



Extracting Problem Structure with LLMs for Optimized SAT Local Search

André Schidler · Stefan Szeider

Algorithms and Complexity Group
TU Wien, Vienna, Austria



BilateralAI
Bilateral AI - Cluster of Excellence

Motivation and Background

Motivation:

- Modern SAT solvers integrate local search preprocessing to find high-quality starting assignments.
- Generic local search methods miss encoding-specific patterns that can significantly improve solver performance.

Background:

- Previous work on CDCL solvers and local search (e.g., GSAT, WalkSAT) laid the groundwork for this research.

The paper builds on the insight that problem-specific structure—hidden in SAT encodings—can be exploited by tailoring local search algorithms, thereby enhancing the performance of CDCL solvers.

Problem Statement

Core Issue:

- Existing local search methods are general-purpose and do not account for the structural nuances of the encoding scheme.

Objective:

- Automatically generate specialized local search functions by having LLMs interpret PySAT code and extract problem structure.
- Optimize for encoding scheme, abstracted from instance data

Impact:

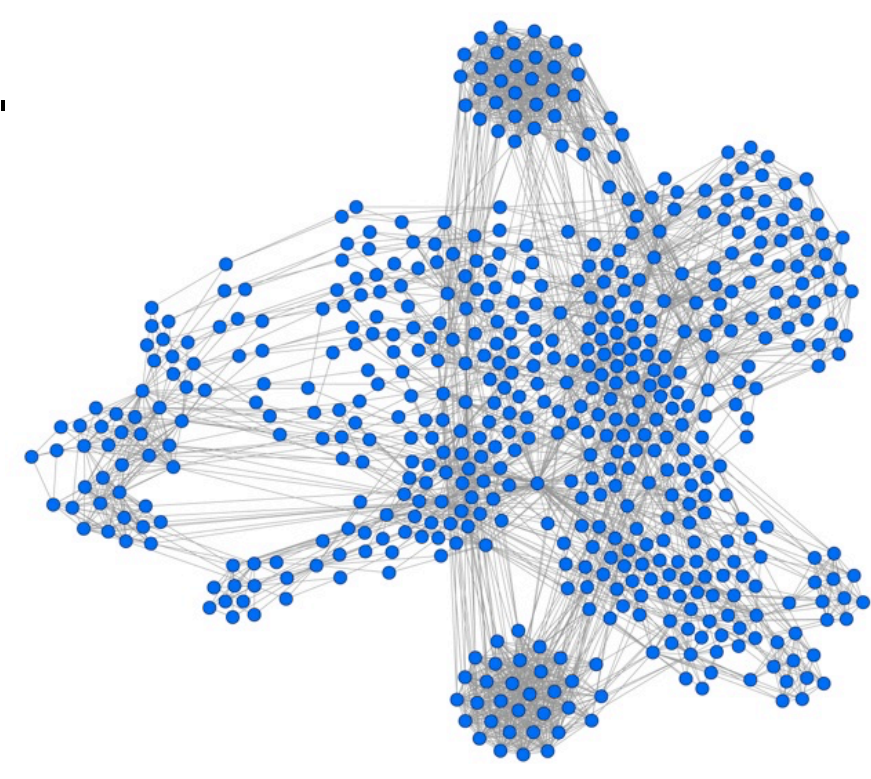
- Tailored local search functions can produce better initial assignments, reducing overall solving time.

encoding scheme

```
16
17
18
19
20
21
22
23
24
25
26
27
28
for u, v in g.edges:
    if u == v:
        continue
    for cl in range(d):
        encoding.append([-pool.icl, pool.icl])
for cni, cn in enumerate(clique):
    encoding.append([pool.id(f"cl{cni}"), pool.id(f"cl{cn}")]
return encoding, pool
```

PySAT

instance data



encoding

c	DIMACS
-1	2 0
-1	3 0
1	-2 -3 0
2	-3 4 0
...	

By focusing on the encoding rather than instance-specific features, the approach aims to generate local search strategies that are effective across all instances sharing the same encoding.

