Practical SAT Solving (ST 2024) | Assignment 5

Markus Iser, Dominik Schreiber, Tomáš Balyo
Algorithm Engineering (KIT)

2024-07-09 – 2024-07-23

1 Application-specific Analysis (8 Points)

Pick **one** of the 20 largest benchmark families of SAT Competition 2022 (see lecture 11 slide 4). Research the following points:

- Where does the family originate from and which people authored it?
- What is the purpose of the benchmark family? (concrete application, exposing pathological solver behavior, highlighting certain techniques, ...)
- Are the instances advertised to have any special properties? (e.g., clause length distribution, proof complexity, only SAT / only UNSAT, structural particularities, ...)
- How did solvers in the 2022 anniversary track perform on the family? Are there discrepancies between the globally best solver(s) and the best solver(s) for that family? If so, can you find an explanation?
- Are the instances good to parallelize, i.e., what is the distribution over the speedups which 2022 parallel solvers can achieve on the family?

Use Markus' GBD tool[1], performance data of the 2022 anniversary track,[2] and the proceedings of past SAT Competitions, where you should find one or several abstracts describing the instances. *You should not need to run a SAT solver nor download/open a SAT instance for this task.*

2 Planning via MaxSAT (4 Points)

Consider a classical (STRIPS-style) automated planning instance where each action a is associated with a cost $c(a) \in \mathbb{N}^+$. We want to find a plan $P = \langle a_1, \ldots, a_n \rangle$ of *minimum cost*, i.e., minimizing $C^* := \sum_{i=1}^{n} c(a_i)$. We also know an upper bound U on the optimal plan cost, i.e., $C^* \leq U$. Design a MaxSAT encoding that results in a cost-optimal plan with a *single* MaxSAT call.

3 Clause Filtering in Distributed SAT (6 Points)

Distributed clause-sharing solvers like HordeSat and MallobSat filter a repeated clause based on *exact syntactical equivalence* (i.e., if a clause c is shared

[1] Basic online functionality: `https://benchmark-database.de`;
local installation via `pip`: `https://github.com/Udopia/gbd`

[2] `https://satcompetition.github.io/2022/downloads.html`

successfully, then clause c will be blocked for some period). Find a way to generalize this approach to *subsumed* clauses, i.e., if clause c is shared, then all clauses $c' \supseteq c$ are blocked for some period. Provide a rough analysis of the running time complexity and the memory footprint of your approach.

4 Miter Challenge (5+10 Points)

Devise a program which takes two CNF files of formulas F_1 and F_2 and which outputs a CNF F_{neq} such that F_{neq} is satisfiable if and only if $F_1 \not\equiv F_2$. Your program may perform non-trivial reasoning itself (e.g., transformations, simulations, internal SAT calls). The running time of your approach is measured as the running time of your program plus the running time of KISSAT to solve your program's output F_{neq}. You receive 5 points for any correct submission and up to 10 bonus points based on your approach's performance. We will test the approach on (a) equivalent formula pairs, e.g., obtained via preprocessing techniques, and (b) "broken" instances of type (a) that are tampered with to render the instances non-equivalent.