All logs of experiments are available online.[2]

To test the proposed implementation changes the modifications of `MapleLCMDistChronoBT` were launched on the instances from SAT Race 2019.

Main contributions

- Implementing the deterministic strategy for switching between branching heuristics.

[1] http://hpc.icc.ru/en/

[2] https://github.com/veinamond/SAT2020

All experiments were performed using a single node of the "Academician V.M. Matrosov" cluster.[1]

- Two 18-core Intel Xeon E5-2695 (Broadwell) CPUs with 128 GB DDR4 RAM.
- The solvers were launched in 36 simultaneous threads with time limit of 5000 seconds.
- All logs of experiments are available online.[2]

To test the proposed implementation changes the modifications of `MapleLCMDistChronoBT` were launched on the instances from SAT Race 2019.

Main contributions

- Implementing the deterministic strategy for switching between branching heuristics.
- Improving the handling of *Tier2* and *Core* learnts.

[1] http://hpc.icc.ru/en/

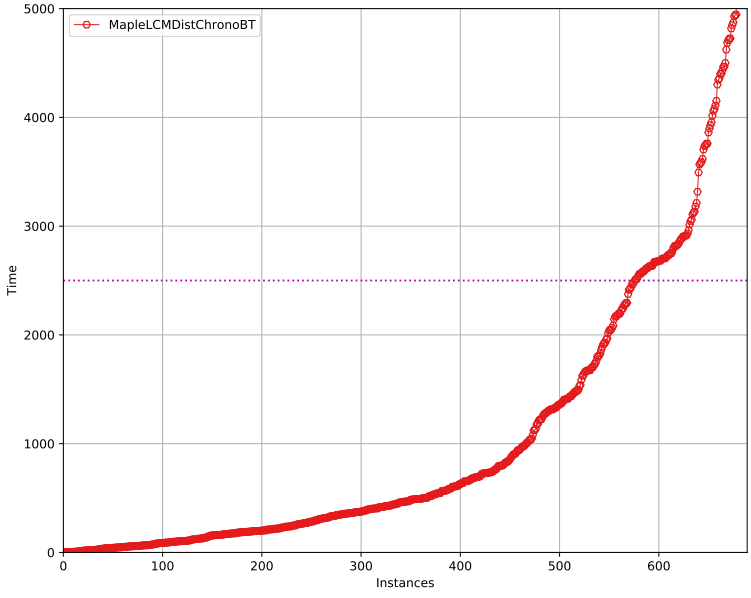
[2] https://github.com/veinamond/SAT2020

- In `COMiniSatPS` it was proposed to use distinct two sets of VSIDS scores: one with `Minisat`-like Luby restarts and another with `glucose` restarts. `COMiniSatPS` frequently switches between these two phases.

- In `COMiniSatPS` it was proposed to use distinct two sets of VSIDS scores: one with `Minisat`-like Luby restarts and another with `glucose` restarts. `COMiniSatPS` frequently switches between these two phases.
- In `MapleCOMSPS` the first set of VSIDS scores was replaced by LRB scores.

- In COMiniSatPS it was proposed to use distinct two sets of VSIDS scores: one with Minisat-like Luby restarts and another with glucose restarts. COMiniSatPS frequently switches between these two phases.
- In MapleCOMSPS the first set of VSIDS scores was replaced by LRB scores.
  - In MapleCOMSPS the switch happens once at 2500 seconds: the solver starts with LRB+Luby restarts and then switches to VSIDS+glucose restarts.
  - The MapleCOMSPS implementation of the switching between branching heuristics was inherited by Maple_LCM_Dist and MapleLCMDistChronoBT, the winners of SAT Competitions 2017 and 2018.
  - The MapleLCMDistChronoBT-DL-v3 solver that won at SAT Race 2019 uses a deterministic strategy for switching between LRB+Luby restarts and VSIDS+glucose restarts.