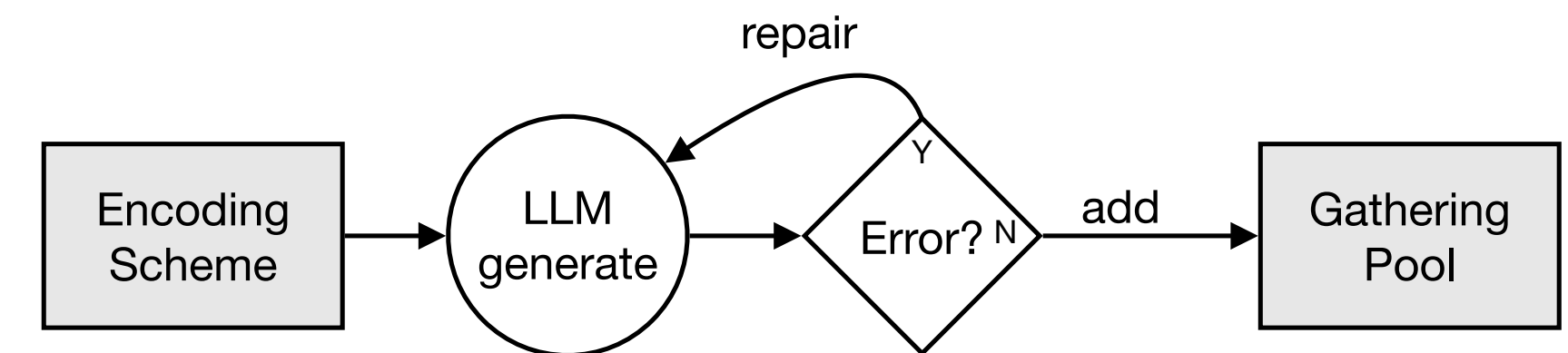
Methodology – Standardizing Encodings and Gathering Phase

Standardization:

- Encoding schemes are expressed in Python using the PySAT framework, making them both human- and LLM-readable.

Gathering Phase:

- LLMs are provided with the PySAT encoding (without problem-specific instance data) and prompted to generate candidate local search functions.



Verification:

- Each candidate is tested for correctness and execution within a specified timeout.

This phase establishes a common representation of SAT encodings and leverages LLMs to generate a diverse pool of candidate local search algorithms through iterative generation and verification.

Methodology – Scoring and Refinement Phase

Scoring:

- Candidate local search functions are evaluated on a set of training instances based on their ability to reduce SAT solving time.

