Answer 3

1 Introduction

It is also known that 3-SAT exhibits an easy-hard-easy computational pattern. Determining the satisifiability of sets of clauses that are small in relation to the total number of distinct propositional variables in the set is usually easy because there are fewer constraints in assigning truth values to the propositional variables. Determining the satisifiability of sets of clauses that are large in relation to the total number of distinct propositional variables in the set is usually easy because there are too many constraints to assign truth values to the propositional variables and the set is unsatisfiable. Somewhere in between these two extremes the satisfiability problem becomes hard.

2 Generating Test Cases

I wrote a program *generator.py* and used it to generate 16 cases for this question from *file0.cnf* to *file15.cnf*. At the beginning, I only generated 8 cases and test them. Then based on the results, I generated other several case and test them.

Example: *file0.cnf*

c example CNF file with 1000 propositional variables and 100 clauses

p cnf 1000 100

-888 364 762 0

680 217 -942 0

868 -556 -108 0

-616 -640 412 0

…

3 Testing

I tested these 13 cnf files by **minisat** on the CSE machines. I recorded the details in the txt format files from *file0Statistics.txt* to *file15Statistics.txt*. In these two extremes, it’s easy to solve since some of them have zero or almost zero CPU time. Even some of them are unsatisfiable, but the CPU time is very low. Some of cases are as follows.

**Case 0**:

Number of variables: 997

Number of clauses: 100

restarts: 1

conflicts: 0 (-nan /sec)

decisions: 1 (0.00 % random) (inf /sec)

propagations: 0 (-nan /sec)

conflict literals: 0 (-nan % deleted)

Memory used: 5.00 MB

CPU time: 0 s

SATISFIABLE

**Case 1**:

Number of variables: 99

Number of clauses:100

restarts: 1

conflicts: 0 (-nan /sec)

decisions: 1 (0.00 % random) (inf /sec)

propagations: 0 (-nan /sec)

conflict literals: 0 (-nan % deleted)

Memory used: 5.00 MB

CPU time: 0 s

SATISFIABLE

**Case 2**:

Number of variables: 300

Number of clauses: 900

restarts: 1

conflicts: 3 (750 /sec)

decisions: 93 (0.00 % random) (23250 /sec)

propagations: 349 (87250 /sec)

conflict literals: 38 (0.00 % deleted)

Memory used: 5.00 MB

CPU time: 0.004 s

SATISFIABLE

**Case 3:**

Number of variables:300

Number of clauses:1200

restarts: 127

conflicts: 44036 (45119 /sec)

decisions: 55281 (0.00 % random) (56640 /sec)

propagations: 2200239 (2254343 /sec)

conflict literals: 734293 (19.01 % deleted)

Memory used: 5.00 MB

CPU time: 0.976 s

SATISFIABLE

Case 4:

Number of variables:300

Number of clauses: 1500

restarts: 536

conflicts: 235531 (67141 /sec)

decisions: 284650 (0.00 % random) (81143 /sec)

propagations: 10546821 (3006505 /sec)

conflict literals: 2856219 (23.50 % deleted)

Memory used: 5.00 MB

CPU time: 3.508 s

UNSATISFIABLE

**Case 6**:

Number of variables: 300

Number of clauses: 3000

restarts: 7

conflicts: 1057 (33031 /sec)

decisions: 1227 (0.00 % random) (38344 /sec)

propagations: 32902 (1028188 /sec)

conflict literals: 7313 (24.34 % deleted)

Memory used: 5.00 MB

CPU time: 0.032 s

UNSATISFIABLE

**Case 7**:

Number of variables:100

Number of clauses: 1500

restarts: 1conflicts: 34 (8500 /sec)

decisions: 37 (0.00 % random) (9250 /sec)

propagations: 354 (88500 /sec)conflict literals: 98 (14.04 % deleted)Memory used: 5.00 MBCPU time: 0.004 sUNSATISFIABLE

**Case 8**:

Number of variables:100

Number of clauses:2000

restarts: 0

conflicts: 0 (-nan /sec)

decisions: 0 (-nan % random) (-nan /sec)

propagations: 8 (inf /sec)

conflict literals: 0 (-nan % deleted)

Memory used: 5.00 MB

CPU time: 0 s

UNSATISFIABLE

We could see that the most of them are easy to solve except two cases which C is 4 and 5. Therefore, I guess that C value may between 4 and 5 and then generated other cases for testing. Based on the 15 cases, I found that when C approaches 4.4, the CPU time gradually becomes very high and the number of conflicts is very large. It means that the problem becomes very hard. When C equals 4.4, CPU time is 56.484 s. It also has many conflicts and unsatisfiable after a very long time. The details are as follows. If C equals 4.25, CPU time is 7.192s. Except it, when C equals 4.5, the CPU time is 13.408s. When C equals 4.75, the CPU time is 8.636s. You could also see other case in the folder.

**Case 13**:

Number of variables: 300

Number of clauses:1320

restarts: 5121

conflicts: 2973670 (52646 /sec)

decisions: 3599524 (0.00 % random) (63726 /sec)

propagations: 144270703 (2554187 /sec)

conflict literals: 43445740 (22.59 % deleted)

Memory used: 6.00 MB

CPU time: 56.484 s

UNSATISFIABLE

**Case 9**:

Number of variables: 300

Number of clauses: 1275

restarts: 149

conflicts: 49039(46089 /sec)

decisions: 61114(0.00 % random) (57438 /sec)

propagations: 61114(0.00 % random) (57438 /sec)

conflict literals: 783284(18.21 % deleted)

Memory used: 5.00 MB

CPU time: 1.064 s

SATISFIABLE

**Case 10**:

Number of variables: 300

Number of clauses: 1350

restarts: 1534

conflicts: 813813(60696 /sec)

decisions: 981996(0.00 % random) (73240 /sec)

propagations: 38284768(2855368 /sec)

conflict literals: 10755298(23.18 % deleted)

Memory used: 5.00 MB

CPU time: 13.408 s

UNSATISFIABLE

**Case 10**:

Number of variables: 300

Number of clauses: 1425

restarts: 1534

conflicts: 493121(50114 /sec)

decisions: 595337(0.00 % random) (60502 /sec)

propagations: 38284768(2855368 /sec)

conflict literals: 6441324(23.47 % deleted)

Memory used: 5.00 MB

CPU time: 9.84 s

UNSATISFIABLE

4 Conclusion

According to the test, I came up with the constant value C≈4.4 empirically. The number of variables is 300 and the number of clauses is 1320. I also made a line chart (*Line Chart.jpg*) by *linechart.py* in the folder in order to show the conditions of all cases. The 15 cases show the point where the problem is hard to determine.

图表

描述已自动生成