Satisfiability Problem with MiniSat - Benchmarks, Comprehension and Challenges

Andries Rafael Gabriel, Stoentel Alexandru-Eduard, Nenescu Eugeniu

West University of Timișoara, Bulevardul Vasile Pârvan 4, Timișoara 300223

January 2025

Overview

1. Introduction
2. MiniSat
3. How MiniSat Works
4. Hamiltonian family of problems
5. Benchmark
6. Results
7. Conclusion

Introduction

- Our goal
  - Analyze MiniSat by looking into the installation, challenges, benchmarks, and algorithms.
  - Work on developing modifications to the problems related to the runtime of the MiniSat solver.
- What is MiniSat?
  - The satisfiability of a SAT solver is the key principle of the design of the SAT solver.
- What is the Hamiltonian family of problems and their complexity?
  - Hamiltonian problems test the SAT solver on graph instances.
  - They are NP-Complete
- Benchmark results
  - Comparison and evaluation of performance, and functionality

MiniSat

Installation

- The MiniSat solver is provided in its original repository from GitHub.
- A Linux environment (e.g., Virtual Machine) is recommended to properly use MiniSat.

Challenges

- Configure the Makefile to ensure it is compatible with our systems.
- Potential library dependency issues or compiler version conflicts.

How MiniSat Works

- MiniSat uses Conflict-Driven Clause Learning (CDCL) to solve SAT problems.
- Its functions include: making choices, distributing values, resolving disputes, and developing new statements to eliminate future disputes.
- The search for a solution is directed by unit propagation and decision heuristics.

Hamiltonian Family of Problems

What is the Hamiltonian Problem?

- Determines if a Hamiltonian path or cycle exists in a graph.
- Example: Tour through every city once.

Why is it important?

- NP-complete: Highlights the complexity of SAT solvers.
- Benchmarks SAT solvers on challenging, real-world problems.

Challenges for SAT Solvers

- Graph encoding as SAT formulas.
- Ever-increasing cost as the size of the graph becomes larger.

Benchmarking MiniSat

Setup for Benchmarks

- Problems can be downloaded from SAT Competitions.
- Mandatory use of .cnf files.
- Benchmarks of different sizes of Hamiltonian graphs are included.

Key Metrics

- Runtime (seconds, minutes, or hours per problem).
- Determining if the problem is SAT or UNSAT.
- Memory usage and scalability.

Execution Steps

- Example command: `minisat graph.cnf output.result`
- Analyze runtime and results from `output.result`.

Benchmark example

```
eugen@eugen-VMware-Virtual-Platform:~/minisat-master/build/release/bin$ ./minisat sample.cnf output.out
WARNING: for repeatability, setting FPU to use double precision
============================[ Problem Statistics ]=============================
|                                                                             |
|  - Retained clause: 0 -2
|  - Retained clause: 0 1 2
|  - Retained clause: -1 2
|  - Retained clause: 0 -1 -2
|  Number of variables:             3                                         |
|  Number of clauses:               4                                         |
|  Parse time:                   0.00 s                                       |
|  - Propagating literal: -1
|  Eliminated clauses:           0.00 Mb                                      |
|  Simplification time:          0.00 s                                       |
|                                                                             |
============================[ Search Statistics ]==============================
| Conflicts |          ORIGINAL         |          LEARNT          | Progress |
|           |    Vars  Clauses Literals |    Limit  Clauses Lit/Cl |          |
===============================================================================
===============================================================================
restarts              : 1
conflicts             : 0              (0 /sec)
decisions             : 1              (0.00 % random) (404 /sec)
propagations          : 1              (404 /sec)
conflict literals     : 0              (-nan % deleted)
Memory used           : 5.77 MB
CPU time              : 0.002475 s
```

Figure: Example of running a benchmark

Output of the improved code

```
Adjusted var_decay to: 0.950 after 216 conflicts
Conflict detected with clause: -254 -258
Learned clause: 472 487 181 486 83 -426 91 88 479 392 58 0 139 410 288 483 484 411 451 13 285 385 497 219 221 255 292 50 458 491 480 496 79 372 413 205 477 488 408 207 31 183 44 343 38
8 -110 468 471 299
```

Figure: Example of running a benchmark

Results

Treatments	**MiniSat**
SAT	22
UNSAT	18
Average Time	13 minutes
Runtime	8 hours 30 minutes

Table: Results Table

Conclusion

- MiniSat is a powerful SAT solver based on the CDCL algorithm, efficiently solving complex problems.
- We explored its installation, key components, and benchmarks on Hamiltonian problems.
- Future improvements could focus on optimizing runtime and handling larger problem sizes.