Cactus plot for MapleLCMDistChronoBT over benchhmarks from SC2017-SR2019

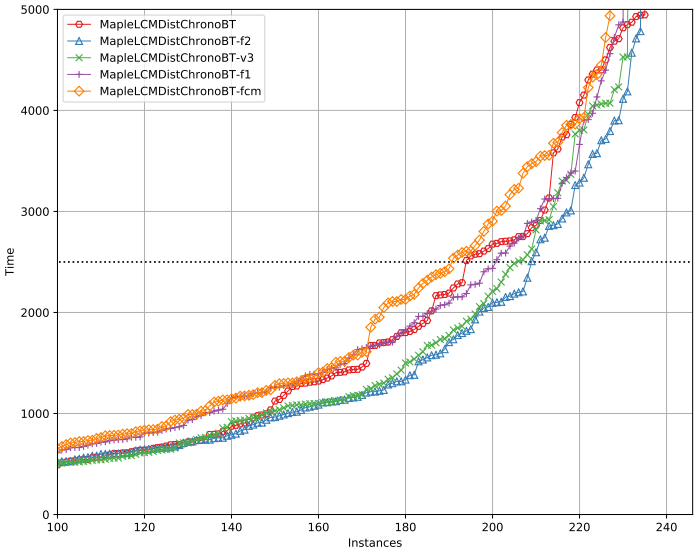


Abstracted part of the solve_ procedure

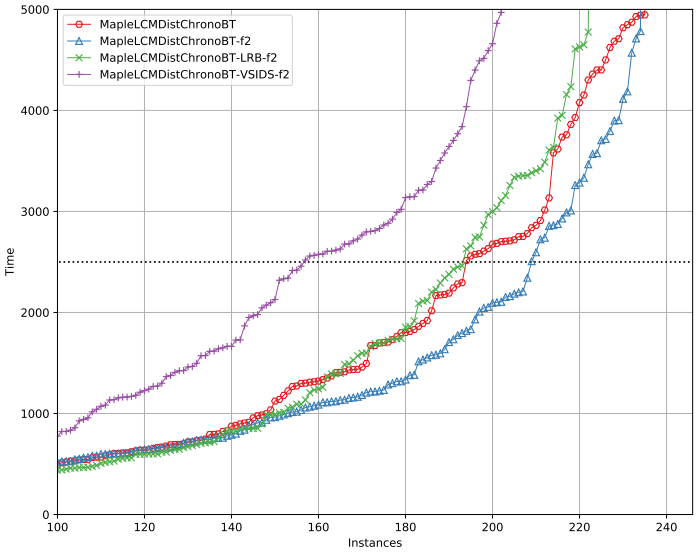
```
VSIDS = false;
int phase_allotment = initial_value;
int c_r = 0;
for (;;){
  int weighted =  VSIDS ? phase_allotment * VSIDS_mult : phase_allotment;
  while (status == l_Undef && weighted > 0)
    if (VSIDS)
      status = search(weighted);
    else{
      int nof_conflicts = luby(r_inc, c_r) * r_first;
      c_r++;
      weighted -= nof_conflicts;
      status = search(nof_conflicts);
    }
  if (status != l_Undef) break;
  VSIDS = !VSIDS;
  if (!VSIDS) phase_allotment *= mult1;
  else phase_allotment *= mult2;
}
```

	Initial value	VSIDS_mult	mult1 / mult2	Source
fcm	100 conflicts	2	1.1/1	COMiniSatPS
f1	100 conflicts	1	1.1/1	MapleCOMSPS_LRB
f2	10K conflicts	1	2/1	MapleCOMSPS_LRB_VSIDS_2
v3	30M propagations	1	1.1/1.1	MapleLCMDistChronoBT-DL-v3

Solver	SCR:S:U	PAR-2
Chrono	236:138:98	4799
f1	231:135:96	4882
f2	235:140:95	**4716**
v3	232:137:95	4787
fcm	228:133:95	4985



Solver	SCR:S:U	PAR-2
Chrono	236:138:98	4799
f2	235:140:95	4716
LRB-f2	223:134:89	4996
VSIDS-f2	203:115:88	5589



- In COMiniSatPS-based solvers all learnt clauses are split into three tiers:

- In COMiniSatPS-based solvers all learnt clauses are split into three tiers:
  - *Core* tier contains clauses with lbd$\leq$ core_lbd_cut
    - core_lbd_cut$=3$ intially but is increased to 5 if after the first 100 000 conflicts the size of *Core* is < 100.
    - *Core* learnts are never removed.

