Using Algorithm Configuration Tools to Generate Hard SAT Benchmarks

Tomáš Balyo[1] Lukáš Chrpa[2]

[1]Karlsruhe Institute of Technology (KIT)

[2]Czech Technical University in Prague & Charles University in Prague

Why do we need hard SAT instances ?

- SAT Competition benchmarks (challenging for SAT solvers)
- Understanding what makes SAT instances hard
- In Cryptography (e.g. a password can be encrypted as a solution of a SAT instance)

State of the Art of SAT benchmark generation

- Generate uniform random SAT formulas from the phase transition region and filter out unsatisfiable instances
- Generate random SAT formula satisfying a given assignment ϕ (the 1-hidden algorithm)
- the Clause Distribution Control (CDC) algorithm
  - a given assignment ϕ
  - probabilities p_i for "accepting" a clause with exactly i satisfied literals
  - a clause/variable ratio r

The CDC algorithm

A configuration of Barthel et al. [Barthel et al. 2002]

$$
\begin{aligned}
r &> 4.25 \\
0.077 < p_1 &< 0.25, \\
p_2 &= (1-4p_1)/6 \\
p_3 &= (1+2p_1)/6
\end{aligned}
$$

Q-hidden configuration [Jia et al., 2005]

$$p_i = q^i$$

```
$\textit{cdc-generate}$ ($vars$, $\phi$, $p_1$, $p_2$, $p_3$, $r$)
    $F := \emptyset$
    while $|F| < r * vars$ do
        $C$ = generateRandom3Clause($vars$)
        $i$ = numberOfSatisfiedLiterals($C$,$\phi$)
        if $i > 0$ then
            with probability $p_i$ do $F = F \cup \{C\}$
    return $F$
```

Algorithm Configuration

- Automated tuning of free parameters to enhance solvers' performance
- SMAC [Hutter et al., 2011]
- Used in SAT (e.g. Spear)
- Used in Planning (e.g., ParLPG, domain configuration)
- ...

Our approach

- We tune the parameters of the CDC algorithm to generate SAT formulas such that we minimize coverage of a given solver
- We increase the number of variables until a given solvers fails to solve at least a half of the formulas

```
evaluate-configuration ($p_1, p_2, p_3, r$)
  $score := 0$
  for $i := 20$ to $600$ step $5$ do
    $solved := 0$
    repeat 8 times:
      $vars :=$ generateVars($i$)
      $\phi :=$ generateAssigment($vars$)
      $F :=$ cdc-generate($vars, \phi, p_1, p_2, p_3, r$)
      if the solver solves $F$ in 1 minute
        then $solved := solved + 1$
    $score := score + solved$
    if $solved < 4$ then break
  return $score$
```

Experimental Settings

- Considered SAT solvers
    - Local Search solvers: ProbSAT, Dimetheus
    - CDCL solvers: Lingeling, Glucose
- Considered configurations
    - ProbSAT, Dimetheus, Lingeling, Glucose
    - Combination, Combination Barthel, Combination Q-hidden

Results

	Number of Solved Instances (out of 410)			
Benchmark Category	ProbSAT	Dimetheus	Lingeling	Glucose
ProbSAT config.	2	88	409	410
Dimetheus config.	52	46	345	391
Lingeling config.	245	245	289	293
Glucose config.	410	410	310	293
Combination	24	31	392	410
Combination Barthel	218	222	288	301
Combination Q-Hidden	51	47	381	410
Uniform 3SAT	410	410	294	284
Barthel et. al	409	409	333	307
Q-Hidden	142	144	365	406

Results II

	Size of Largest Solved Formula (Max. 600)			
Benchmark Category	ProbSAT	Dimetheus	Lingeling	Glucose
ProbSAT config.	80	540	600	600
Dimetheus config.	170	170	600	600
Lingeling config.	600	600	420	400
Glucose config.	600	600	460	420
Combination	80	100	600	600
Combination Barthel	600	600	460	460
Combination Q-Hidden	230	230	600	600
Uniform 3SAT	600	600	480	440
Barthel et. al	600	600	580	500
Q-Hidden	600	600	600	600

Results III

	Size of Hardest Solved Group (Max. 600)			
Benchmark Category	ProbSAT	Dimetheus	Lingeling	Glucose
ProbSAT config.	0	500	600	600
Dimetheus config.	90	90	480	600
Lingeling config.	320	320	340	360
Glucose config.	600	600	400	360
Combination	70	70	600	600
Combination Barthel	560	560	360	380
Combination Q-Hidden	90	90	600	600
Uniform 3SAT	600	600	360	340
Barthel et. al	600	600	460	400
Q-Hidden	270	270	600	600

Each group contains 10 instances of the same size and category. A group is considered to be solved if at least 5 instances are solved.

Take-home Message

- The Uniform and Barthel et. al configurations generate relatively hard instances for the CDCL solvers, however, they seem to be easy for the local search solvers.
- The Q-Hidden approach gives reasonably hard instances for the local search solvers.
- The ProbSAT, Dimetheus, Combination and Combination Q-hidden configurations yield hard instances for the local search solvers. Only a few instances with more than 100 variables are solved.
- The Lingeling and Combination Barthel configurations yield instances that are similarly hard for all the solvers.

Configuring a solver for configured “hard” formulas

- Initially, we generated “hard” SAT formulas for the (configurable) Spear solver [Hutter et al., 2007]
- Then, Spear was tuned on the generated “hard” instances

- The performance improvement (tuned vs default Spear) was around 3.5%
- From these preliminary results we conjecture that Spear cannot be satisfyingly tuned

Thank you !