Compilation

The file cnf.py is used to generate k literals, m clauses with n symbols in a new file "cnf". C++11 extension is used in WalkSAT.cc, so it should be compiled with command "g++ -std=c++11 WalkSAT.cc", and it should run with command "./a.out cnf" with cnf file. "Clause.h" is the header file to the C++ implementation of WalkSAT. The python version should be run with command "python3 WalkSat.py cnf".

Explanation

WalkSAT class structure is stated as below. There's a vector *clauses* to store the clauses, and an unordered map *model* to store the symbols and their Boolean values. A clause *ranclause* is used later. RAN_WALK_PROB and MAX_FLIPS are the probability and maximum of flips, which are set as 0.5 and 10^5 as default.

There are two public functions. The function **Walksat()** is the constructor for the algorithm, which reads the input and add the clauses to model. The function int **algorithm()** is the main body of the algorithm. If the current model is logically true, it should break and return current flips. Otherwise, it sets random clause that is false under the given model from clauses. Then, flip the value in model of a randomly selected symbol from clause, or run flipmax().

Three private functions are built. The function bool **sat()** checks if the model is satisfied, and **flipmax()** flips the value of the symbol in the model that increases the most satisfied clauses. The function void **getclauses()** parses the file and pushes clauses to the vector *clauses*.

Experiment

There are 3 literals and 200 symbols. The ratio of clauses to symbols varies from 1 to 9, which means number of clauses increases from 200 to 1800.

Ratio	1	2	3	4	5	6	7	8	9
C++ time	0.0045s	0.1575s	0.0564s	20.9436s	fail	fail	fail	fail	fail
C++ flips	38	66	238	61866	fail	fail	fail	fail	fail
Py time	fail	fail	fail	fail	fail	fail	fail	fail	fail
Py flips	fail	fail	fail	fail	fail	fail	fail	fail	fail
Minisat time	0.006s	0.007s	0.001s	0.002s	0.001s	0.001s	0.001s	0.003s	0.003s

I find that python version of WalkSAT cannot complete within the bound of 10^5 flips if there are 200 symbols, and C++ version of WalkSAT fails as well when the ratio increases up to 5. However, Minisat runs quite fast and even faster as size increases.

Then, in order to explore the performance of Python version, I decrease the size of symbols to 50 and do the experiment again, and size of clauses varies from 50 to 450.

Ratio	1	2	3	4	5	6	7	8	9
C++ time	0.0002s	0.0015s	0.0079s	0.274s	Fail	Fail	fail	Fail	fail
C++ flips	5	19	64	3003	fail	Fail	Fail	Fail	Fail
Py time	0.0503s	0.0274s	0.1460s	0.1376s	fail	Fail	Fail	Fail	Fail
Py flips	279	130	908	916	fail	Fail	Fail	fail	Fail
Minisat time	0.001s	0.001s	0.001s	0.002s	0.001s	0.001s	0.001s	0.001s	0.001s

After the size of symbols decreases, the python version is able to complete within the bound of 10^5 flips, while both versions fail after the ratio increases up to 5. The Minisat version remains the efficiency around 0.001s.