MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Metaheuristic Optimization

Lab: Satisfiability Checker

Propositional Satisfiability Problems

The Propositional Satisfiability Problem (SAT) can be represented by a pair F = (V, C) where, V indicates a set of Boolean variables and C a set of clauses in conjunctive normal form (CNF).

The following formula shows a SAT example described by means of the CNF.

$$F = (x_1 \vee x_3 \vee \text{-}x_4) \wedge (x_4) \wedge (x_2 \vee \text{-}x_3)$$

Where $\vee$ represents the ***OR*** Boolean connective, $\wedge$ represents **AND,** and $\text{-}x_i$ is the negation of x_i. Given a set of clauses, $C_1, C_2, \ldots, C_m$ on the variables $x_1, x_2, \ldots, x_n$, the satisfiability problem is to determine if the formula

$$C_1 \wedge C_2 \wedge \ldots \wedge C_m$$

is satisfiable. In other words, is there an assignment of True/False to the variables so that the above formula evaluates to *True*?

For instance, a potential solution for F would be

$$x_1 = \text{True}, x_2 = \text{False}, x_3 = \text{False}, x_4 = \text{True}.$$

X_1 satisfies the first clause, X_4 satisfies the second clause, and $\text{-}x_3$ satisfies the last clause.

File format for Satisfiability Problems

The preamble. The preamble contains information about the instance. This information is contained in lines. Each line begins with a single character (followed by a space) that determines the type of line. These types are as follows:

- **Comments** : comment line give human-readable information about the file and are ignored by programs. Comment lines appear at the beginning of the preamble. Each line begins with a lower-case character c

  c This is an example of a comment line

- **Problem line**: there is one problem line per input file. The problem line must appear before any node or arc descriptor lines. For cnf instances, the problem line has the following format, the problem line begins with a lower-case character p

  p FORMAT VARIABLES CLAUSES

  The FORMAT field allows programs to determine the format that will be expected, and should contain the word “cnf”. The VARIABLES field contains an integer value specifying ***n***, the number of variables in the instance. The CLAUSES field contains an integer value specifying ***m***, the number of clauses in the instance. This line must occur as the last line of the preamble.

- **The CLAUSES** . The clauses appear immediately after the problem line. The variables are assumed to be numbered from 1 up to n. It is not necessary that every variable appear in an instance. Each clause will be represented by a sequence of number, each separated by either a space, a tab, or a newline character. The non-negated version of a variable i is represented by i; the negated version is represented by -i.

  Each clause is terminated by “0”. Unlike many formats that represent

the end of a clause by a new-line symbol, this format allows clauses to be on multiple lines.

Example: $(x_1 \vee x_3 \vee -x_4) \wedge (x_4) \wedge (x_2 \vee -x_3)$

A possible input file would be

c Example CNF format file
c
p cnf 4 3
1 3 -4 0
4 0
2 -3 0

Output or solution files

Every algorithm or heuristic creates an output or solution file. This output file should consist of one or more of the following lines, depending on the type of algorithm and problem being solved.

- Comment. Comment line give human-readable information about the file and are ignored by programs. Comment lines can appear anywhere in the file. Each comment line begins with a lower-case character c. Note that comment lines can be used to provide solution information not otherwise available (i.e., computation time, number of calculations).

  c This is an example of a comment line

- Variable Line

  vV

  The lower-case character v means that this is a variable line. The

value V is either a positive value *i* , which means that i should be set to true or a negative value *-i* , implying it should be set to false.

Lab Work.

Write a program that takes two files, a SAT instance I (in .cnf format) and a solution for I. Your program should return whether the provided solution is a valid solution for I or not. A sample program, "satFileReaders.py" is on Canvas with functions for reading the two file types.

You need to run your code using the command:

python SAT <cnf instance> <solution file>

For example:

"python SAT instance.cnf solution.txt"

means the problem will use "instance.cnf" as a SAT instance and solution.txt as a solution for this instance.

Write a variant of this that returns either "satisfiable" or returns the **number of unsatisfied clauses**.

A test dataset (data.zip) is available on Canvas. ***Inst*** includes 9 sat instances uf20-*n*.cnf for different values of *n* and ***sol*** includes a set of potential solutions (one of which is a solution for each instance). Identify which is the correct solution for each instance.

Consider how we could implement a heuristic approach to generate a solution (as opposed to just random) similar to the TSP insertion heuristic. In other words, until all variables are assigned a value: select a variable, assign it a value. What criteria could we use for the variable selection?