MiniSat Report

Șova Dumitru Ștefan Andrei dumitru.sova01@e-uvt.ro
, Andries Rafael Gabriel rafael.andries00@e-uvt.ro
, Stoentel Alexandru-Eduard alexandru.stoentel01@e-uvt.ro
, and Nenescu Eugeniu eugeniu.nenescu00@e-uvt.ro

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

Abstract. Since time immemorial, logic has helped us make sense of the world that surrounds us, from our ancestors' philosophical inquiries to its use in modern mathematics and informatics. Logic bridges the gap between formal and informal knowledge, proving essential in fields as diverse as databases, programming languages (example (e.g.), Prolog), and propositional logic. This paper focuses on a compelling area within informatics: the Boolean satisfiability problem (SAT), a foundational challenge in computational complexity. SAT asks whether a set of variable assignments can satisfy a Boolean formula and stands as the first problem proven to be Nondeterministic Polynomial-Time complete (NP-complete). Our main focus will be on MiniSat, an SAT solver designed to address this challenge efficiently. Finally, such solvers enabled tackling practical problems with millions of variables, benefiting research and industry.

This report includes benchmarks from recent SAT competitions, alongside minor modifications to MiniSat's original code, to compare results and evaluate its performance across different problem instances. With time results and outputs available on GitHub, this analysis highlights MiniSat's capabilities and underscores the broader impact of SAT solvers in advancing computational problem-solving. GitHub link for the project:
`https://github.com/AndiSova/VF-Software-Engineering-2024-Project`

Keywords: MiniSat · SAT · logic.

1   Introduction

SAT is a cornerstone in computational theory and practical applications, namely electronic design automation (EDA), artificial intelligence, and combinatorial optimization (see [1]). Since 2003, MiniSat has continued to be a small, efficient, and readable SAT solver, representing a perfect tool to introduce students and professionals alike to the nuances of SAT solving (see [2]).

While MiniSat is often praised for its straightforward design, working with it as a beginner can present various challenges. For instance, setting up MiniSat, understanding its underlying algorithms, such as Davis Putnam Logemann Loveland (DPLL) and conflict-driven clause learning (CDCL), and configuring it for experimental benchmarks requires both theoretical and practical knowledge. In addition, interpreting MiniSat's internal workings can be challenging, as it demands familiarity with conflict-driven clause learning, unit propagation, and constraint management, all of which are fundamental to modern SAT solvers' efficiency.

In this report, we will explore these challenges from the perspective of newly introduced users. Our study includes the steps for installing MiniSat, configuring it to run benchmarks from the SAT competition, and analyzing the results. We'll also delve into the algorithms that power MiniSat, examining ways they could be enhanced. Throughout, we apply principles from software engineering, using Unified Modeling Language (UML) diagrams and detailed code documentation to bridge the gap between theoretical concepts and practical implementation. This report aims to provide a comprehensive overview for those new to SAT solvers, emphasizing the complexities and rewards of exploring this powerful problem-solving tool.

1.1   Motivation

Our motivation for studying MiniSat stems from its influential role in advancing SAT solver technology and its accessibility as a learning tool. MiniSat has not only set a standard for efficiency in solving SAT problems but also provides a straightforward and readable code that allows us to delve into its algorithms and structure. It represents a perfect starting line for our introduction into the world of SAT solvers and their particularities, allowing us to understand key SAT-solving techniques like DPLL and CDCL and explore potential improvements in these areas. By examining MiniSat's contributions and considering modifications, we aim to gain a deeper appreciation of SAT solvers' impact on fields like AI, optimization, and verification.

2   SAT problem and solutions

2.1   Problem description

The SAT problem in propositional logic asks to determine in a given formula that combines atomic propositions using the Boolean operators "and" ($\wedge$), "or" ($\vee$), and "not" ($\neg$), whether there are truth values for the atoms in the formula such that the formula evaluates to true. Formally, SAT is the challenge of determining if a logical formula, typically represented in Conjunctive Normal Form (CNF), has at least one solution.

While possible to be resolved by hand, after a certain number of atoms, the solution for such formulas becomes increasingly harder to be determined by ourselves alone, resulting in the need for such solvers to exist. This challenge has fueled the development of specialized SAT solvers, which apply advanced algorithms to efficiently handle these vast and intricate formulas.

With SAT solvers like MiniSat, which are designed specifically to tackle these computational hurdles, we reached a level where it is feasible to solve SAT problems within a reasonable timeframe, even as formula complexity scales up. Because of its ease of modifiability, we will look in this paper at ways that we could improve the current code and compare our results with the original.

2.2   MiniSat installation and first challenges

To begin using MiniSat on our Windows machines, we opted for the Windows Subsystem for Linux (WSL) approach, which allows us to run a Linux environment natively on Windows without the need for virtual machines or terminals such as Cygwin. Below are the steps followed for the installation process:

Here are the steps for installing WSL and Ubuntu: To enable WSL, we ran the command

```
wsl --install
```

in PowerShell, running it as an Administrator. This command installs the required features and the default Linux distribution (Ubuntu) for WSL. However, in some cases, users may encounter issues where the installation hangs or doesn't complete. To resolve this, it's necessary to ensure that Windows is fully updated and that the proper components for WSL 2 are manually enabled. This can be done using PowerShell commands like

```
dism.exe
```

to enable WSL and Virtual Machine Platform features. After completing the steps, the

```
wsl --set-default-version 2
```

command was run to ensure WSL 2 was set as the default version.

Installing MiniSat on Ubuntu: Once Ubuntu was installed from the Microsoft Store(not necessary, Ubuntu can also be installed from its home site as well,

Microsoft Store was used for convenience) and set up with a username and password, the installation of MiniSat was straightforward. First, we updated the package list using the command: sudo apt update. Next, we installed the MiniSat solver by running: sudo apt install minisat.

With MiniSat successfully installed, we can run the solver and evaluate its performance. To ensure that MiniSat is working properly, we can first test its basic functionality by running a sample input. To do this, we simply need to open the Ubuntu terminal and type the following command:

```
minisat
```

This should display MiniSat's usage instructions, confirming that the installation was indeed successful.

Next, to run a benchmark, we used test instances from recent SAT competitions. These competitions provide challenging SAT instances that can be used to evaluate the efficiency and performance of various solvers, including MiniSat.

To run a benchmark, we can follow these steps:

Firstly, we need to download the benchmark files, they are typically available from SAT competition websites, such as `https://satcompetition.github.io/2024/`.

Next, we run MiniSat on a Benchmark. Having downloaded a benchmark file (usually in .cnf format), we can use the following command to run MiniSat on it:

```
minisat <benchmark_file>.cnf
```

List of acronyms

References

1. Authors: Niklas Een, Niklas Sorensson; Institute: Chalmers University of Technology, Sweden; Paper title: An Extensible SAT-solver[extended version 1.2]
2. MiniSat documentation, `http://minisat.se/Main.html`