PROJECT REPORT

Using Simulated Annealing as an informed heuristic for DPLL in SAT solving

Submitted To

Department of Computer Science
(Declarative Method: Term Project)

Johns Hopkins University

By

Santosh Bahir
Unmil Tambe

Table of contents

1. Abstract

Stochastic search methods are faster than the systematic approach while solving the SAT problems. But in case of the UNSAT problem, stochastic search methods are not guarantied to reach to the conclusion. In such cases we can’t use systematic solver also because enumerating all the possible variable value combination would lead to exponential blow up. So it is need of the time to get the hybrid solver which is blend of these two different methods. In this project we would try to achieve the same by using simulated annealing as a heuristic to DPLL. Simulated annealing will help the DPLL to choose the next variable or in best case will solve the problem itself.

We will apply these changes to the MINISAT solver since it is open source with good development support.

2. Introduction

Satisfiability(SAT) is the problem of determining if the variables of given Boolean formula can be assigned in such way that the whole formula becomes true or else determine that there is no variable assignment exists.

But SAT problem is the NP-Complete and intractable problem. Lots of research is going on in solving the SAT problem because of its applicability in various domains. Two of the approaches are 'Systematic Approach' and 'Stochastic Approach'.

In Solving the problem, the Stochastic Approach outperforms the Systematic Approach i.e. stochastic SAT solver gives the satisfying assignments far faster than its systematic counterpart. But the Stochastic search Algorithm is not complete algorithm. It means if the Stochastic Solver fails to find the satisfiable solution to given SAT problem; we can’t draw a conclusion about the existence or non-existence of the satisfiable solution. On the other hand Systematic algorithm can prove unsatisfiability because of its completeness. One such complete algorithm is DPLL.

But DPLL will lead to exponential run time in the case of the UNSAT problem. So we need to devise a methodology which would be trade-off between completeness and runtime. One such approach, we are trying in this project is blend stochastic algorithm i.e. simulated annealing with systematic algorithm i.e. DPLL.

3. DPLL and Simulated Annealing Algorithms

DPLL is complete systematic algorithm. The important part in the implementation of the DPLL is the variable ordering. There are different heuristic applied for the variable ordering. We will discuss the DPLL algorithm briefly along with the activity based heuristic for variable ordering.

.1 DPLL Algorithm

```
Function DLL(Φ):
    • while Φ contains at least one unit clause:
        ○ pick any unit clause X
        ○ remove all clauses containing X
        ○ remove ~X from all remaining clauses
        ○ if Φ now has no clauses left,
            ▪ return SAT
        ○ else if Φ now contains an empty clause,
            ▪ return UNSAT
        ○ else
            ▪ pick any variable X that still appears in Φ
            ▪ if DLL(Φ ^ X) = SAT or DLL(Φ ^ ~X) = SAT
                • return SAT
            ▪ else
                • return UNSAT
```

The DPLL algorithm works as follow. Its input us CNF formula. It checks if any unit clause is present if yes it does the unit propagation. If CNF formula does not have any clause left, which means satisfiable variable assignments has found and return UNSAT. If any clause contains empty clause that means satisfiable variable assignment does not exists so return UNSAT.

If there is no unit clause in the CNF formula Φ left, pick any variable X that is present in the Φ, and do the recursive call to the function DLL. Key here is the picking of the picking of variable X. There are different variable ordering proposed. One of them is most constrained which is implemented in the MINISAT. It is knows as activity.

.2 Simulated Annealing Algorithm

- Pick a variable at random
    - If flipping it improves assignment: do it.
    - Else flip anyway with probability p = e -°/T

      Where,

      ° = damage to score

      T = temperature
- As we run, decrease T from high temperature to near 0.

Picking of the variable at random is the important factor here because it will decide if there is any improvement in the total assignment or not. In the case of reduction in the score, we not completely discarding the chances of selecting or flipping the variable. Now, instead of flipping it surely, we will flip the variable with some probability which is p = e -°/T.

Here, deciding the value of Temperature and cooling factor has to be decided by experimental data. Observation is more the temperature, more random the search will be and less chances of getting stuck in the local minima. The cooling factor should be adjusted during the run of the algorithm depending upon how frequent there is damage to the score.

.3 DPLL with Simulated Annealing Algorithm

```
Function DLL(Φ):
    • while Φ contains at least one unit clause:
        ○ pick any unit clause X
        ○ remove all clauses containing X
        ○ remove ~X from all remaining clauses
        ○ if Φ now has no clauses left,
            ▪ return SAT
        ○ else if Φ now contains an empty clause,
            ▪ return UNSAT
        ○ else
            ▪ if with probability p then
                • If SIM_ANNEAL(Φ) then //Do simulated annealing
                    ○ Return SAT
            ▪ pick any variable X that still appears in Φ
            ▪ if DLL(Φ ^ X) = SAT or DLL(Φ ^ ~X) = SAT
                • return SAT
            ▪ else
                • return UNSAT
```

```
Function(SIM_ANNEAL):-
    • Pick a variable at random
    • If flipping it improves assignment: do it.
    • Else flip anyway with probability p = e -°/T
      Where,
            ° = damage to score
            T = temperature
    • As we run, decrease T from high temperature to near 0.
```

In the hybrid version of the DPLL with Simulated Annealing, before choosing the variable for next recursive call to DPLL, we do simulated annealing with some probability p. This is because we don'twant to do the simulated annealing before making each decision in original DPLL. We need to decide the value of the p experimentally by testing against different SAT problems for different p values.

