REPORT

Q2 Time Table Scheduling

Variables (M, N, P): (#courses, #rooms, #profs)

I. Assumptions:

1. N < M (which is a legit assumption for a university or a college like IIITD)
2. Every course will have only 2 lectures scheduled in a week
3. It will be pre-known to us (hard-coded) that which prof. takes which course
4. A prof. can take a maximum of 2 courses only

II. Logical Constraints:

1. A professor can take a maximum of 2 courses
2. For each course, two classes per week is mandatory
3. For each course, a maximum of 1 class per day can be taken
4. A professor taking more than 1 course should not be teaching two different courses at the same time
5. These should not be more than one lecture scheduled in a room at the same time

III. Methodology:

1. Alphabet: takes values from 1 to M, Alphabet size = M
2. Structure of a chromosome:
   - Matrix with dimensions (R, C, N) = (#days, #time_slots, #rooms): corresponds to a structure of a time table
   - Gene: Each cell having value equal to course_id of the course for which the lecture is scheduled for a particular day, particular time slot and particular room
   - An empty cell (value = 0) represents vacant room for that slot for that day
   - Room 1 corresponds to n(value in 3rd dimension) = 0, Room 2: n=1 ... Room N: n=N-1
3. Fitness Function:
   Reference paper: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5705788&tag=1
   - A chromosome is penalised in the following cases:
     - When lecture for a subject is scheduled more than once in a day, then penalise with the count with which it exceeds
     - When lecture for a subject is scheduled more than twice in a week, then penalise with the count with which it exceeds (this may occur during MA)
     - When a lecture taught by same professor is scheduled in same time slot.

   - All such penalty is accumulated in a variable (say c) and the following formula is applied: fitness = (1/(1+c)). Desired time table (chromosome) is the one which has c=0, hence fitness = 1
4. Genetic Algorithm:
   - Initial Population has only 2*M slots filled, rest 0. #Occurences of each cours = 2
   - Crossover: A point is randomly selected in 2nd half of 1st dimension (days) and the crossover is conducted for both the chromosomes. It is taken care that this does not result in uneven no. of occurrences of each course.
   - Mutation: Values for 10 randomly selected cells are pairwise swapped
   - Selection Operator: Chromosome with maximum value is chosen while selection
   - Stopping Criterion: When a chromosome with fitness = 1 appears in the population

5. Memetic Algorithm:

- Local search: swapping the value of current cell with 4 cells whose indices are generated randomly to generate 4 different neighbours from current chromosome, then selection of chromosome with best fitness value
- Recombination is done using crossover as described above

IV. Following is the **comparison between MA and GA** for same initial population, along with inferences and results (time table, graphs)

<u>GENETIC ALGORITHM A: CONVERGENCE AFTER 8 ITERATIONS</u>:
(M, N, P) = (20, 2, 10)
Note: Numbers in bold denote (#rooms remained empty each day, #completely vacant slots)

```
DAY : 1
[0. 2.] [13.  1.] [6. 3.] [4. 0.] [7. 0.] [ 0. 17.] [14. 11.] [0. 0.]          (6, 1)
DAY : 2
[0. 8.] [ 0. 10.] [0. 0.] [17.  4.] [18.  0.] [0. 0.] [0. 0.] [0. 0.] (11, 4)
DAY : 3
[ 0. 13.] [0. 1.] [ 0. 12.] [ 0. 10.] [20.  5.] [8. 0.] [3. 7.] [0. 0.]        (7, 1)
DAY : 4
[15.  0.] [0. 0.] [9. 0.] [19.  5.] [ 0. 12.] [16. 20.] [0. 0.] [0. 0.]        (9, 3)
DAY : 5
[16. 18.] [15. 19.] [0. 0.] [0. 6.] [ 0. 14.] [9. 2.] [11.  0.] [0. 0.]        (7, 2)
```

