A Study on the Impact of Randomness in SAT Solving

Sivasanjai G A

November 2024

Introduction:

Motivation:

Through the years, a variety of methods have been conceived of to tackle the boolean satisfiability problem. However, a lot of them involve highly complex procedures keeping track of various structures and statistics and identifying bottlenecks to solve. In this regard, we consider **Stochastic Local Search** methods, which try to arrive at solutions by locally searching the landscape created by the propositional formula at hand[1]. These methods, while being extremely effective in solving random-SAT problems, are noted for their seeming simplicity.

This study aims to further examine the effectiveness of these methods at a basic implementation level and investigates whether integrating these methods with a back-tracking approach can further increase effectiveness. In the following passages, the problem is first framed appropriately and then the experiments and results are explained.

Framing the local search problem:

For a formula F with n variables and m clauses, one can think the landscape created by the space $\{0,1\}^n\{0,1,..m\}$ where the 2^n truth assignments to the variables of F correspond to points in $\{0,1\}^n$ and the height of such a point is a number $\in 0,1,...,m$ which corresponds to the number of unsatisfied clauses in F caused by the truth assignment/point.[2]

The solution search for SAT under this view is nothing but a search for a global minimum in the above defined landscape. Should such a landscape always have single global minima and no local minima, any greedy descent algorithm would be very effective in obtaining the solution. This in turn opens up the need for search methods which incorporate **random walk** moves to survey the landscape and not get stuck in local minima, which is bound to happen in a purely greedy strategy.

Outline of investigations performed:

First, to demonstrate the need for randomness in stochastic search based methods that we consider, a basic implementation of a purely greedy strategy, a purely random strategy, and a hybrid strategy combining both controlled by a probabilistic parameter p are implemented. As the logical next step, p is varied in $[0, 1]$ to find the optimal value as an indicator for the optimal amount of randomness that needs to be incorporated. Then, a basic implementation of CDCL algorithm is considered and the effectiveness of randomness based approach against it is examined. Furthermore, a comparison with the state-of-the-art solvers such as miniSAT and Glucose-3 are also performed to serve both as a yardstick and to help appreciate the capabilities of the most advanced solvers at present against our basic approaches.

Experiments Performed:

SAT Instances Used:

In-order to perform the study, standard code was used to generate a total of 200 3-SAT instances with up to 200 variables and up to 800 clauses. The generated instances are in the DIMACS conceived '.cnf' format, as is standard. Of these 200 instances, 100 are known to have a satisfiable solution.

Greedy Strategy:

In a greedy strategy, the primary idea is to iteratively improve the current assignment by making the most beneficial variable flip at each step (as in flipping the variable that leads to most number of clauses being satisfied.)[3].

Algorithm:

1. **Initialization:** Start with a random assignment of truth values to all variables.

2. **Evaluation:** Identify all clauses that are currently unsatisfied.

3. **Selection:** For each variable involved in the unsatisfied clauses, calculate the impact of flipping its value. The impact is measured by the net gain in the number of satisfied clauses minus the number of newly unsatisfied clauses.

4. **Flipping:** Select the variable with the highest positive impact (i.e., the one that maximizes the number of satisfied clauses) and flip its value.

5. **Iteration:** Repeat the evaluation and flipping steps until all clauses are satisfied, or a termination condition is met (e.g., a maximum number of iterations).

Purely Random Approach:

The random flipping approach introduces stochasticity into the local search process to mitigate the pitfalls of non-random, i.e, deterministic methods, such as getting stuck in local optima[4].

Algorithm:

1. **Initialization:** Start with a random assignment of truth values to all variables.
2. **Evaluation:** Identify all clauses that are currently unsatisfied.
3. **Selection:** Randomly select a variable from the set of variables involved in the unsatisfied clauses.
4. **Flipping:** Flip the value of the selected variable.
5. **Iteration:** Repeat the evaluation and flipping steps until all clauses are satisfied, or a termination condition is met.

Hybrid Approach:

In a hybrid approach, the decision as to picking a variable to flip randomly, or to pick the one that would lead to most clauses being satisfied is controlled by the probabilistic parameter p.

CDCL Approach:

A basic implementation of the CDCL algorithm is considered. **Algorithm:**

1. **Initialization**: Start with no variable assignments and at decision level 0.
2. **Unit Propagation**: Apply unit propagation to simplify clauses and assign necessary variables.
3. **Conflict Detection**:
   - If unit propagation leads to a conflict:
     - If at decision level 0, declare the formula unsatisfiable.
     - Otherwise, backtrack by reverting the last decision and attempting an alternative assignment.
4. **Decision-Making**: If no conflict arises and not all variables are assigned, select a new variable to assign and proceed.
5. **Iteration**: Repeat the process of unit propagation and conflict detection until a satisfying assignment is found or unsatisfiability is determined.