While doing Simulated Annealing if function SIM_ANNEAL returns SAT, then our problem is solved and we can return else we can hope that Simulated Annealing will reduce the problem size. So that when we do the DPLL recursive call after returning from the function SIM_ANNEAL, we would be having smaller problem size.

In simulated annealing we don't do conflict analysis because this will add the learnt clause in the CNF formula and chances of damage to the score rather than increasing the assignment would be more. In simulated annealing, we need to decide the temperature as well as cooling factor experimentally.

Heuristics for p(to do simulated annealing): We need to choose the value for p i.e. probability of simulated annealing such that we can make the use of fastness of stochastic approach as well as utilize the completeness of systematic approach to guarantee the answer.

Heuristics for temperature and cooling factor: The higher the value of temperature, the more random will be choice of the variable. So if there is improvement to the total number of assignments by choosing particular variables then temperature should decrease (high cooling factor) faster when compared to the case wherein there is damage to the score. This could be interpreted as the damage to the score is because of the local minima. And hence we need more randomness to get out of this local minimum.

4. Result of Experiment and Comparison with Standard MiniSat

Note: the dataset here used is the same as the one used by another team for the survey propagation project. Also, all the code changes in the original MINISAT are marked as SIM_ANNEAL_COMM.

Results with Minisat v1.14:

File Name	Restarts	Conflicts	Decisions	Propagations	Conflict Literals	CPU Time
CBS_k3_n100_m411_b10_990.cnf	1	72	128	1970	544	0.004
CBS_k3_n100_m411_b10_991.cnf	3	323	389	8171	2294	0.008
CBS_k3_n100_m411_b10_992.cnf	2	142	182	3497	1077	0.004
CBS_k3_n100_m411_b10_993.cnf	1	86	119	2027	584	0
CBS_k3_n100_m411_b10_994.cnf	1	69	109	1780	467	0.004
CBS_k3_n100_m411_b10_995.cnf	1	29	55	837	199	0
CBS_k3_n100_m411_b10_996.cnf	1	62	90	1483	494	0
CBS_k3_n100_m411_b10_997.cnf	1	74	110	2026	643	0.004
CBS_k3_n100_m411_b10_998.cnf	2	205	278	5371	1537	0.004
CBS_k3_n100_m411_b10_999.cnf	2	112	145	2903	877	0.004

Results with MINISAT-Simulated Annealing:

File Name	**Res Tarts**	**Conflicts**	**Stoch Conflicts**	**Decisions**	**Stoch Guess**	**Propa Gations**	**Conflict Literals**	**CPU Time**
CBS_k3_n100_m411_b10_990.cnf	6	46	1346	71	2246	28853	11666	0.028001
CBS_k3_n100_m411_b10_991.cnf	2	3	201	9	347	4809	1749	0.004
CBS_k3_n100_m411_b10_992.cnf	2	5	115	7	218	2601	1088	0.004
CBS_k3_n100_m411_b10_993.cnf	4	14	543	27	846	12869	4413	0.016001
CBS_k3_n100_m411_b10_994.cnf	4	13	611	32	1054	14655	5314	0.016001
CBS_k3_n100_m411_b10_995.cnf	3	5	281	14	456	6191	2395	0.008
CBS_k3_n100_m411_b10_996.cnf	5	26	799	40	1351	17620	6767	0.020001
CBS_k3_n100_m411_b10_997.cnf	5	22	894	47	1550	20709	8622	0.024001
CBS_k3_n100_m411_b10_998.cnf	5	33	1028	57	1779	23344	9073	0.024001
CBS_k3_n100_m411_b10_999.cnf	6	48	1578	61	2371	35760	14267	0.036002
CBS_k3_n100_m411_b30_990.cnf	6	35	1869	83	2949	40296	14780	0.040002
CBS_k3_n100_m411_b30_991.cnf	3	13	311	12	520	7234	2882	0.004
CBS_k3_n100_m411_b30_992.cnf	9	204	6748	286	9796	144046	57966	0.112007

Analysis:

If we could see the number of conflicts and decisions by original MINSAT are reduced when compared to the MINISAT with Simulated Annealing. But with some probability we are calling SIM_ANNEAL function where we are making random choice for next variable. There we are making huge number of guesses. But since we are not doing this simulated annealing with each call of DPLL this is accepted. Further while doing simulated annealing there is no overhead of calculating activities of variables. But still there is decrease in the performance. It is mainly because the time taken by simulated annealing. And this time is largely dependent upon the different values of temperature and cooling factor.

5. References

- [1] Niklas Een, Niklas Sorensson, An Extensible SAT-solver. In proc. of 6 th international conference on
- theory and application of satisfiability testing, Portofino, Italy, 2003
- [2] Brian Ferris and Jon Froehlick, WalkSat as an informed heuristic to DPLL in SAT Solving,
- Depatment of Computer Science, University of Washington, Seatle, WA 98195
- [3] Prof, Jason Eisner, Department of Computer Science, Johns Hopkins University