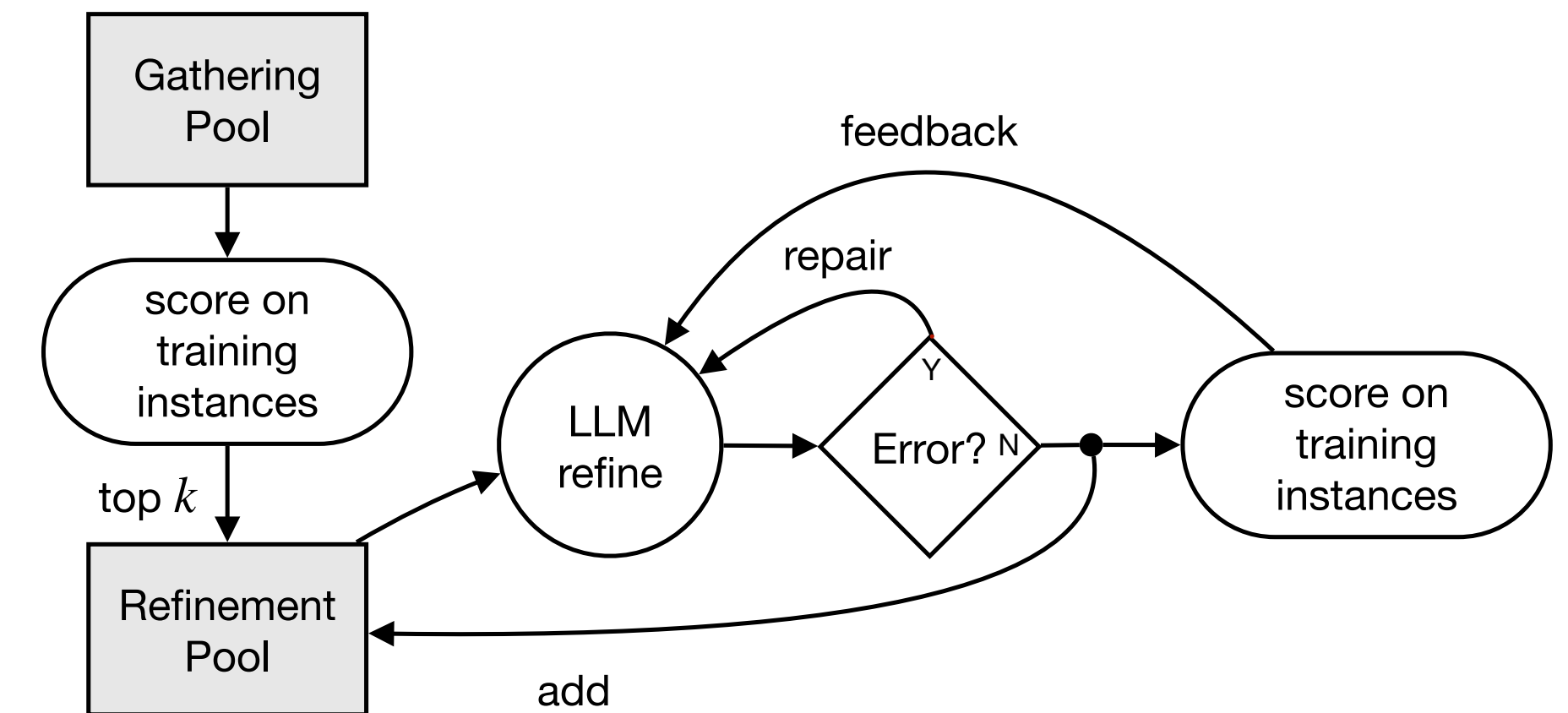
Refinement Phase:

- The top candidates from the Gathering Phase are refined iteratively, with the LLM receiving performance feedback to adjust the function while preserving its core idea.

Outcome:

- This process yields multiple versions (Refined and Structure variants) for each candidate.

The scoring and refinement phases ensure that only the most effective local search prototypes—those that lead to significant runtime improvements—are advanced for final evaluation.



Experimental Setup and Benchmarks

Benchmarks:

- Three encoding schemes are used: Graph Coloring, Directed Feedback Vertex Set (DFVS), and Bounded Depth Decision Trees (BDDT).

Instance Split:

- Training instances (solved by the SAT solver in 10–60 seconds) guide the refinement process, while test instances (taking over one minute) assess final performance.

Solver and Tools:

- Experiments utilize Cadical as the SAT solver and run on both a MacBook M1 (for gathering/refinement) and multi-core servers (for test evaluation).

problem	source	training	test
GC	TreewidthLib3 DIMACS	10	38
DVFS	PACE 2022, Zhou 2016	18	124
BDDT	various	9	32

A diverse set of benchmarks allows the approach to be tested under varying encoding complexities, ensuring the generated local search functions are robust across different types of SAT-encoded problems.

Test Instance Performance

Evaluation Metrics:

- Two key metrics are used: the total number of instances solved by the SAT solver with the local search assignment, and the number of additional instances solved that the SAT solver alone could not.

