DEEC
DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

INSTITUTO SUPERIOR TÉCNICO

Artificial Intelligence and Decision Systems (IASD)
Lab classes, 2015/2016

Assignment #2

Version 1.0 - October 23, 2015

SAT is the abbreviation for the satisfiability problem: given a propositional sentence, determine if it is satisfiable, and if it is, show which propositions have to be true to make the sentence true. The SAT problem can be defined as follows:

- A *literal* is a proposition symbol or its negation (e.g., P or $\neg P$).
- A *clause* is a disjunction of literals; a k-clause is a disjunction of exactly k literals (e.g., $P \vee Q \vee \neg R$).
- A sentence in CNF (*conjuntive normal form*) is a conjunction of clauses; a k-CNF sentence is a sentence of k-clauses.

For example,

$$(P \vee Q \vee \neg S) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee \neg R \vee \neg S) \wedge (P \vee \neg S \vee T)$$

is a 3-CNF sentence with four clauses and five propositional symbols.

In this assignment, you have to implement and test three algorihms for solving SAT problems that have been used to investigate how hard SAT problems are. The algorithms to be used are:

1. GSAT (explained below)
2. WalkSAT
3. DPLL

GSAT is a random-restart, hill-climbing search algorithm[1]. Next, a function that implements GSAT is presented.

[1]See section 4.1.1 of the course's book for more information concerning hill-climbing search.

```
*function* GSAT(*sentence*, *max-restarts*, *max-climbs*) returns a truth assignment or failure
    for $i \leftarrow 1$ to *max-restarts* do
        $A \leftarrow$ A randomly generated truth assignment
        for $j \leftarrow 1$ to *max-climbs* do
            if $A$ satisfies *sentence* then return $A$
            $A \leftarrow$ a random choice of one of the best successors of $A$
        end
    end
    return failure
```

The initial state is a random assignment of true or false to the proposition symbols. For instance, for the preceding 3-CNF sentence, we might start with P and Q false and R, S and T true.

The evaluation function measures the number of satisfied clauses, that is, clauses with at least one *true* disjunct. Thus, the initial state gets an evaluation of 3, because the second, third and fourth clauses are true. If there are n proposition symbols, then there are n operators, where each operator is to change the truth assignment for one of the symbols. As a hill-climbing search, we always use the operator that yields the best evaluation (`best successor of A`), randomly choosing one if there are several equally good operators (`random choice of best successors of A`). Our example 3-CNF sentence is solved in one step, because changing S from true to false yields a solution. As with random-restart algorithm, unless we find a solution after a certain amount of hill-climbing, we give up and start over from a new random truth assignment. After a certain number of restarts, we give up entirely.

The other two algorithms (DPLL and WalkSAT) are extensively described in the course's book, in section 7.6.1 and section 7.6.2, respectively.

The input file format should be the DIMACS format:

```
c
c start with comments
c
c
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

- The file can start with (optional) comments, that is lines beginning with the character c:

  c start with comments

- Right after the comments, there is the line starting with a p:

  p FORMAT VARIABLES CLAUSES

where FORMAT is a string indicating the format of the instance file (for this assignment, you may consider that all instances are in CNF format; so FORMAT = cnf); VARIABLES is an integer establishing the exact number of variables appearing in the file; and CLAUSES is also an integer defining the exact number of clauses contained in the file.

- Then the clauses follow. Each clause is a sequence of distinct non-null numbers between -VARIABLES and +VARIABLES ending with 0. Positive numbers denote the corresponding variables. Negative numbers denote the negation of the corresponding variables.

The output file format should also be in the DIMACS format.

A detailed description of the DIMACS format, both the input and the output files, can be found in
`http://www.domagoj-babic.com/uploads/ResearchProjects/Spear/dimacs-cnf.pdf`

In the following website you will find many SAT benchmark problems that you can use to test your algorithms:
`http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`

Assignment Questions

For each of the three algorithms:

(i) Is the algorithm sound? Is it complete?

(ii) Implement it using Python.

(iii) Use the algorithm to solve randomly 3SAT problems of different sizes. There are two key parameters: N, the number of propositional symbols, and C, the number of clauses. You have to investigate the effects of the ratio C/N on the execution time of the algorithm.

(iv) What can you conclude about the difficulty of 3SAT problems for different values of C, N and ratio C/N?

(v) What can you say about the relative effectiveness of the three algorithms for solving SAT problems?

Note 1: All code should be adequately commented. In particular, for each function you should include comments describing the function's purpose, inputs and outputs.

Note 2: Do not forget to identify the version of Python used.

The deliverable of this Lab Assignment consists of the following components:

- the Python source code
- the short report (pdf format) with no more than 4 pages.

- all input and output files used to test the algorithms and mentioned in the report.

Deadline: 24h00 of 21-Nov-2015, by email to lmmc@isr.ist.utl.pt