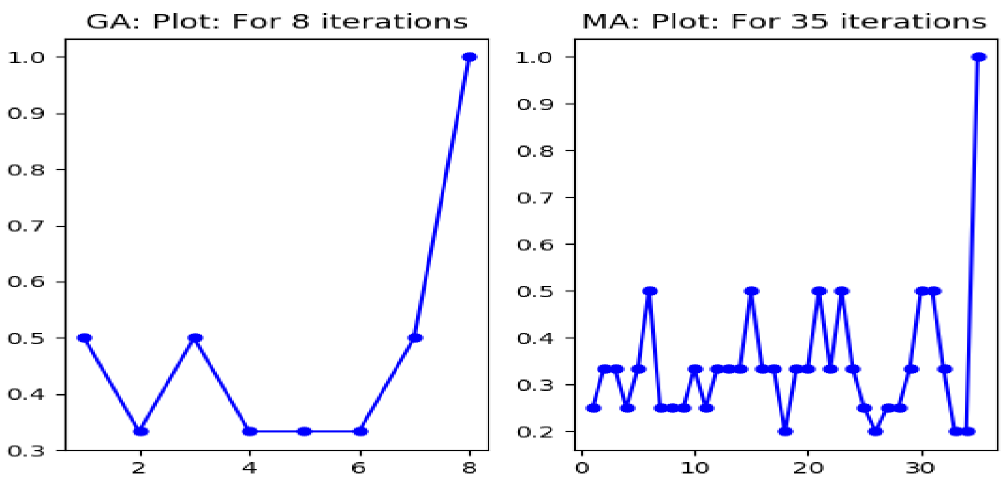
Avg no. of rooms left empty each day = 40/5 = 8
Corresponding Std Dev. = 2

Avg. no. of completely vacant slots = 11/5 = 2.2
Corresponding Std. Dev = 1.3

<u>MEMETIC ALGORITHM A: CONVERGENCE AFTER 35 ITERATIONS</u>:
(M, N, P) = (20, 2, 10)

```
DAY : 1
[0. 1.] [12.  0.] [0. 0.] [8. 4.] [0. 0.] [7. 0.] [19. 11.] [ 0. 14.]                  (8, 2)
DAY : 2
[0. 1.] [0. 0.] [8. 0.] [19.  4.] [0. 0.] [3. 0.] [ 0. 11.] [ 0. 14.]          (9, 2)
DAY : 3
[2. 0.] [10.  7.] [20.  0.] [ 0. 17.] [0. 0.] [0. 0.] [0. 0.] [ 0. 12.]                (10, 3)
DAY : 4
[ 0. 13.] [0. 6.] [ 0. 18.] [3. 0.] [ 0. 16.] [20. 10.] [0. 0.] [15.  9.]      (7, 1)
DAY : 5
[0. 6.] [ 0. 18.] [0. 0.] [15. 17.] [0. 2.] [ 0. 13.] [ 0. 16.] [0. 9.]                (8, 1)
```

Avg no. of rooms left empty each day = 42/5 = 8.4
Corresponding Std. Dev = 1.14
Avg. no. of completely vacant slots = 9/5 = 1.8
Corresponding Std. Dev = 0.8



x axis: generation, y axis: best fitness value

INFERENCE: The values of standard deviation show that:
-> deviation is 2 from mean for GA, while it is ~1 for MA
-> thus I can conclude that MA results in a better time table because the number of rooms left empty for each day is more uniform for MA (because of less std dev). Similarly for no. of slots completely left empty
-> this means GA is resulting in **premature convergence** i.e. to achieve convergence it is giving a suboptimal solution, while MA resulted in better solution
-> More the number of iterations for MA, better are the results
-> More intuitively comparable results are obtained on larger values of M (#courses) and P (#profs) and smaller values of N (#rooms) (like M,N,P = 20,2,10 or 25,3,20) for values lesser than these, sparse time tables for MA and GA are not comparable
-> for larger values of M and P, no. of slots left completely empty seemed to be a better metric to measure the uniformity of lectures in the time table generated by GA and MA.

Below are few more cases for the same:

