

CSE422: Artificial Intelligence [C02]

Lab Assignment 2

Part 1 [7 points]

Brac University plans to optimize its course scheduling for the upcoming academic semester. The university offers a variety of courses across different disciplines, each with specific scheduling requirements and constraints. The university needs to find a way to schedule its courses into a limited number of timeslots per day while ensuring that each course is scheduled **exactly once** and **no timeslot has more than one course planned at the same time**.

You are tasked with optimizing the schedule for courses offered at Brac University using the popular *Genetic Algorithm*.

Chromosome Representation (Encoding):

Each chromosome will be a binary string that encodes the schedule of courses across different timeslots. Here's how we will represent a chromosome:

- **Length of the Chromosome:** The length of a chromosome will be equal to $N \times T$ (product of N and T), where N is the number of courses and T is the number of timeslots.
- **Structure of the Chromosome:** Each chromosome will be divided into T segments, where each segment will be of length N . Each segment will represent a timeslot, and each bit within a segment will represent whether a particular course is scheduled in that timeslot.

Fitness Calculation:

- The fitness function will evaluate each solution based on the number of course overlaps and consistency of a course.
- The fitness function evaluates the quality of a schedule based on minimizing course overlaps and making sure a course is scheduled exactly once:

$$Fitness(S) = - [Penalty_{overlap}(S) + Penalty_{consistency}(S)]$$

Here:

- *S*: Binary String representing schedule
- $Penalty_{overlap}(S)$: \sum
(Number of courses overlap in schedule *S* in the same timeslot).
- $Penalty_{consistency}(S)$:
 \sum (Number of times courses appeared more than once in schedule *S*)

Overlap Penalty:

- For each timeslot, count the number of courses scheduled.
- If more than one course is scheduled in the same timeslot, add a penalty ***equal to the number of extra courses***.

Consistency Penalty:

- For each course, check if it is scheduled exactly once.
- If a course is ***not scheduled exactly once, add a penalty***.

Task Breakdown:

1. Model the course schedule array in a way suitable for the problem.
2. Implement the fitness function that penalizes overlapping courses and ensures each course is scheduled exactly once.
3. Choose two parents based on **random selection** for crossover. **Show it as a separate function.**
4. Perform **single-point crossover** to create **2 offspring** from each pair of selected parents. **Show it as a separate function.**
5. Write the **mutation function** to introduce random changes.
6. Create a population of randomly generated course schedules.
7. Run genetic algorithms on the population until the highest fitness has been reached and/or the number of maximum iterations has been reached.

Input

The first line has a number N denoting the number of courses and a number T denoting the number of timeslots for a particular day. It will be followed by N lines each having a string that represents a course code that needs to be scheduled where,

$$T \geq N$$

[In this problem statement, we are considering that 1 course will have only 1 section]

Output

The output should be a binary string denoting 1 for scheduled courses and 0 for not scheduled courses in each timeslot. A string consisting of all zeros won't be accepted. You also need to print the fitness value of the output string.

Example:

Sample Input
3 3 CSE110 MAT110 PHY112
Sample Output
110110010 -6
Explanation
<p><u>Chromosome Representation</u></p> <ul style="list-style-type: none">• $N \times T = 3 \times 3 = 9$• A chromosome of length 9 represents the schedule of courses across 3 timeslots.• Each timeslot is represented by a segment of length $N=3$. <p><u>Fitness Calculation</u></p> <p>Let's take the output chromosome: 110110010</p> <ul style="list-style-type: none">• Timeslot 1: 110<ul style="list-style-type: none">○ CSE110: 1 (scheduled)○ MAT110: 1 (scheduled)○ PHY112: 0 (not scheduled)• Timeslot 2: 110<ul style="list-style-type: none">○ CSE110: 1 (scheduled)○ MAT110: 1 (scheduled)○ PHY112: 0 (not scheduled)• Timeslot 3: 010<ul style="list-style-type: none">○ CSE110: 0 (not scheduled)○ MAT110: 1 (scheduled)○ PHY112: 0 (not scheduled)

Interpretation of the Chromosome

1. *Timeslot 1: CSE110, MAT110 are scheduled.*
2. *Timeslot 2: CSE110, MAT110 are scheduled.*
3. *Timeslot 3: MAT110 is scheduled.*

Penalty Calculation

Overlap Penalty:

- *Timeslot 1: 2 courses scheduled, penalty = $2-1=1$*
- *Timeslot 2: 2 courses scheduled, penalty = $2-1=1$*
- *Timeslot 3: 1 course scheduled, penalty = $1-1=0$*
- ***Total overlap penalty = $1+1+0=2$***

Consistency Penalty:

- *CSE110: scheduled 2 times, penalty = $|2-1|=1$*
- *MAT110: scheduled 3 times, penalty = $|3-1|=2$*
- *PHY112: scheduled 0 times, penalty = $|0-1|=1$*
- ***Total consistency penalty = $1+2+1=4$***

Total penalty = overlap penalty + consistency penalty = $2+4=6$

Summary

- *Chromosome 110110010 results in a penalty of 6. So Fitness will be -6*

Part 2 [3 points]

For this part randomly select two parents from the initial population of your problem statement. Then perform a **two-point crossover** to generate two children. The two points have to be chosen **randomly**, but it has to be made sure the second point always comes after the first point.

Here is an example of how **two-point crossover** works:

Parent 1: 000111000

Parent 2: 111000111

For two points crossover, we have randomly chosen the following points:

1st point:- between index 2 and index 3

2nd point:- between index 6 and index 7

So the two resultant offsprings are, 000000100 & 111111011

[In this part, you just need to iterate once and print the resultant offspring after doing the crossover]

Part 3 [0 points]

In part 1, you selected parents through random sampling from the initial population. Another advanced technique for parent selection is known as **Tournament Selection**. Please take some time to research and understand this method at home. Might be helpful in the near future!