On the Effect of Learned Clauses on Stochastic Local Search

Jan-Hendrik Lorenz   Florian Wörz

July 8th, 2020

Motivation

SLS $\widehat{=}$ Search
CDCL $\widehat{=}$ Intelligent Search

Rough idea: Use preprocessing in SLS to find a logically equivalent formula.

Suspicion: Runtime of SLS on these instances can vary dramatically.

AIM: Find (efficently computable) log. equiv. formula which is beneficial to the runtime.

- Operates on complete assignments,
- starts with a complete initial assignment α,
- tries to find a solution by repeatedly flipping variables.

Input: Formula F, $maxFlips$, function f
α := complete assignment for F
for *i = 1* **to** *maxFlips* **do**
  if α *satisfies* F **then return** “satisfiable”
  Choose a falsified clause $C = (u_1 \vee u_2 \vee \cdots \vee u_\ell)$
  Choose $j \in \{1, \ldots, \ell\}$ with probability according to f
  Flip the chosen variable u_j and update α

- probSAT-based solvers performed excellently on random instances:
  - probSAT won the random track of the SAT competition 2013,
  - dimetheus [BM16] in 2014 and 2016,
  - YalSAT [Bie17] won in 2017.
- Only recently, in 2018, other types of solvers significantly exceeded probSAT based algorithms. $\rightarrow$ Reason for choosing probSAT in this study.

Backbone, General and Deceptive Model

First idea:

- Use a formula F as a base.
- Add a set of clauses $S = \{C_1, \ldots, C_t\}$ to F to obtain a new formula $G := F \cup S$.

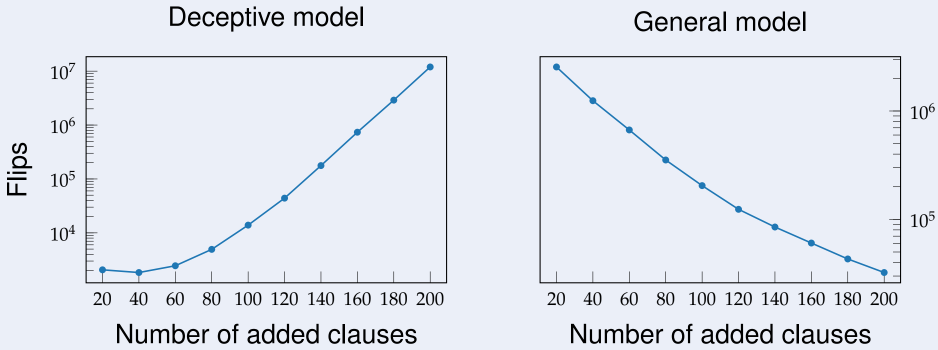
Definition ([Kil+05])

The **backbone** $\mathcal{B}(F)$ are the literals appearing in all satisfying assignments of F.

Deceptive model: $(x \vee \overline{y} \vee \overline{z})$ where $x, y, z \in \mathcal{B}(F)$

General model: $(x \vee y \vee z)$ where $x \in \mathcal{B}(F)$ and $y, z \in \mathrm{Var}(F)$

Effect of the Models



Definition

We call clauses that have a high number of correct literals w. r. t. a fixed solution **high-quality clauses**.

General and Deceptive Model Are Not Realistic

Evident: It is crucial which clauses are added.

Problem: Neither the deceptive nor the general model can be applied to real instances (we would need to know the solution space / calculating Backbones is not efficient).

Idea: Compare models based on resolution and CDCL.

Definition

- Let F be a formula and let $B, C \in F$ be clauses such that there is a resolvent R.
  We call R **level 1 resolvent**.
- Let D or E (or both) be level 1 resolvents and S be their resolvent.
  Then we call S a **level 2 resolvent**.

F_1, F_2, F_C

Let F be a 3-CNF formula with m clauses. We obtain new and log. equiv. formulas:

- F_1 Randomly select $\leq m/10$ level 1 resolvents of width ≤ 4 and add them to F.
- F_2 Randomly select $\leq m/10$ level 2 resolvents of width ≤ 4 and add them to F.
- F_C Randomly select $\leq m/10$ clauses of width ≤ 4 from Glucose (with a time limit of 300 seconds) and add them to F.

Tests on Uniform Random Instances: Setting and Results

- Observe behavior of probSAT over 1000 runs per instance on instance types F_1, F_2, F_C.
- Testbed of uniformly generated 3-CNF instances with
  - 5000 – 11 600 variables and
  - ratio of 4.267.

