CSIS, BITS Pilani K. K. Birla Goa Campus

Artificial Intelligence (CS F407)

Programming Assignment 1
Total Marks: 15
Submission Deadline: 9 PM on 07/09/2025 (Sunday)

Each student must complete this assignment individually. Your program must be written in Python and should run without errors on Python 3.10.

Plagiarism Warning: Any form of plagiarism will result in **zero marks for everyone involved**. No distinction will be made between minor and major cases.

Late Submission Policy: The deadline is strictly **9 PM**. Late submissions will incur a penalty of 5 marks per day. Submit early to avoid issues such as power or internet failures.

Question 1 (15 marks)

The propositional logic formula shown below is in conjunctive normal form (3-CNF):

$$(a \vee \neg b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee \neg e \vee \neg d) \wedge (\neg b \vee c \vee d) \wedge (\neg c \vee \neg d \vee e).$$

A *model* is an assignment of logical values to all the variables. For example, $(a = T, b = T, c = F, d = F, e = T)$ is a model. A *clause* is a disjunction of literals (e.g., $(a \vee \neg e \vee \neg d)$). A clause is satisfied if it evaluates to *True* under a given model.

Example. The model $(a = T, b = T, c = F, d = F, e = T)$ satisfies 4 out of the 5 clauses above.

Note on CNF_Creator.py. The provided program `CNF_Creator.py` creates a propositional logic formula in 3-CNF format as a **list of lists**. Each inner list contains exactly 3 integers, where each integer represents a literal. For example:

$$[-50,\ 37,\ -23]$$

represents the clause $(\neg x_{50} \vee x_{37} \vee \neg x_{23})$. Here:

- A positive integer k denotes variable x_k.
- A negative integer $-k$ denotes $\neg x_k$.

This is the format you will see when reading the generated CNF formulas.

Task

You will be given a propositional logic formula in 3-CNF form. Your task is to find a model that maximizes the percentage of satisfied clauses in the formula (the **fitness function**). The formula will use 50 variables, so the state space contains 2^{50} possible models.

- Use a **Genetic Algorithm (GA)** to search for a model that maximizes the number of satisfied clauses.
- Define the fitness function as:

$$\text{Fitness} = \frac{\text{Number of satisfied clauses}}{\text{Total number of clauses}} \times 100.$$

Requirements

1. **Baseline GA.** Implement the standard GA algorithm from the textbook.
   - Population size = 20.
   - Initial population selected uniformly at random.
2. **Improved GA.** Design an improved variant of GA that finds the best model as quickly as possible (maximize fitness while minimizing runtime).
   - You may modify any aspect of the GA.
   - Ensure that your program terminates within **45 seconds**.
   - You may also terminate earlier if (i) the fitness value has not improved for several generations, or (ii) the fitness value reaches 100%.
3. **Data generation.** Use the provided function `CreateRandomSentence()` to generate 3-CNF sentences.
   - Number of variables: $n = 50$ (fixed).
   - Number of clauses: $m \in \{100, 120, 140, \ldots, 300\}$.

Report

Submit a short report including:

1. A graph of the average fitness value of the best model found by the improved algorithm for different values of m.
   - For each m, generate at least 10 random sentences and report the average best fitness.
   - Stop execution at 45 seconds if needed.
2. A graph of the average running time for different values of m.
   - For each m, generate at least 10 random sentences.

- Running time must be capped at 45 seconds.

3. A description of the improvements you made to the GA and approaches that failed.
4. Observations on the types of problems where GA struggles to find good solutions.
5. Insights about the difficulty of satisfying 3-CNF sentences. When do such problems become hardest to satisfy?

Instructions for Submission

- Submit exactly **two files**:
  1. Your Python program: `ROLLXYZ_FIRSTNAME.py`
  2. Your report: `ROLLXYZ_FIRSTNAME.pdf`
- Use only capital letters in filenames. Example: `2020H1030999G_ADARSH.py`, `2020H1030999G_ADARSH.pdf`.
- Your program must:
  - Implement only the improved GA version.
  - Read the 3-CNF from `CNF.csv` using the provided `ReadCNFfromCSVfile()` function.
  - Terminate within 45 seconds.
  - Produce output in the format shown in the sample image (`sample_output.png`).
- Multiple `.csv` test files are provided; check that your program works correctly on them.
- Submit files directly on Quanta. **Do not zip**.
- Report any bugs in `CNF_Creator.py` to the course IC.