

LLM-Guided: A Neuro-Symbolic SAT Solver with LLM-Driven Branching

Faraja Bienvenu
Independent Researcher
faraja.bien@gmail.com

Abstract—We present LLM-Guided, a SAT solver for the SAT Competition 2026 Experimental Track that replaces the classical VSIDS branching heuristic with a large language model (LLM). Given an unresolved subformula, the solver queries an LLM to suggest the next branching variable and polarity. The solver is built on a recursive DPLL backbone with unit propagation. It is submitted to the Experimental Track because the use of non-deterministic LLM calls precludes the generation of standard LRAT/DRAT certificates. However, the solver emits a structured “ProofTrace” in DIMACS comments, enabling transparent replay and manual audit of the decision sequence.

I. ALGORITHM AND HEURISTICS

The core architecture follows the Davis-Putnam-Logemann-Loveland (DPLL) procedure [1]. The primary novelty lies in the variable selection heuristic:

A. Unit Propagation

A simple iterative propagation engine is used to resolve unit clauses and detect conflicts.

B. LLM-Guided Selection

At the root of the search (depth 0), the solver serializes up to 50 of the shortest unresolved clauses and queries the LLM for a batch of 10 variable/polarity suggestions.

C. Queue-Based Branching

These suggestions are queued and prioritized. If the queue is exhausted or if suggestions are invalid (e.g., variable already assigned), the solver falls back to a frequency-based heuristic (**VSIDS-lite**), which selects the variable appearing most frequently in unresolved clauses.

D. Transparency

Every decision is recorded as a `ProofStep` with metadata (source, reasoning, confidence).

II. IMPLEMENTATION DETAILS

- **Language:** TypeScript (Node.js ≥ 18).
- **LLM Provider:** GitHub Models (OpenAI-compatible endpoint), model: `gpt-4o-mini`.
- **Packaging:** Distributed as a standalone ESM bundle via `esbuild`.
- **I/O:** Standard DIMACS format for input and SAT Competition conventions for output (`s SATISFIABLE / s UNSATISFIABLE / v` line).

The solver tracks its own LLM token consumption and cost based on standard pricing models (\$0.0001 per 1k input tokens).

III. TRANSPARENCY LAYER (PROOFTRACE)

To address the “black box” nature of LLMs, the solver emits every decision as a `c proof-step: comment` line. This includes the LLM’s reasoning string. A verifier that replays these steps through deterministic unit propagation can independently confirm the result without trusting the LLM’s logical validity.

IV. LIMITATIONS AND FUTURE WORK

As a submission to the Experimental Track, the current implementation lacks **CDCL** (Conflict-Driven Clause Learning) and **watched literals**. The focus is on demonstrating the viability of LLM guidance for structured SAT instances (e.g., factorization circuits) where problem semantics can be captured by the model’s pre-trained knowledge.

REFERENCES

- [1] M. Davis, G. Logemann, and D. Loveland, “A machine program for theorem-proving,” *Communications of the ACM*, vol. 5, no. 7, pp. 394–397, 1962.
- [2] A. Biere et al. (eds.), *Handbook of Satisfiability*, 2nd ed. IOS Press, 2021.