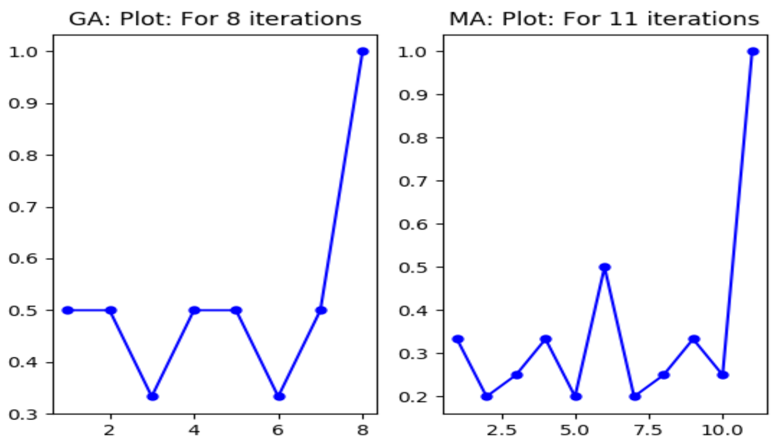
<u>GENETIC ALGORITHM B: CONVERGENCE AFTER 8 ITERATIONS</u>:
(M, N, P) = (20, 2, 10)

```
DAY : 1
[[18.  0.] [ 3.  0.] [20.  0.] [ 0.  0.] [11.  0.] [ 0. 10.] [ 0. 17.] [ 5.  0.]]
DAY : 2
[[ 0.  0.] [ 8.  0.] [ 0.  0.] [ 0.  0.] [ 0.  6.] [14.  1.] [12.  0.] [ 0.  4.]
DAY : 3
[[13.  1.] [ 9.  4.] [ 8.  2.] [ 0.  0.] [ 7. 17.] [16.  6.] [ 0.  0.] [14.  0.]]
DAY : 4
[[16.  0.] [ 0.  7.] [ 0.  2.] [19.  3.] [ 0.  0.] [ 0.  0.] [11. 15.] [ 0.  0.]]
DAY : 5
[[10. 13.] [15.  0.] [ 0.  0.] [18.  5.] [20.  0.] [ 0. 12.] [19.  9.] [ 0.  0.]]
```

<u>MEMETIC ALGORITHM B: CONVERGENCE AFTER 11 ITERATIONS</u>:
(M, N, P) = (20, 2, 10)

```
DAY : 1
[[ 5.  0.] [18. 14.] [ 0. 12.] [ 0.  8.] [ 0. 13.] [ 9.  0.] [ 0. 16.] [ 2.  7.]]
DAY : 2
[[19. 15.] [ 3.  0.] [ 5.  7.] [ 0.  8.] [ 0. 11.] [20.  0.] [ 0.  0.] [ 9.  0.]]
```

```
DAY : 3
[[11.  0.] [10.  0.] [ 0.  4.] [ 0.  0.] [15.  0.] [ 0. 13.] [ 0. 16.] [17.  3.]]
DAY : 4
[[ 4. 20.] [17.  0.] [ 0.  0.] [ 0.  1.] [ 0.  0.] [ 0. 10.] [ 0.  0.] [19.  6.]]
DAY : 5
[[ 0.  1.] [ 0.  0.] [ 0. 14.] [ 0.  0.] [ 0.  6.] [ 0.  0.] [ 0.  2.] [18. 12.]]
```



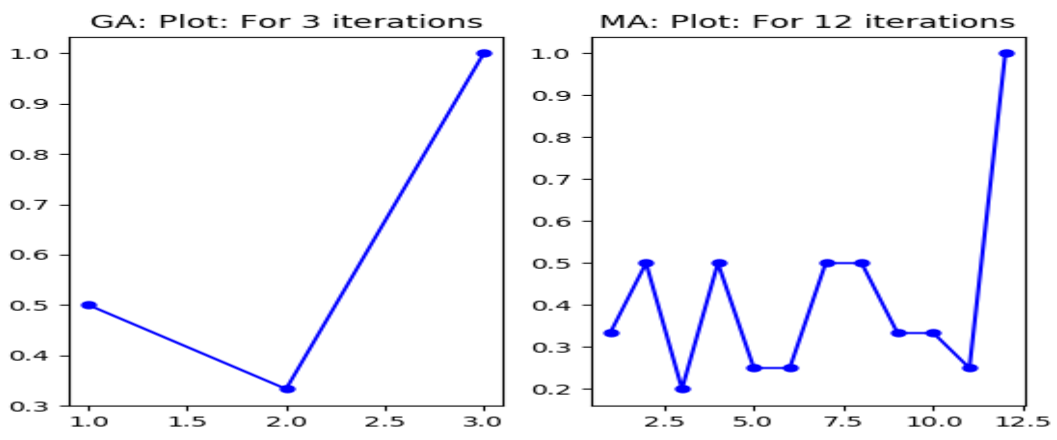
x axis: generation, y axis: best fitness value

