Ilia Benson

Comp 610

Prof. Noga

Max2Sat Approach

Some usage and disclaimers can be found in the readme. The repo can be found here: https://github.com/iliabenson/2sat

My approach to the project was very simple, there are a lot of sub routines but it is fairly straight forward. Aside from doing the regular checking of number of parameters passed in the algorithm does the following:

* Reads in the values and creates an array of “terminals” these are just all variables, with repetition, in all the clauses as they appear
* It then calculates the used variables, ignoring missing ones, as in the case of test1.txt, and marking them with an X for later printing, this is done to signify that the value of this variable does not effect this instance of max2sat because it does not appear in any clause and thus is not in any solution of this instance
* It then generates a variable table where all information about each variable is kept and updated throughout the program
* Finally a reverse table is built, this just maps the variables in the variable table to their position in the input file, mostly used for easy variable value update just before an evaluation takes place

My program uses a weighted greedy search. That is it calculates the number of times a variable appears in the input file, both negated and non-negated, this becomes the starting weight value for each variable. Each variable gets 2 weights negated and non-negated this way. The one with the highest weight value for negated or non-negated, is chosen and satisfied. Each variable that shows up with selected variable in the input file has it’s weight decreased by one for each appearance to signify that it cannot satisfy that clause as that clause is already satisfied. This is repeated for all variables except ones that have a weight of 0, they are skipped since evaluating them does not add any value to the potential solution.

The output from greedy becomes the input for local search. 2 variants of local search were chosen, both are similar. The first is unSat local search, this variant looks at the set of clauses with selected values for the variables and creates a subset of unsatisfied clauses. From that subset it selects a random clause and from that clause a random variable and flips it value. And updates to all the clauses where that variable appears is performed and then an evaluation is performed on all the clauses with the hopes to find a better result. If a worse result is produced the chosen variable is flipped back, all clauses that it appears in in the unsat array are removed and a new one is selected at random from the remaining unsat clauses and the process repeats. If a selected flip increases the resulting values of max2sat, that value is not flipped back and a new unsat clause array is built and the process repeats. This processes is ran equal to the amount of clauses in the input file, however a maximum tries value maybe given as input.

The second local search does the exact same thing as the first except the random variable to flip is selected from all of the clauses. They are both enabled to run in tandum but it can be easily disabled by commenting out a call to one of them inside the main function, this will produce a compiler warning though. In any case the result, without truth assignments, of the highest amount of variables satisfied is printed at the end of each algorithms run.