LAB ASSIGNMENT 3 Genetic Algorithm & Constraint Satisfaction Problem

Course: CSC-462 Artificial Intelligence
CLO: CLO-6 Implement search algorithms, CSPs, and optimization methods
Total Marks: 25

PART A Genetic Algorithm (GA) *(12 Marks)*

Problem A1 Max-One GA (Coding Required) *(6 Marks)*

Write a Python program that implements a Genetic Algorithm to solve the Max-One problem for a chromosome length of L = 16 bits.

Your program must:

1. **Initialize a population of size 40 with random 0/1 chromosomes.**
2. **Evaluate fitness as the number of 1s in the chromosome.**
3. **Perform roulette-wheel selection to choose parents.**
4. **Apply single-point crossover using crossover probability Pc = 0.7.**
5. **Apply bit-flip mutation using mutation probability Pm = 0.05.**
6. **Stop when a chromosome of all 1s is produced or 500 generations are completed.**

Output Requirements:

- **Print the best chromosome and its fitness for each generation.**
- **Print the generation number at which the optimal solution is found (if found).**
- **If not found, print *"Optimal solution not found"*.**

Problem A2 GA Behaviour Analysis *(6 Marks)*

After writing the code in A1, answer the following questions based on your program's output:

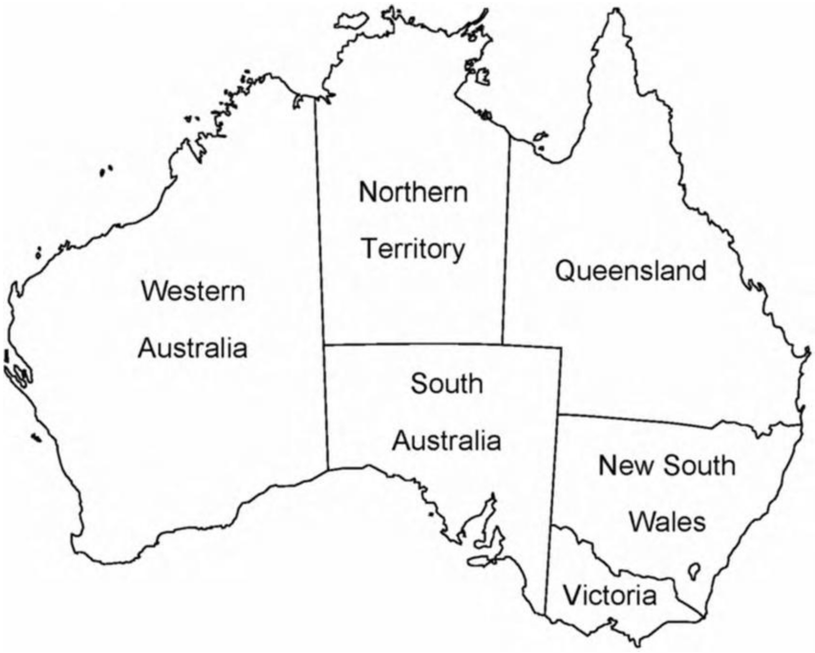
a. How many generations were required to reach the optimal chromosome?
b. What happens when you increase mutation probability Pm from 0.05 → 0.20?
c. Why are Genetic Algorithms less likely to get stuck in local maxima compared to

PART B Constraint Satisfaction Problem (CSP) ***(13 Marks)***

Problem B1 Australia Map Coloring CSP ***(5 Marks)***

Write a Python program that solves the Australia Map Coloring Problem using Backtracking Search.

Use the seven regions shown in Fig. 35 of the lab manual:
WA, NT, SA, Q, NSW, V, T



Requirements:

1. **Variable domains: {Red, Green, Blue}**
2. **Constraint: Adjacent regions must not have the same color.**
3. **Implement a backtracking solver that checks consistency at each step.**

Output:

- **Print the final color assigned to each region.**
- **Print the total number of backtracking steps performed.**

Problem B2 CSP with Forward Checking ***(5 Marks)***

Extend your solution from B1 by adding Forward Checking.

Your improved program must:

- **Reduce domain values of neighboring variables after assigning a value.**

- **Stop early if any region's domain becomes empty.**

Output:

- **Print the solved assignment.**
- **Compare the number of backtracking steps with the solution from B1 (Which one required fewer steps? Why?)**

Problem B3 4-Queens as a CSP *(3 Marks)*

Write a Python program to solve the 4-Queens problem using Backtracking + Forward Checking.

Model the problem as:

- **Variables: Q1, Q2, Q3, Q4 (each representing a row)**
- **Domains: {1, 2, 3, 4} (column positions)**
- **Constraints:**
  - **No two queens share the same column**
  - **No two queens lie on the same diagonal**

Output:

- **Print one valid solution.**
- **Print number of backtracking steps.**

<u>Submission Instructions</u>

Students must submit:

1. **Python files for each task.**
2. **A short PDF with explanations & screenshots of outputs.**
3. **All code must be original and well-commented.**