Discussion on Results:

Upon using the greedy strategy over the 3-SAT CNF files, it is observed that there's a success rate of around 10-20 percentage. Upon using the purely random flipping approach, it has been observed that there's a success rate of around 40 percentage. However, upon combining the greedy and random approaches, it is possible to see that there's a significant improvement in success rate. Upon varying the probabilistic parameter p in discrete increments of 0.1, it seems that the highest success rate is achieved at a $p = 0.4$ leading to a success rate of 55 %. This bests both the pure random and pure greedy strategy.

Next up, upon using the basic CDCL implementation, a success rate of 19 percentage is achieved. Random flipping, however, consistently solves around 44 % of the problems. Upon introducing randomness in the CDCL process, i.e, deciding between selecting variable with most occurrence and a random variable according to p, and deciding between assigning true, and randomly selecting between true and false similarly, a performance improvement of 3.5 % is observed.

From these results, it becomes possible to see within the class of Stochastic Local Search algorithms random walk is a highly effective strategy by itself. Furthermore, upon its combination with other search strategies such as greedy descent, it is justified how the introduction of randomness leads to a sizeable elevation in performance. Even with the basic CDCL algorithm, introduction of randomness leads to a consistent improvement in performance. It could therefore be taken that upon combination with more powerful and complex CDCL based algorithms, randomness may further help enhance their efficiency. MiniSAT, a state-of-the-art lightweight SAT solver[5], clocks in at around 76 percentage on the instances used for the study. This demonstrates just how effective SOTA solvers have become in comparison to the basic implementations used in this study. Hence, it is to be further seen if introducing randomness in an advanced deterministic solver would be enhancing.

Taken together, these experiments demonstrate how randomness can be a useful tool in various approaches to SAT solving.

Limitations and Further Study:

In order to further study and quantify the impact of randomness in SAT solving in a more robust setting, 2 things need to be taken into further consideration. First, the impact of max number of flips allowed needs to be taken into account, as a higher number of flips allowed may lead to increased effectiveness of the random flipping approach. Next up, while this study used a randomly generated set of SAT instances, a carefully curated set with respect to parameters such as clause to variable ratio, number of variables, etc. can be used to further examine whether the strategies used are particular effective or otherwise for a certain class of SAT instances.

References:

1 Inˆes Lynce, 2007, Modern SAT Solving, https://sat.inesc-id.pt/ ines/cp07.pdf

2 Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh: Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications 185, IOS Press 2009, ISBN 978-1-58603-929-5

3 B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In Proceedings of AAAI'92, pages 440–446. MIT Press, 1992.

4 Selman, Bart, Henry A. Kautz, and Bram Cohen. "Noise strategies for improving local search." In AAAI, vol. 94, pp. 337-343. 1994.

5 Eén, N., Sörensson, N. (2004). An Extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds) Theory and Applications of Satisfiability Testing. SAT 2003. Lecture Notes in Computer Science, vol 2919. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24605-3$_3$7

Appendix:

The following are some plots and images for various results.

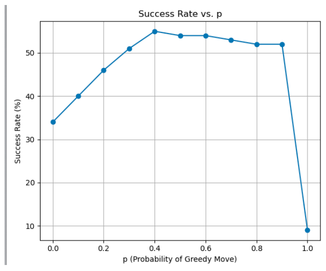
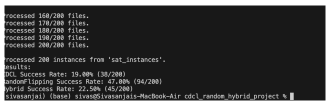


Figure 1: Success-rate vs P for Greedy-Random Hybrid Strategy



```
rocessed 160/200 files.
rocessed 170/200 files.
rocessed 180/200 files.
rocessed 190/200 files.
rocessed 200/200 files.

rocessed 200 instances from 'sat_instances'.
esults:
DCL Success Rate: 19.00% (38/200)
andomFlipping Success Rate: 47.00% (94/200)
ybrid Success Rate: 22.50% (45/200)
sivasanjai) (base) sivas@Sivasanjais-MacBook-Air cdcl_random_hybrid_project %
```

Figure 2: CDCL and Random Flipping Compared

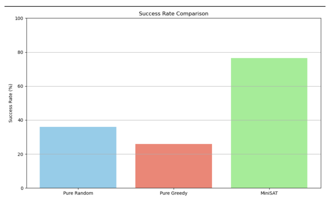


Figure 3: Success Rates of Pure Strategies and MiniSAT