ENSTA Paris
2022-2023

Logical models for Artificial Intelligence – INF656L

Alexandre Chapoutot

SAT Part - Practical work 2

Goal(s)

- ✭ Implementation of WalkSAT algorithm

Exercise 1 – The Labyrinth Guardians.

You are walking in a labyrinth and all of a sudden you find yourself in front of three possible roads: the road on your left is paved with gold, the one in front of you is paved with marble, while the one on your right is made of small stones. Each street is protected by a guardian. You talk to the guardians and this is what they tell you:

- *The guardian of the gold street*: "This road will bring you straight to the center. Moreover, if the stones take you to the center, then also the marble takes you to the center."
- *The guardian of the marble street*: "Neither the gold nor the stones will take you to the center."
- *The guardian of the stone street*: "Follow the gold and you'll reach the center, follow the marble and you will be lost."

Given that you know that all the guardians are liars, can you choose a road being sure that it will lead you to the center of the labyrinth? If this is the case, which road you choose?

Exercise 2 – WalkSAT Algorithm

We will use a simple representation of the Boolean constraints in CNF. Specifically, we will consider a data structure of list of integer lists as in the case of the TD1.

Question 1

Inspired by the pseudo-code[1] given to Figure , implement this algorithm (*e.g.*, in Python)

To test your solver some problems in DIMACS format can be found on

`https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`

A Programming help

Python's SymPy library provides an implementation of the DPLL/CDCL algorithm that you may find useful in verifying your WalkSAT implementation.

For example, the Python source code uses the DPLL algorithm in Function `satisfiable`

[1]Image coming from *Artificial Intelligence: A Modern Approach*

```
function WALKSAT(clauses, p, max-flips) returns a satisfying model or failure
    inputs: clauses, a set of clauses in propositional logic
                p, the probability of choosing to do a "random walk" move, typically around 0.5
                max-flips, number of flips allowed before giving up
    model ← a random assignment of true/false to the symbols in clauses
    for $i = 1$ to max-flips do
        if model satisfies clauses then return model
        clause ← a randomly selected clause from clauses that is false in model
        with probability p flip the value in model of a randomly selected symbol from clause
        else flip whichever symbol in clause maximizes the number of satisfied clauses
    return failure
```

Figure 1: Pseudo code WalkSAT

```python
from sympy.logic.boolalg import And, Or, Implies, Equivalent, Not, to_cnf
from sympy.abc import p, q, r
from sympy.logic.inference import satisfiable

expr = Implies(p, Equivalent (q, r))
print(expr)

expr_cnf = to_cnf(expr)
print(expr_cnf)

print(satisfiable(expr_cnf))

from sympy.logic.utilities.dimacs import load
expr2_cnf = load('1 2 \n 3')
print(expr2_cnf)
print(satisfiable(expr2_cnf))
```