<u>GENETIC ALGORITHM C: CONVERGENCE AFTER 3 ITERATIONS</u>:
(M, N, P) = (20, 2, 10)

```
DAY : 1
[0. 3.] [0. 0.] [0. 0.] [0. 4.] [5. 2.] [ 1. 12.] [0. 0.] [6. 0.]
DAY : 2
[0. 0.] [11.  0.] [ 0. 10.] [0. 0.] [0. 8.] [ 9. 13.] [0. 0.] [0. 0.]
DAY : 3
[16.  8.] [17.  0.] [0. 0.] [4. 1.] [14.  0.] [13.  0.] [15.  7.] [ 0. 20.]
DAY : 4
[0. 0.] [ 0. 12.] [0. 7.] [ 0. 20.] [19. 15.] [ 3. 14.] [0. 9.] [ 0. 18.]
DAY : 5
[0. 6.] [ 5. 10.] [0. 0.] [ 0. 11.] [16. 18.] [17.  0.] [2. 0.] [ 0. 19.]
```

<u>MEMETIC ALGORITHM C: CONVERGENCE AFTER 12 ITERATIONS</u>:
(M, N, P) = (20, 2, 10)

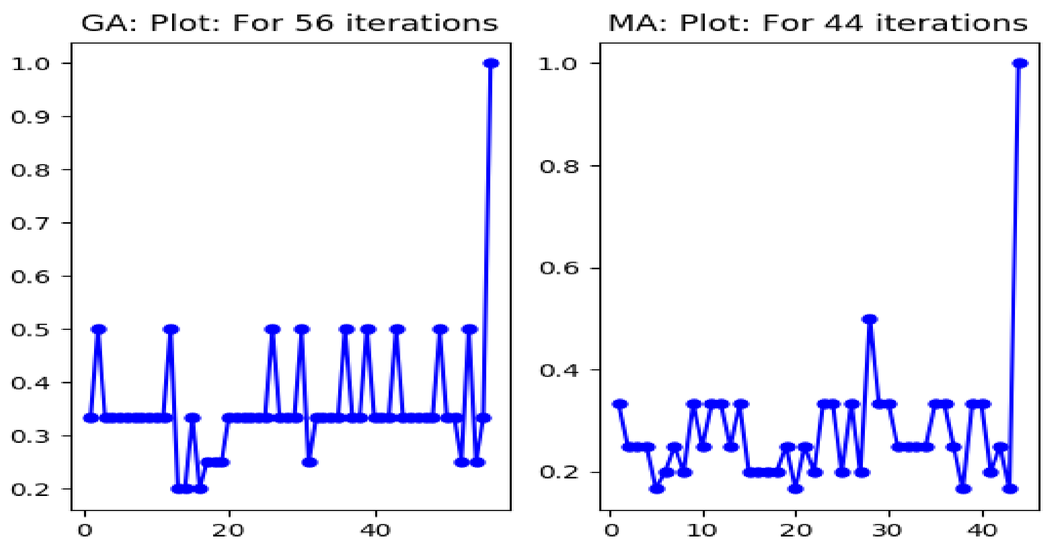
```
DAY : 1
[ 0. 20.] [2. 0.] [0. 0.] [15.  0.] [8. 9.] [13.  4.] [18. 11.] [5. 0.]
DAY : 2
[0. 3.] [ 9. 19.] [5. 0.] [15.  0.] [1. 0.] [0. 0.] [0. 0.] [0. 0.]
DAY : 3
[13.  0.] [0. 0.] [16.  6.] [ 0. 18.] [ 0. 17.] [0. 2.] [ 8. 11.] [12.  0.]
DAY : 4
[0. 0.] [14.  0.] [1. 6.] [ 0. 20.] [4. 7.] [0. 0.] [19. 10.] [0. 0.]
DAY : 5
[ 3. 17.] [ 0. 16.] [0. 0.] [ 7. 10.] [12.  0.] [0. 0.] [14.  0.] [0. 0.]
```



