CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Metaheuristic Optimization

Lab: Genetic Algorithms for the Satisfiability problem

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.

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

Where $\vee$ represents the ***OR*** Boolean connective, $\wedge$ represents **AND,** and $-x_i$ 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 $-x_3$ 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

Lab Work.

Write a program that reads a SAT instance in the format defined above and solves it using a genetic algorithm.

You may need to adapt the checker from last week to count the number of unsat clauses for a given candidate solution.

You should implement roulette wheel for selection.

Implement both 1-point and uniform crossover operators.

For mutation it should iterate through each gene and mutate with probability based on the mutation rate.

Initial values for the standard parameters (maximum number of iterations 500, mut rate 1/n, pop size 50).

Note that we know the optimal value for this case so our stopping condition

has two components, the optimal value of 0 or the maximum number of iterations.

Test on the instances from the SAT checker lab.