Atanas Golev

SAT Solver Performance Review

Heuristic Searches Implemented

Local Search

The local search takes a two-dimensional integer array representing the clauses and the variables in each clause as an input. Then, it randomly selects each variable as either true or false and builds another two-dimensional integer array representing each variable as 0 or 1 for true or false. Then, it goes through 10000 iterations of a for-loop in which it looks for more satisfiable clauses than there previously was. It does this by selecting a random variable and flipping the bit, and then recomputing the truth matrix. Once it finds a new assignment with more true clauses, it proceeds on to the next iteration where it tries this again.

Because the local search was getting stuck into local maxima frequently, I also included a *walkSideways* boolean. Once the for-loop randomly selects an arbitrary number of times without success (currently set the number of variables in the file) the *walkSideways* boolean is flipped. This enables the program to shift to a different bit configuration of the variables if the number of clauses satisfied is equal to the previous number of clauses satisfied. This also turns *walkSideways* back off, after which we look for a better assignment once again, until the program hits the maximum limit of random searches.

WalkSAT

WalkSAT also takes a two-dimensional integer array representing the clauses and variables as input. It randomly selects a clause, after which it negates each of the three variables, determining which of the three flips satisfies the most clauses.

If this number is greater or equal to than the number of clauses previously satisfied, that bit is flipped in the variable table. WalkSAT also goes through 10000 iterations before giving up.

Graphs

The exact initial random assignment of variables for is used for both WalkSat and Local Search. This initial assignment is different for each trial.

Graphs of how WalkSAT performed in comparison to Local Search, based on the number of clauses in the CNF file are shown below. All data points are averages of the 10 files for each CNF number of clauses (for the regular category) and averages for all 50 files (in the hard category). Those, themselves, are averages of 10 different runs of the program, because both of these algorithms required random assignments. For the exact values, refer to results.csv.

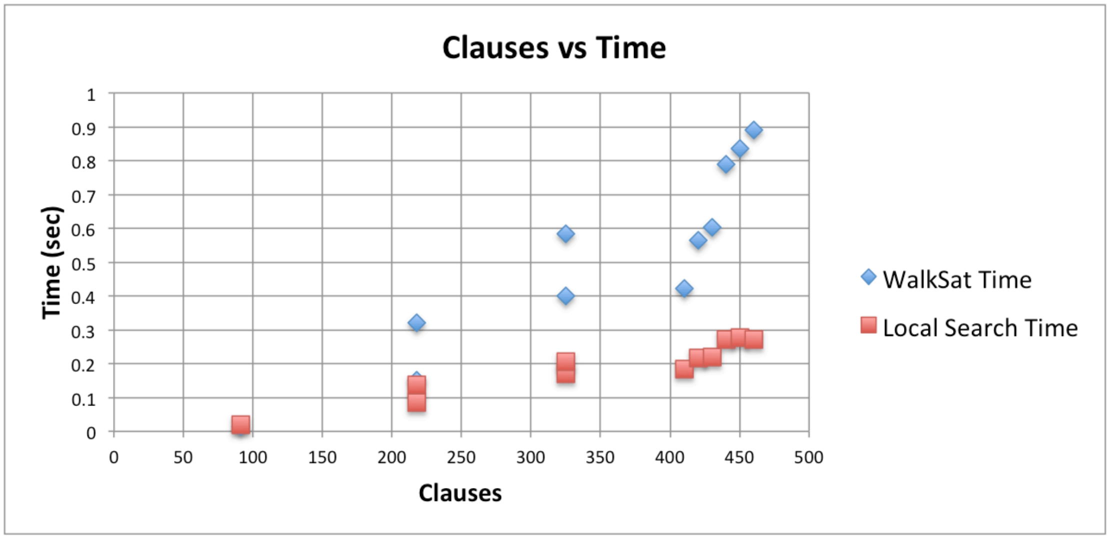

Figure 1:Local Search was quicker than WalkSAT in all cases.