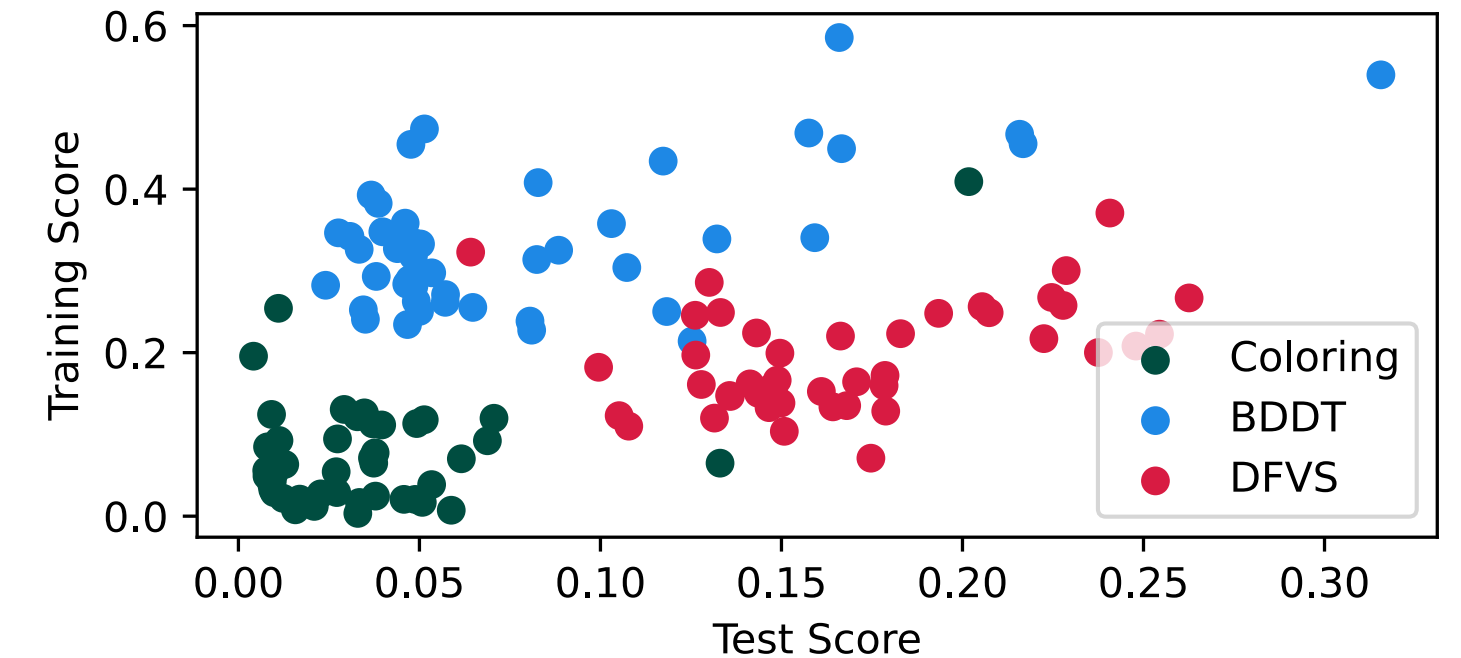
Results Overview:

- For each problem (Graph Coloring, DFVS, BDDT), the performance of the Base, Refined, and Structure versions of the best local search functions is compared.

Observations:

- The results indicate that, in many cases, the refined local search functions significantly improve performance on hard instances.

The experimental results demonstrate that specialized local search functions, particularly those refined via the proposed approach, enable the SAT solver to solve additional hard instances that it could not solve on its own.



instances solved within timeout



Results illustrate that the refined local search functions can enable the SAT solver to solve additional hard instances, validating the approach's effectiveness across different problem encodings.

Analysis of Generated Search Strategies

Diversity and Adaptation:

