Practical SAT Solving (ST 2026) | Assignment 2

Ashlin Iser, Dominik Schreiber, Niccolò Rigi-Luperti

Algorithm Engineering & Scalable Automated Reasoning (KIT)

2025-05-19 – 2024-06-02

1 Application-specific Analysis (8 Points)

Pick **one** of the 30 largest benchmark families of the Anniversary SAT benchmark set (see lecture 4 slide 5). 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 Ashlin's 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.* Prepare 1–3 slides to present in the exercise.

2 Automated Planning (6+3 Points)

(a) Show that classical automated planning is PSPACE-complete.
Hint: This entails that (a) classical automated planning is in PSPACE and (b) any polynomially space-bounded Turing machine program can be reduced to classical automated planning.

(b) Consider the connection between classical automated planning domains and Turing machines that follows from this result. What is the correspondence of a Turing machine operating deterministically vs. non-deterministically in a classical automated planning domain?

3 Local Search Competition (7(+7) Points)

Using the methods from the lecture, as well as any others you find in the literature or invent yourself, implement an efficient local search SAT solver. The solver should accept one argument, which is the path to an input file in DIMACS format. Easy-to-solve, satisfiable instances can be found in the Global Benchmark Database (GBD) [1]. You can use the parser provided in the lecture [2]. The output format of your solver should adhere to the format used in the SAT competition which is specified online [3]. Solvers will be evaluated on a random set of instances, and we will verify that found models are correct.

[1] **Benchmark Instances:** `https://benchmark-database.de/?minisat1m=yes&result=sat`.

[2] **CNF Parser:** `https://github.com/satlecture/kit2025/blob/main/code/src/util/CNFFormula.h`.

[3] **Output Format:** `https://satcompetition.github.io/2025/output.html`.

Together with your code (and build instructions), submit a single (PDF) slide to present your solution in a two-minute lightning talk. You will receive seven points for a local search solver that can correctly solve easy instances in a few minutes. The overall most convincing submission(s) get(s) up to seven bonus points.

[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`