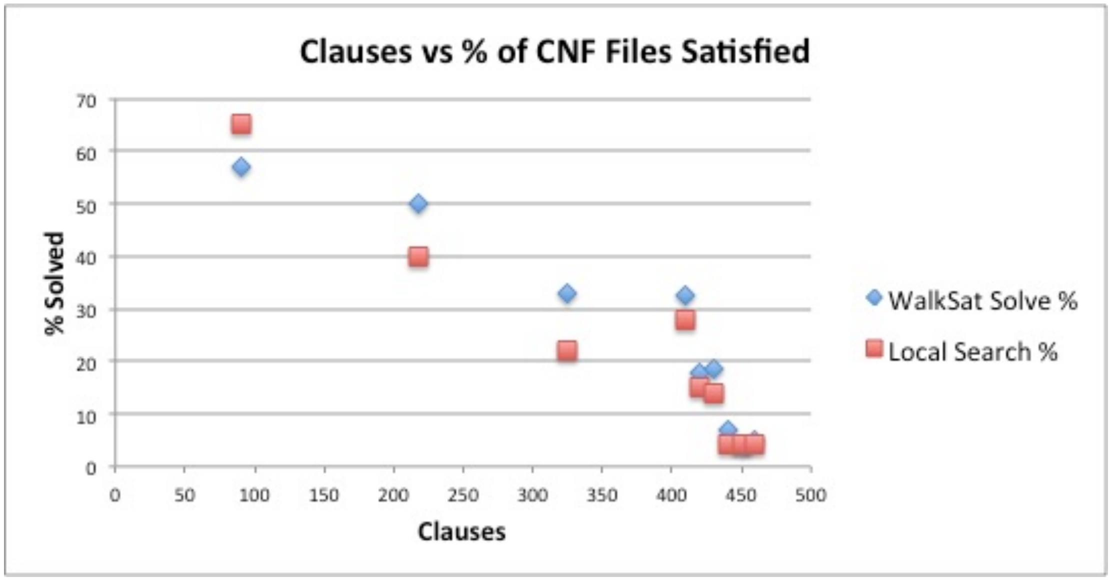

Figure 2: WalkSAT solved a greater percentage of the CNF files in all cases except for the n=91 clauses.

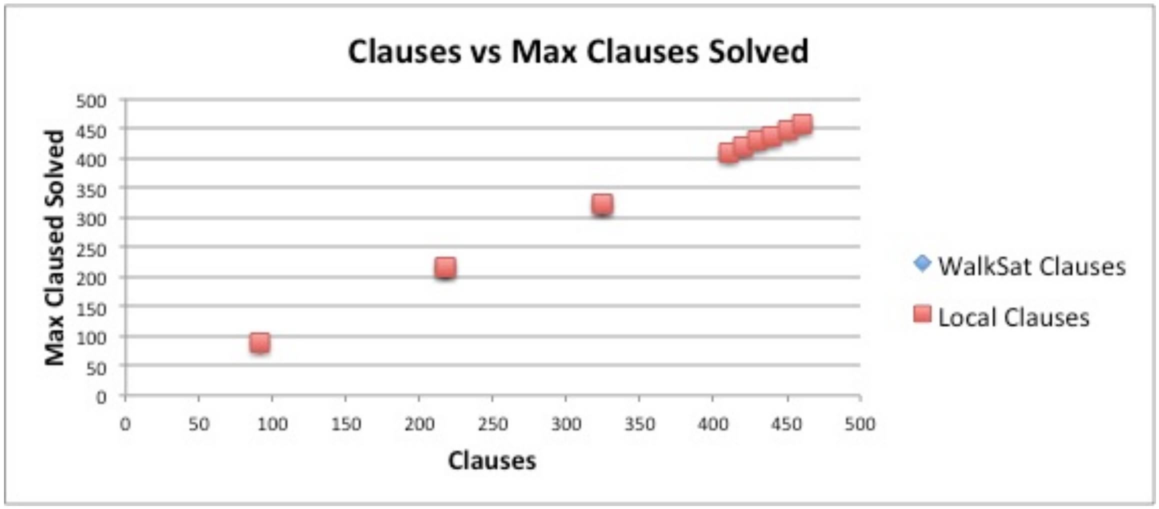

Figure 3: WalkSAT and Local Search were virtually identical in terms of the max number of clauses satisfied in each condition. You can't see the blue dots because hey are overlapping at all points.

Summary of Data

Clauses	Variables	WalkSAT Time	Local Search Time	WalkSAT Max Clauses	Local Search Max Clauses
91	20	0.01328493	0.02061483	90.41	90.57
218	50	0.15134064	0.08489582	217.28	217.18
325	75	0.4011948	0.1700439	323.71	323.67
218	*50*	*0.3217199*	*0.1375117*	*216.28*	*216.34*
325	*75*	*0.5835804*	*0.2070236*	*322.88*	*322.79*
410	100	0.423451102	0.183540222	408.95	408.856
420	100	0.564921112	0.215701426	418.53	418.442
430	100	0.60384242	0.219494912	428.34	428.274
440	100	0.7899138	0.27011474	437.792	437.7
450	100	0.83490768	0.27796338	447.442	447.414
460	100	0.89097704	0.27044726	457.252	457.192

Table 1: Averages for Each Condition based on Clauses and Variables. The rows in italics are the unsatisfiable files.

Observations

Some of the easy .cnf files were very rarely satisfied. In one case, *uf75-03.cnf,* it was only satisfied one single time out of 50 trials by WalkSAT, and wasn't satisfied at all in those trials by Local Search. This makes me feel like those files had configurations which needed a very precise first random assignment in order to be solved.