- In COMiniSatPS-based solvers all learnt clauses are split into three tiers:
  - *Core* tier contains clauses with lbd$\leq$ core_lbd_cut
    - core_lbd_cut$=3$ intially but is increased to 5 if after the first 100 000 conflicts the size of *Core* is < 100.
    - *Core* learnts are never removed.
  - *Tier2* tier contains clauses with core_lbd_cut$<$lbd≤ 6.
    - reduceDB_Tier2() is invoked every 10000 conflicts and demotes all *Tier2* learnts that have not participated in the most recent 30 000 conflicts to *Local*.

- In COMiniSatPS-based solvers all learnt clauses are split into three tiers:
  - *Core* tier contains clauses with lbd$\leq$ core_lbd_cut
    - core_lbd_cut$=3$ intially but is increased to 5 if after the first 100 000 conflicts the size of *Core* is < 100.
    - *Core* learnts are never removed.
  - *Tier2* tier contains clauses with core_lbd_cut$<$lbd≤ 6.
    - reduceDB_Tier2() is invoked every 10000 conflicts and demotes all *Tier2* learnts that have not participated in the most recent 30 000 conflicts to *Local*.
  - *Local* tier contains all clauses with lbd > 6 and the ones demoted from *Tier2*.
    - reduceDB() is invoked every 15000 conflicts and removes roughly half of *Local* learnts with the lowest activity.

- In COMiniSatPS-based solvers all learnt clauses are split into three tiers:
  - *Core* tier contains clauses with `lbd`$\leq$ `core_lbd_cut`
    - `core_lbd_cut`$= 3$ intially but is increased to 5 if after the first 100 000 conflicts the size of *Core* is < 100.
    - *Core* learnts are never removed.
  - *Tier2* tier contains clauses with `core_lbd_cut`$<$`lbd`≤ 6.
    - `reduceDB_Tier2()` is invoked every 10000 conflicts and demotes all *Tier2* learnts that have not participated in the most recent 30 000 conflicts to *Local*.
  - *Local* tier contains all clauses with `lbd` > 6 and the ones demoted from *Tier2*.
    - `reduceDB()` is invoked every 15000 conflicts and removes roughly half of *Local* learnts with the lowest activity.
- The Learnt Clause Minimization (LCM) introduced in `Maple_LCM_Dist` is periodically applied to *Core* and *Tier2* learnts.
  - Each learnt clause can undergo LCM only once.
  - *Tier2* clauses for which `lbd` was reduced by LCM can go to *Core*.

Maintaining the size of *Tier2*

- In `COMiniSatPS` the goal of *Tier2* learnts was to store clauses with average `lbd` to help with tackling UNSAT instances.

Maintaining the size of *Tier2*

- In `COMiniSatPS` the goal of *Tier2* learnts was to store clauses with average `lbd` to help with tackling UNSAT instances.
- The original concept did not take into account LCM.
  - Over the benchmarks from SR2019, the average size of *Tier2* varies from 200 learnt clauses to 12 000 since for some instances the average `lbd` of clauses tends to be larger.

Maintaining the size of *Tier2*

- In `COMiniSatPS` the goal of *Tier2* learnts was to store clauses with average `lbd` to help with tackling UNSAT instances.
- The original concept did not take into account LCM.
  - Over the benchmarks from SR2019, the average size of *Tier2* varies from 200 learnt clauses to 12 000 since for some instances the average `lbd` of clauses tends to be larger.
  - The potential of LCM might not be realized in full for instances on which *Tier2* tends to be small
- Idea: maintain the size of *Tier2* at some reasonable point.
  - Reorganize `ReduceDB_Tier2()` to be invoked once *Tier2* size exceeds 7 000.

