CSIS, BITS Pilani K. K. Birla Goa Campus

Artificial Intelligence (CS F407)

Programming Assignment 1
Total Marks: 15
Submission Deadline: 9 PM on 29/09/2021 (Wednesday)

Each student must individually do this programming assignment. Your program must be written in Python and should run (without errors) on Python 3.6 or later.
Any form of plagiarism will result in -5 marks being awarded to everyone involved. There will be no differentiation between minor and major plagiarism.
Note that the deadline is **9 PM** and not midnight. Five marks per day will be deducted for submissions after the deadline. It will be your responsibility to submit the assignment well in advance and avoid unforeseen problems like power failures etc.

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. For example, $(a \vee \neg e \vee \neg d)$ is a clause. We say that a clause is satisfied if the clause becomes *True* under the variable assignments given by a model. The model $(a = T, b = T, c = F, d = F, e = T)$ satisfies four (out of the five) clauses in the 3-CNF sentence shown above.

In this programming assignment, you will be given a propositional logic formula in 3-CNF form. You have to find a model that maximizes the percentage of satisfied clauses in the formula (fitness function). The propositional logic formula will be built using 50 variables. So, there are 2^{50} possible models (states) in the state space.

Run the Python program ROLLNO_NAME.py and see the output. In the 3-CNF sentence, we will be using numbers instead of letter symbols. Also, we will be using the $-$ (minus) symbol instead of $\neg$ symbol. For example, $(-32 \vee -12 \vee 48)$ is a clause where symbols 32 and 12 appear as negative literals.

Use genetic algorithm to find a model (state) that will maximize the number of true (satisfied) clauses. Let, Fitness function = Percentage of clauses that are satisfied in the 3-CNF sentence.

- First, implement the version of the GA algorithm given in the textbook. Assume, that initially all the states in the population are uniformly randomly selected from the state space. For the first algorithm, let the population size be 20.
- Next, come up with a variant of the GA algorithm that is able to find the model having the maximum fitness function value as quickly as possible. In other words, you must try to improve the algorithm such that the fitness function value is maximized and the running time is minimized. To improve the algorithm, you can make changes to *any* aspect of the Genetic Algorithm.

Use the *CreateRandomSentence*() function (see ROLLNO_NAME.py file) to generate 3-CNF sentences containing different number of clauses. The number of clauses m can be varied from 100 to 300 in multiples of 20 (i.e. 100, 120, 140 etc.). The number of variables n must be fixed to 50.

Write a report that includes the following:

1. A graph that shows the average fitness function value of the best model found by the improved algorithm for different values of m. For each value of m, randomly generate multiple (at least 10) 3-CNF sentences and report the average fitness value of the best model found. If your improved algorithm takes more than 45 seconds, then you can terminate and report the best model found. This means that you will have to check the time elapsed inside your program and terminate if the time elapsed crosses 45 seconds[1]. You can also terminate your algorithm if the best fitness function value does not change over several generations or the fitness value of 100% is reached.
2. A graph that shows the average running time of the improved GA algorithm for different values of m. For each value of m, randomly generate multiple (at least 10) 3-CNF sentences and report the average running time. As mentioned above, the running time will be at most 45 seconds.
3. How did you improve the GA algorithm? Which approaches failed to give the expected results?
4. From the above graphs, what can you tell about the problems where the GA algorithm might find it difficult to find a good solution.
5. What does the above graphs tell you about the difficulty of satisfying a 3-CNF sentence in n variables. When does a 3-CNF sentence become difficult to satisfy? (Note: A CNF sentence is satisfied when all its clauses are satisfied.)

Instructions for submission

- You must submit a single program file with the name "ROLLXYZ_FIRSTNAME.py". Your program needs to include only the improved version of the GA algorithm. The program that you submit must read the 3-CNF from the CNF.csv file using the *ReadCNFfromCSVfile*() function (see ROLLNO_NAME.py file). As mentioned earlier, the program must terminate within 45 seconds. The output of the program that you submit should be as shown below:

```
Roll No : 2020H1030999G
Number of clauses in CSV file :  100
Best model :  [1, -2, 3, -4, -5, -6, 7, 8, 9, 10, 11, 12, -13, -14, -15, -16, -17, 18, 19, -20, 21, -22, 23, -24, 25, 26,
-27, -28, 29, -30, -31, 32, 33, 34, -35, 36, -37, 38, -39, -40, 41, 42, 43, -44, -45, -46, -47, -48, -49, -50]
Fitness value of best model : 99%
Time taken : 5.23 seconds
```

- The assignment includes multiple csv files. You can check how your program runs for the sentences given in these csv files.

[1]This can be easily done in Python using very few lines of code.

- Your report must be named "ROLLXYZ_FIRSTNAME.pdf".
- Please use only capital letters for both the file names. Eg. 2020H1030999G_ADARSH.py and 2020H1030999G_ADARSH.pdf.
- Submit **only** the two files mentioned above. **Don't** zip the files. The assignment submission will be through quanta.
- If you notice any bugs in CNF_Creator.py program, inform the course IC.