SAT Solver Report
CS508 Optimization Methods

Rajat Shukla (194101039)
Computer Science and Engineering, IIT Guwahati
rajat.shukla@iitg.ac.in

1) Type of SAT solver:

Backtracking SAT solver is implemented by using the DPLL (Davis–Putnam–Logemann–Loveland) algorithm. DPLL is a complete, backtracking search algorithm for deciding the satisfiability of SAT problems.

Backtracking algorithms recursively assign truth values to each literal, one at a time, and tries to build the solution. The algorithm backtracks to reverse the assignment of already assigned literals when the formula becomes unsatisfiable. It returns unsatisfiable when the formula becomes unsatisfiable and it is not possible to backtrack..

2) Idea Behind Implementation:

The idea is to implement the DPLL algorithm with the **Jersolow Wang** decision heuristic.

2.1) DPLL algorithm:

```
// Returns true if the CNF formula F is
// satisfiable; otherwise returns false.

DPLL(F)
  G ← BCP(F)
  if G = ⊤ then return true
  if G = ⊥ then return false
  p ← choose(vars(F'))
  return DPLL(G{p ↦ ⊤}) = "SAT" ||
         DPLL(G{p ↦ ⊥})
```

Reference:(https://courses.cs.washington.edu/courses/cse507/14au/slides/L3.pdf)

The BCP() method in the above algorithm recursively finds out unit clauses (containing only one literal) and assign truth values to them.

If the formula returned by the BCP() is satisfied, then return satisfiable.

If it is unsatisfiable return unsatisfiable (since unit clause contain only one literal to make the clause true the containing literal must be true, if it is not possible to make some other clause true with the truth value of this literal then the formula becomes unsatisfiable)

The next step to decide which variable to choose. The performance of the SAT solver depends a lot on the decision heuristic that we choose.

Once the literal is chosen recur the same process for true value of the chosen literal as well as for the false value of the chosen literal. If for both values of chosen literal formula becomes unsatisfiable then return unsatisfiable otherwise return satisfiable.

Example:

(p v q) and (¬q v ¬r) and (¬p v q) and (¬q v r)

Since there is no unit clause we decide:

p->true:

Backtrack

the formula becomes

(¬q v ¬r) and (q) and (¬q v r)

Now the formula contains (q) as unit clause. Assigning q as true creates a conflict:

(r) and (¬r)

Backtrack to change the last decision made and restart from there.

p->false:

the formula becomes

(q) and (¬q v ¬r) and (¬q v r)

Now the formula contains (q) as unit clause. Assigning q as true creates a conflict:

(r) and (¬r)

Backtrack to change the last decision made and restart from there.

Since there is no decision left to change the formula becomes unsatisfiable.

2.2) Jersolow-Wang method:

Jersolow-Wang two sided decision heuristic method is implemented for the deciding which literal to choose in the DPLL algorithm.

This method gives the score to each literal according to the length of clauses in which the literal is present. Score is inversely proportional to the length of clauses.

$$\text{Score\_of\_literal}(x) = \Sigma\, 2^{\wedge -|c|} \quad \text{for each clause c that contains literal x or -x}$$

Where |c| is the length of clause c

In Jersolow-Wang one sided method score for both polarity of a literal is calculated separately where as in Jersolow-Wang two sided method only one score is calculated for a literal (both x and –x are consider for score calculation of literal x).

This gives an exponentially higher weight to literals in shorter clauses and literal with maximum score is chosen.

Refernce:https://ths.rwth-aachen.de/wp-content/uploads/sites/4/teaching/vorlesung_satchecking/ws14_15/02a_sat_handout.pdf

3) Organization of the Code:

The code for the SAT solver have the following four functions:

1. if __name__=="__main__"
2. dpll_algorithm(sentence)
3. remove_unit_literals_recursively(sentence, flag)
4. jersolow_wang_2_sided_method(sentence)
5. assign(literal)

3.1) if __name__=="__main__"

This function is the main function of the code which asks for the cnf file name from the user. It opens the file entered by the user, store the formula to a list and call the dpll_algorithm() method. It also prints the result of the given SAT problem.

3.2) dpll_algorithm(sentence)

This method implements the DPLL algorithm which is discussed in the above section 2.1. first it calls the remove_unit_literals_recursively() method assign the values to all the unit clause's literal. If the formula comes out to be satisfiable or unsatisfiable then return the result to the main function else call the function jersolow_wang_2_sided_method() for choosing next literal for assignment. Once the literal is chosen its call itself recursively for both true and false value of the chosen literal.

3.3) remove_unit_literals_recursively(sentence, flag)

This method assign the truth values to the unit clauses and updates the formula by replacing the value of unit clause literal with its truth values in all the clauses where it is present. It returns the updated formula to the dpll_algorithm() method.

This method have two parts if part and the else part. The if part runs when no literal is passed to this method and the else part runs when the chosen literal in the dpll_algorithm() method is passed to this function in the flag variable.

4.4) jersolow_wang_2_sided_method(sentence)

This method implements the above section 2.2 and return the literal with maximum score to the dpll_algorithm() method.

4.5) assign(literal)

assign(literal) method simply assign the literal passed to it with true/false value based on its polarity.