Maintaining the size of *Tier2*

- In `COMiniSatPS` the goal of *Tier2* learnts was to store clauses with average `lbd` to help with tackling UNSAT instances.
- The original concept did not take into account LCM.
  - Over the benchmarks from SR2019, the average size of *Tier2* varies from 200 learnt clauses to 12 000 since for some instances the average `lbd` of clauses tends to be larger.
  - The potential of LCM might not be realized in full for instances on which *Tier2* tends to be small
- Idea: maintain the size of *Tier2* at some reasonable point.
  - Reorganize `ReduceDB_Tier2()` to be invoked once *Tier2* size exceeds 7 000.
  - In `ReduceDB_Tier2()` preserve half of *Tier2* clauses that participated in the most recent conflicts.

Reducing *Core* learnts

- For about 80% of instances from SR2019, `MapleLCMDistChronoBT-f2` accumulates more than 50 000 *Core* learnts.

Reducing *Core* learnts

- For about 80% of instances from SR2019, `MapleLCMDistChronoBT-f2` accumulates more than 50 000 *Core* learnts.
- The excessive size of *Core* may slow down BCP.

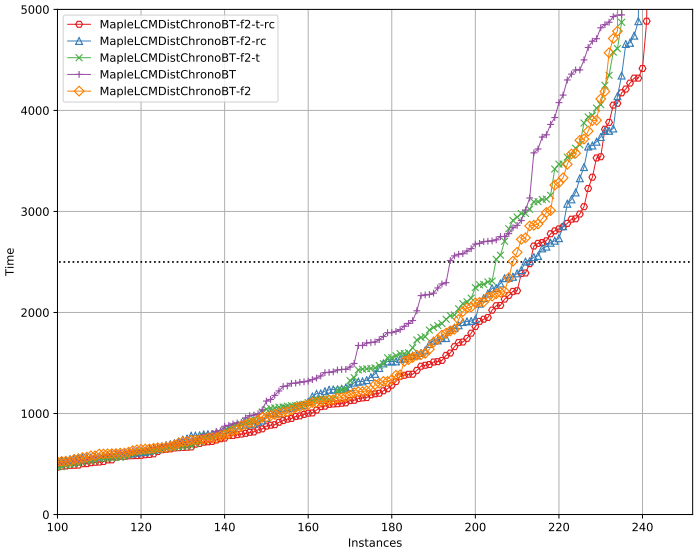
Reducing *Core* learnts

- For about 80% of instances from SR2019, `MapleLCMDistChronoBT-f2` accumulates more than 50 000 *Core* learnts.
- The excessive size of *Core* may slow down BCP.
- Idea: gently reduce the *Core* database via `reduceDB_Core`
  1. Sort learnts according to their `lbd` and size for equal `lbd`.
  2. Move all clauses from the second half (with larger `lbd` and size), that did not participate in any of the most recent 100 000 conflicts into *Tier2*.