x axis: generation, y axis: best fitness value

GENETIC ALGORITHM D: CONVERGENCE AFTER 46 ITERATIONS:
Note: Numbers in bold denote no. of slots remained empty each day
(M, N, P) = (25, 3, 20)
DAY : 1
[17. 0. 0.] [18. 0. 0.] [0. 2. 14.] [0. 10. 3.] [19. 4. 0.] [0. 0. 0.] [23. 0. 22.] [0. 0. 0.] **(2)**
DAY : 2
[0. 0. 0.] [22. 0. 8.] [0. 0. 0.] [0. 0. 7.] [0. 0. 0.] [0. 1. 25.] [0. 20. 11.] [14. 0. 0.] **(3)**
DAY : 3
[21. 0. 0.] [1. 5. 0.] [0. 15. 0.] [0. 0. 0.] [19. 0. 0.] [23. 0. 0.] [8. 0. 10.] [7. 0. 0.] **(1)**
DAY : 4
[0. 11. 0.] [6. 0. 0.] [20. 0. 16.] [24. 0. 0.] [0. 0. 0.] [0. 18. 25.] [0. 12. 17.] [0. 9. 13.] **(1)**
DAY : 5
[15. 0. 5.] [0. 0. 6.] [4. 0. 24.] [2. 0. 0.] [3. 16. 0.] [0. 0. 9.] [0. 21. 0.] [12. 0. 13.] **(0)**
Avg. = 7/5 = 1.4
Std. Dev. = 1.14

MEMETIC ALGORITHM D: CONVERGENCE AFTER 54 ITERATIONS:
(M, N, P) = (25, 3, 20)
DAY : 1
[0. 15. 0.] [0. 0. 3.] [0. 18. 0.] [20. 0. 0.] [24. 5. 0.] [0. 0. 0.] [0. 0. 2.] [0. 0. 10.] **(1)**
DAY : 2
[0. 0. 13.] [3. 0. 22.] [0. 18. 0.] [20. 7. 0.] [17. 0. 0.] [0. 0. 0.] [0. 0. 0.] [0. 23. 9.] **(2)**
DAY : 3
[8. 0. 11.] [16. 7. 0.] [13. 23. 0.] [25. 0. 0.] [17. 0. 5.] [9. 0. 0.] [4. 0. 12.] [21. 0. 0.] **(0)**
DAY : 4
[21. 15. 4.] [0. 0. 24.] [0. 0. 19.] [0. 0. 0.] [1. 14. 0.] [12. 0. 0.] [0. 0. 2.] [0. 6. 0.] **(1)**
DAY : 5
[14. 11. 0.] [19. 0. 22.] [6. 0. 0.] [0. 0. 0.] [0. 8. 1.] [10. 0. 0.] [0. 0. 16.] [0. 25. 0.] **(1)**
Avg. = 1
Std. Dev. = 0.7



x axis: generation, y axis: best fitness value

<u>CONSTRAINTS SATISFACTION PROBLEM</u>:

I. CONSTRAINTS:

1. A professor should not be teaching in two different rooms at the same time
2. For a day, each course should have only one lecture
3. For a week, each course should have only two lectures

II. ASSUMPTIONS:

1. A prof can take a maximum of 2 lectures per day
2. There is no restriction on the time gap between the two lectures taken by a prof.
3. A course must have maximum 2 lectures per week and can have maximum of 1 lecture per day.

III. METHODOLOGY:

1. Forward checking algorithm is employed in the algorithm
2. Instead of maintaining a list of choices, I am maintaining a list of restricted courses that can not be filled in a particular slot given the constraints mentioned above.

III. OBSERVATIONS:

1. Since the first room found empty in the time table is assigned a course lecture given none of the above constraints are violated, the initial days are completely packed while the later days are left unfilled.
2. No uniformity is encompassed in leaving the rooms/ time slots empty throughout the time table.

IV. RESULT:

(M, N, P) = (20, 2, 12)

```
DAY : 1
[1. 3.] [2. 4.] [5. 7.] [6. 8.] [ 9. 11.] [10. 12.] [13. 15.] [14. 16.]
DAY : 2
[1. 3.] [2. 4.] [5. 7.] [6. 8.] [ 9. 11.] [10. 12.] [13. 15.] [14. 16.]
DAY : 3
[17. 18.] [19. 20.] [0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.]
```

```
DAY : 4
[17. 18.] [19. 20.] [0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.]
DAY : 5
[0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.] [0. 0.]
```

V. INFERENCES:

1. Although CSP using forward checking gives a legit time table satisfying all the provided constraints, it is not as good as MA and GA for this kind of problem statement where randomisation and uniformity in terms of empty slots is desired.
2. Modifications in CSP can be made to include a degree of randomisation for uniform distribution of lectures in the time table.

References:

1. **https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5705788&tag=1**

2. ***Fig. 5.3 Pg 166, Ch-5 Adversarial Search,***
Book: Artificial Intelligence - A Modern Approach (3rd Edition) pdf