Results:

- Type F_1 most challenging for probSAT (even harder than original formula).
- Type F_2 better (t-test: $p < 0.01$).
- Type F_C most efficient (t-test: $p < 0.05$) $\rightarrow$ will investigate this further.

Hidden Solution Instances: Definition

Randomly generated instances with hidden solution [BC18]:

- Given a solution α.
- Randomly generate a clause with 3 literals.
- Depending on the number i of satisfied literals under α add the clause with probability p_i.
- Repeat until enough clauses are added.

Quality of Clauses

SAT competition 2018 incorporated 3 types of models with hidden solutions (only differing in the parameters).

Measure quality w. r. t. to the hidden solution:

- On all 3 models, level 2 clauses have a higher quality than level 1 claues.
- On 2 of 3 domains, CDCL clauses have a higher quality than level 2 clauses.

Can this method help to improve probSAT?

- The hardness of an instance is impacted by the added clauses.
- CDCL seems to produce high-quality clauses.
- Going forward, we only use clauses generated by Glucose.

- Which clauses should be added? (We focus on the width)
- How many clauses should be added? (In % of the original number of clauses)

Training Data

- All satisfiable, random instances from the SAT-Competition 2014 to 2017.
- In total: 377 instances.
- 120 instances with a **hidden** solution.
- 149 uniform 3, 5, and 7-SAT instances of **medium size**.
- 108 uniform 3, 5, and 7-SAT instances of **huge size**.

Setup

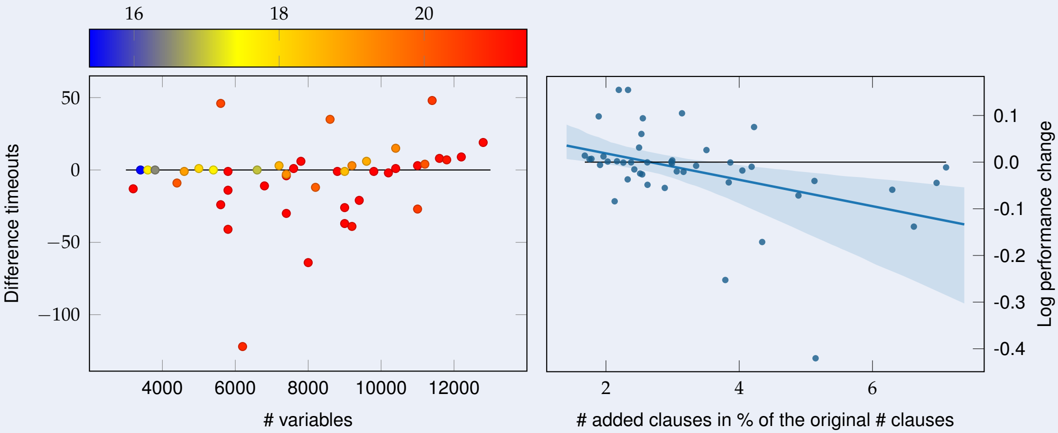
- The experiments were performed on a heterogeneous cluster.
- Thus, seconds are inappropriate to measure the runtime.
- Instead, flips were used.
- Timeouts: 3-SAT: 10^9 flips, 5-SAT: $5 \cdot 10^8$ flips, 7-SAT: $2.5 \cdot 10^8$ flips.

- 1000 runs per instance.
- Performance measure: number of timeouts.