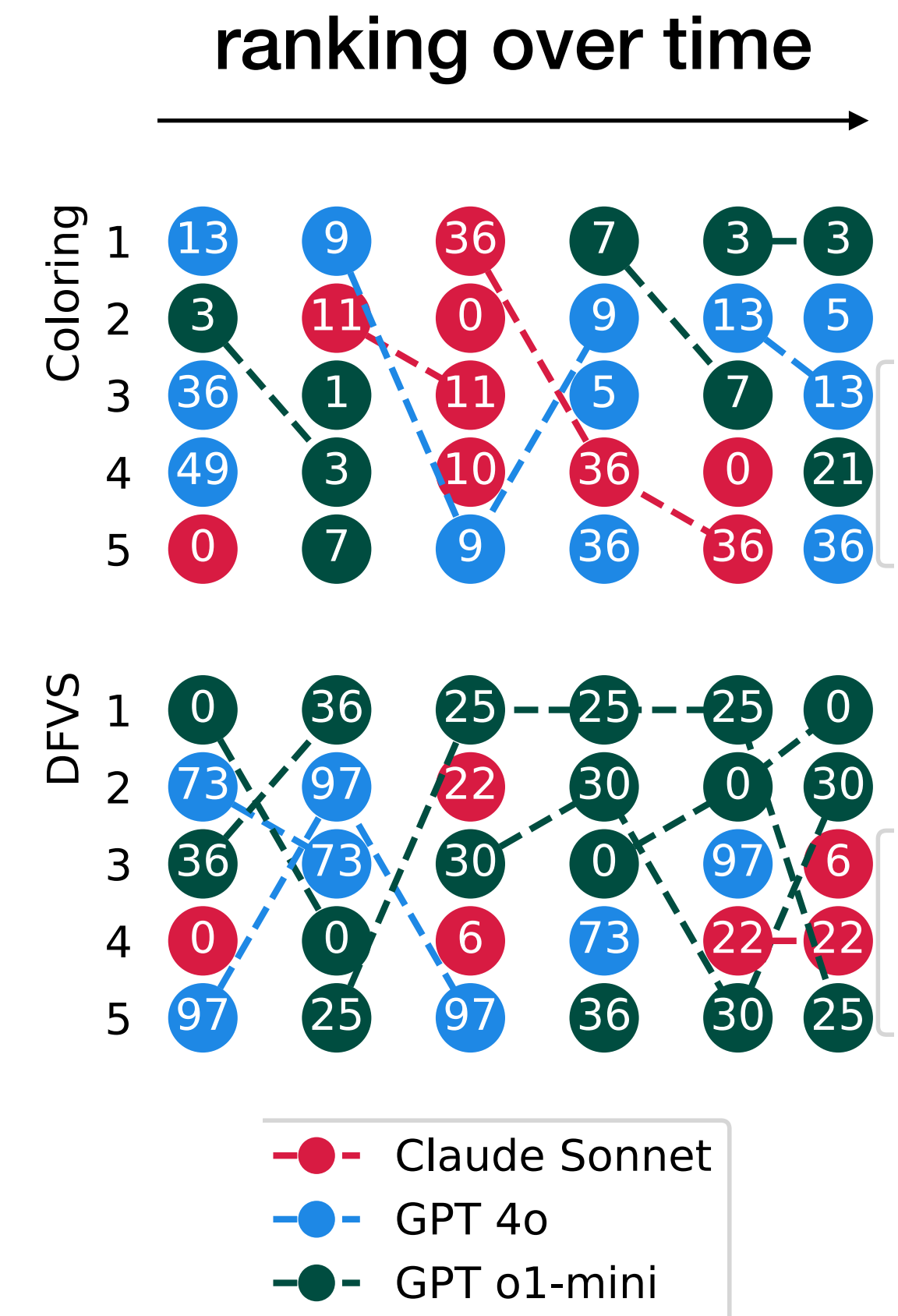
- Generated functions encompass a range of meta-heuristics (e.g., tabu search, WalkSAT, greedy heuristics) adapted to the respective encoding.

Problem-Specific Insights:

- In Graph Coloring, the optimal function employs a combination of tabu search and random selection based on conflict scores.
- For DFVS, a straightforward WalkSAT variant emerged despite some encoding misinterpretations.
- In BDDT, functions are adapted to process variables level by level, reflecting the decision tree structure.

Key Takeaway:

- The approach's success is closely tied to the LLM's understanding of the encoding, highlighting the need for further refinement in some cases



Analyzing the generated strategies reveals both the potential and limitations of the current approach, with diverse methodologies emerging for different problem types.

Discussion and Future Directions

Evaluation Summary:

- The framework demonstrates that LLMs can automatically generate specialized local search functions that improve SAT solving performance.

Challenges:

- Enhancing LLM comprehension of complex encodings and optimizing the performance of the generated code remain critical.

Future Work:

- Investigate distilled LLM variants, utilize model embeddings for diversity assessment, and refine feedback mechanisms during the refinement phase.

Technical Report <https://arxiv.org/abs/2501.14630>