3-SAT Solver Performance Analysis

Arpit Jain

September 21, 2025

A. Objective

The objective is to evaluate the performance of three heuristic search algorithms—**Hill-Climbing (HC)**, **Beam Search (BS)**, and **Variable Neighborhood Descent (VND)**—on randomly generated 3-SAT problems. The evaluation focuses on solution quality (number of satisfied clauses) and computational efficiency.

B. Problem Statement

The 3-SAT problem involves n Boolean variables $\{x_1, x_2, ..., x_n\}$ and m clauses, each with 3 literals. The task is to find an assignment that maximizes the number of satisfied clauses.

State Space: $S = \{0, 1\}^n$ (all 2^n possible assignments)

Goal:

$$s^* = \arg\max_{s \in S} f(s)$$

where $f(s)$ counts the number of clauses satisfied by assignment s.

C. Methodology

- **State Representation:** A state is a complete assignment of Boolean values to all variables.
- **Evaluation Function:** Counts the number of satisfied clauses:

$$f(s) = \sum_{i=1}^{m} C_i(s)$$

  where $C_i(s) = 1$ if clause i is satisfied by s, else 0.

- **Heuristics:**
  - Basic heuristic: maximizes $f(s)$ directly.
  - Make-break heuristic: considers the net gain from flipping a variable.

D. Algorithm Pseudocode

1. Hill-Climbing (HC)

```
FUNCTION HillClimbing(clauses, n, max_iters, restarts):
    best_score ← -
    FOR each restart:
        assignment ← random assignment
        FOR iteration = 1 TO max_iters:
            current_score ← Evaluate(assignment)
            improved ← False
            FOR each variable v:
                Flip v
                new_score ← Evaluate(assignment)
                IF new_score > current_score:
                    improved ← True
                    BREAK
                ELSE:
                    Flip v back
            IF not improved:
                BREAK
            IF current_score > best_score:
                best_score ← current_score
    RETURN best_score
```

2. Beam Search (BS)

```
FUNCTION BeamSearch(clauses, n, beam_width, max_iters):
    beam ← random assignments (beam_width)
    best_score ← -
    FOR iteration = 1 TO max_iters:
        candidates ← []
        FOR assignment IN beam:
            current_score ← Evaluate(assignment)
            IF current_score > best_score:
                best_score ← current_score
            FOR each variable v:
                neighbor ← assignment with v flipped
                candidates.append((Evaluate(neighbor), neighbor))
        Sort candidates by score descending
        beam ← top beam_width assignments
    RETURN best_score
```

3. Variable Neighborhood Descent (VND)

```
FUNCTION VND(clauses, n, max_iters):
    assignment ← random assignment
    best_score ← Evaluate(assignment)
    FOR iteration = 1 TO max_iters:
        improved ← False

        # Neighborhood 1: single flips
        FOR each variable v:
            Flip v
            new_score ← Evaluate(assignment)
            IF new_score > best_score:
                best_score ← new_score
                improved ← True
                BREAK
            Flip v back
        IF improved: CONTINUE

        # Neighborhood 2: pair flips
        FOR each pair (v1,v2):
            Flip v1, v2
            new_score ← Evaluate(assignment)
            IF new_score > best_score:
                best_score ← new_score
                improved ← True
                BREAK
            Flip v1,v2 back
        IF improved: CONTINUE

        # Neighborhood 3: triple flips (random sample)
        FOR k = 1 TO 10:
            v1,v2,v3 ← random 3 variables
            Flip v1,v2,v3
            new_score ← Evaluate(assignment)
            IF new_score > best_score:
                best_score ← new_score
                improved ← True
                BREAK
            Flip v1,v2,v3 back
        IF not improved:
            BREAK
    RETURN best_score
```

E. Mathematical Analysis

- **Hill-Climbing:** Greedy search moves towards the maximum in its immediate neighborhood. May get trapped in local optima.
- **Beam Search:** Keeps top k candidates, improving exploration. Complexity per iteration: $O(beam\_width \times n)$.
- **VND:** Explores increasingly larger neighborhoods to escape local optima. Worst-case complexity is higher but solution quality is better.

F. Results

Qualitative observations:

- HC is fast but often achieves moderate-quality solutions.
- BS balances exploration and exploitation; larger beams improve results.
- VND consistently achieves higher clause satisfaction due to systematic neighborhood exploration.

G. Conclusion

Heuristic algorithms provide practical solutions for 3-SAT. Hill-Climbing is simple and fast but limited by local optima. Beam Search improves solution quality by considering multiple candidates. Variable Neighborhood Descent is the most robust, consistently achieving the highest clause satisfaction and effectively escaping local optima.