Optimal combinations for uniform, medium size instances

	Width	Number (in %)
3-SAT	≤ 4	unlimited
5-SAT	≤ 8	5%
7-SAT	≤ 9	1%

Uniform Instances: 3-SAT



Uniform Instances: 5 and 7-SAT

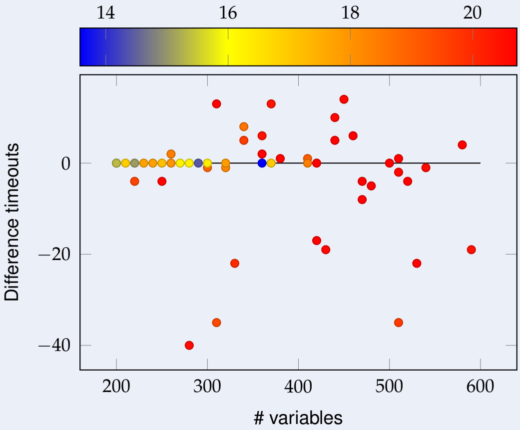


Figure: 5-SAT instances

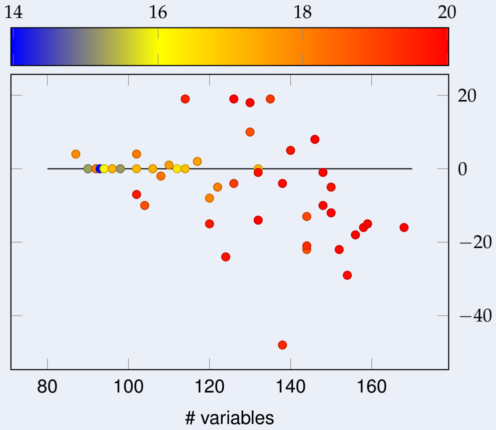
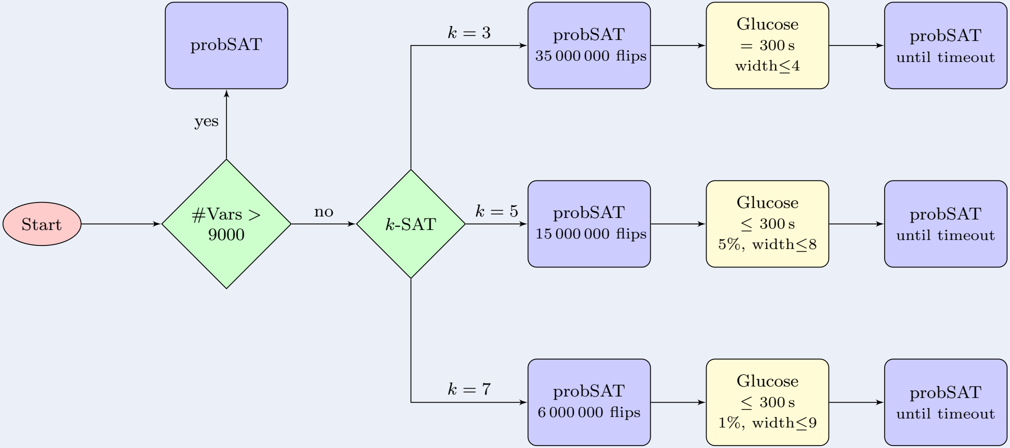


Figure: 7-SAT instances

Hidden Solution and Huge Instances

- Hidden Solution: Similar results, adding new clauses is generally beneficial.
- Huge instances: Few clauses are generated; yielding no significant improvement.


Start
#Vars > 9000
yes
probSAT
no
k-SAT
k = 3
probSAT
35 000 000 flips
Glucose
= 300 s
width≤4
probSAT
until timeout
k = 5
probSAT
15 000 000 flips
Glucose
≤ 300 s
5%, width≤8
probSAT
until timeout
k = 7
probSAT
6 000 000 flips
Glucose
≤ 300 s
1%, width≤9
probSAT
until timeout

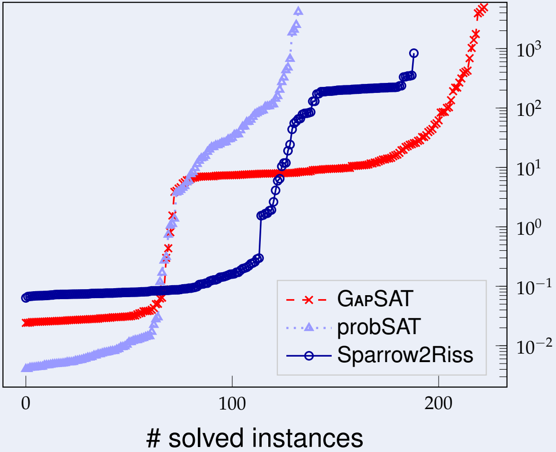
Evaluation

- Random instances of the SAT competition 2018
- Used solvers: probSAT, Sparrow2Riss [BM18], GapSAT
- Timeout: 5000 seconds
- Performance measure: par2

$$\text{par2}(x) = \begin{cases} x, & x < 5000 \\ 10000, & \text{else} \end{cases}$$

Results

	# solved	score
probSAT	133	1 234 986.01
Sparrow2Riss	189	672 335.89
GapSAT	**223**	**347 156.40**



Domain Results

	hidden	medium	huge
probSAT	872 938.74	137 396.83	224 650.43
Sparrow2Riss	8 589.12	171 492.91	492 253.86
GapSAT	**851.36**	**127 982.19**	**218 322.85**

Summary and Outlook

- The presented technique significantly improves probSAT.
- Parameter tuning of probSAT could further improve the results.
- A clause selection heuristic would be useful.
- The supplementary material is available online[1].

[1] `https://zenodo.org/record/3776052`