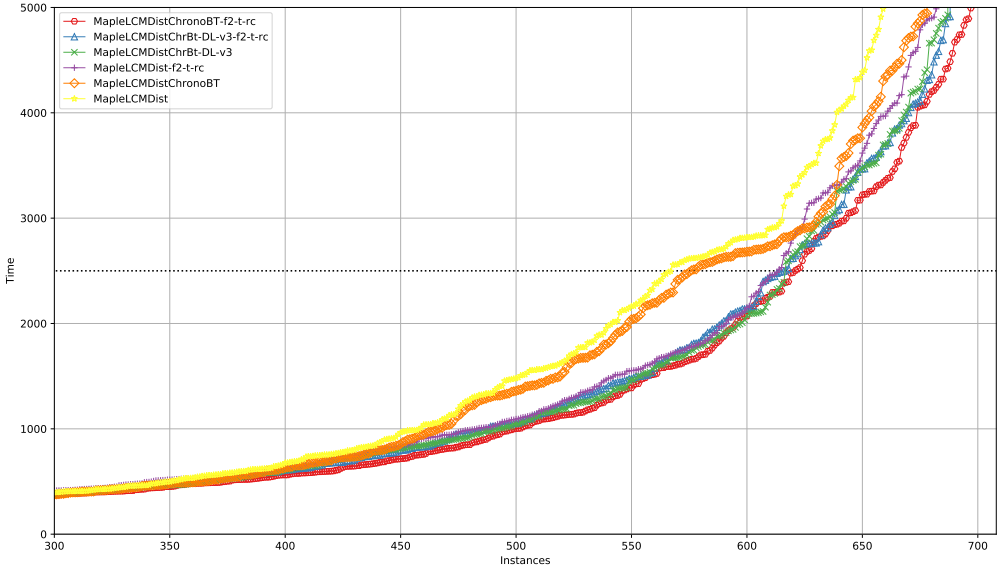
Reducing *Core* learnts

- For about 80% of instances from SR2019, `MapleLCMDistChronoBT-f2` accumulates more than 50 000 *Core* learnts.
- The excessive size of *Core* may slow down BCP.
- Idea: gently reduce the *Core* database via `reduceDB_Core`
  1. Sort learnts according to their `lbd` and size for equal `lbd`.
  2. Move all clauses from the second half (with larger `lbd` and size), that did not participate in any of the most recent 100 000 conflicts into *Tier2*.
- Apply `reduceDB_Core` once the *Core* size exceeds some limit
  - Initially the limit is set to 50 000
  - It is multiplied by 1.1 after each `reduceDB_Core`

Solver	SCR:S:U	PAR-2
Chrono	236:138:98	4799
f2	235:140:95	4716
f2-t	236:139:97	4720
f2-rc	240:143:97	4631
f2-t-rc	**242:143:99**	**4556**



- To evaluate how the winners of SC2017 and SR2019 behave when augmented with the proposed heuristics, the corresponding changes were implemented into `Maple_LCM_Dist` and `MapleLCMDistChronoBT-DL-v3`.
- Then all 6 solvers were launched on the joint set of benchmarks from SC2017, SC2018 and SR2019.


MapleLCMDistChronoBT-f2-t-rc
MapleLCMDistChrBt-DL-v3-f2-t-rc
MapleLCMDistChrBt-DL-v3
MapleLCMDist-f2-t-rc
MapleLCMDistChronoBT
MapleLCMDist
Time
Instances
0
1000
2000
3000
4000
5000
300
350
400
450
500
550
600
650
700

	SC2017		SC2018		SR2019		Total	
	SCR:S:U	PAR-2	SCR:S:U	PAR-2	SCR:S:U	PAR-2	SCR:S:U	PAR-2
MLCMD	206:100:106	4706	229:129:100	4718	225:128:97	4999	660:357:303)	4812
MLCMD*	204: 99:105	4693	237:135:102	4541	242:144:98	4597	683:378:305	4607
	2 : 1: 1	13	8: 6: 2	177	17: 16:11	402	23: 21: 2	205
MLCMDCh	208: 97:111	4659	235:133:102	4573	236:138:98	4799	679:368:311	4678
MLCMDCh*	217:103:114	4373	239:136:103	4490	242:143:99	4556	698:382:316	4477
	9 : 6: 3	286	4: 3: 1	83	6: 5: 1	243	19: 14: 5	201
MLCMDCh-DL	208: 99:109	4586	243:142:101	4402	237:141:96	4675	688:382:306	4553
MLCMDCh-DL*	204: 92:112	4707	242:139:103	4423	243:146:97	4547	689:377:312	4553
	4 : 7: 3	121	1: 3: 2	21	6: 5: 1	128	1: 5: 6	-

- Deterministic switching between branching heuristics appears to significantly improve PAR-2 scores by reducing the average runtime of a solver.
- Careful pruning of the *Core* database together with alternated maintenance of *Tier2* helps the solver tackle harder instances.
- It is possible to construct deterministic variants of the winners of SAT competitions 2017-2019 that achieve comparable or better performance than the non-deterministic solvers over wide range of instances.

Thank you for your attention!