ASSIGNMENT 2 REPORT

QUESTION 1.

**ASSUMPTIONS:**

* The user that plays is always the “X” notation and the computer playes the “O” notation
* The user is the maximiser player and the computer is the minimizer player

**METHODOLOGY:**

* The methodology used in the problem is the game playing between two players

**ALGORITHM –MIN-MAX/ALPHA BETA PRUNING:**

* The user first playes a move which is the maximing player
* Based on the move played my the maximiser the next move by the minimizer is determined which should be optimal move by the minimizer . It is detected based on min max /alpha beta. The minimizer generates all possible moves and for all possible move it runs the min max algo and gets the uthility value. The move which gives the min utility value is the optimal move at that state.
* Similarly for determining the optimal move of the maximiser we take the move having the max utility value. Here we are not considering the part and maximiser is the user itself.
* In case of alpha beta pruning at each move we are storing alpha and beta values. In case of maximiser the alpha value is updated ie max of the utility value and existing alpha value. In case of minimizer the beta value is updated i.e min of the utility value and existing beta value. If the move have beta value less than alpha value we are pruning the move.

**OBSERVATION:**

* The number of node generated for detecting the next move in case of alpha beta pruning is almost he half the number of nodes generated in min max.
* The alpha beta pruning algorithm is faster than min max algorithm.

**RESULTS:**

Enter the pos of X:

0 0

current state is:

[['X', '\_', '\_'], ['\_', '\_', '\_'], ['\_', '\_', '\_']]

Number of node: 64672

next move

[['X', '\_', '\_'], ['\_', 'O', '\_'], ['\_', '\_', '\_']]

Enter the pos of X:

1 0

current state is:

[['X', '\_', '\_'], ['X', 'O', '\_'], ['\_', '\_', '\_']]

next move

[['X', '\_', '\_'], ['X', 'O', '\_'], ['O', '\_', '\_']]

Enter the pos of X:

0 2

current state is:

[['X', '\_', 'X'], ['X', 'O', '\_'], ['O', '\_', '\_']]

next move

[['X', 'O', 'X'], ['X', 'O', '\_'], ['O', '\_', '\_']]

Enter the pos of X:

2 2

current state is:

[['X', 'O', 'X'], ['X', 'O', '\_'], ['O', '\_', 'X']]

next move

[['X', 'O', 'X'], ['X', 'O', 'O'], ['O', '\_', 'X']]

Enter the pos of X:

2 1

current state is:

[['X', 'O', 'X'], ['X', 'O', 'O'], ['O', 'X', 'X']]

Tie

QUESTION 2.

**ASSUMPTION:**

**BASIC Assumption:**

* Number of course multiplied with number of lectures per course should be less than total halls multilplied with slot per day multiplied with number of days
* Number of professors should be less than number of course
* Course should have a single professor but a professor can teach multiple courses

**GENES:** It is represented as a tuple <CourseId,ProfessorId,HallNumber,Slot,Day>

**CHROMOSOMES:** Number of courses multiplied with the number of Lectures per course times the genes represent one chromosome

**POPULATION:** 200 such chromosomes combined to form a population

**CONSTRAINT:** The constraint for the problem have been divided into two sub categories Hard constraint an soft constraint.

Hard Constraint

* No two course can have the same slot number , lecture hall,time
* Same professor cannot have the same slot number and day more than once

Soft Constraint

* Same course cannot be taught more than once in a day
* Same professor cannot teach more than one slot per day

**METHODOLOGY:**

**CROSSOVER : (SINGLE POINT)**

* Take an random point from the chromosome
* Perform single point crossover

**FITNESS FUNCTION:**

* For hard constraint assigning a 50 penalty
* For soft constraint assigning a 10 penalty
* For each chromosome calculate the fitness value based on the constraint
* Min fitness value will be the best chosen chromosome

**PARENT SELECTION:**

* Based on the fitness select the top 14 chromosomes

**NEXT GENERATION:**

* Perform the parent selection method
* For each parent perform the crossover with all other parents among the top 14 selected parents and generate the new children

**LOCAL OPTIMIZATION:**

* For each chromosome select a random gene
* Create two other chromoses having the selected genes day value and slot value increamented by 1 and other having values decremented by 1
* Among all the 3 chromosomes select the best chromosome having minimum fitness value
* This is continued for all the best parent selected to find the best of the best.

**ALGORITHM(GENETIC)**

* Create a population having chromosome and gene specified
* Calculate the fitness value for the population
* Perform parent selection
* Create Next generation using crossover
* Repeat the seteps for 200 generations until a best solution if found

**ALGORITHM(MEMETIC)**

* Create a population having chromosome and gene specified
* Calculate the fitness value for the population
* Perform parent selection
* Perform local optimization on the selected parents and select the best parents
* Create Next generation using crossover
* Repeat the seteps for 200 generations until a best solution if found

**ALGORITHM (CONSTRAINT SATISFACTION PROBLEM)**

* Courses and professors are mapped initially
* Select a course
* For all possible values of hall,slot and day
* Select a hall,slot and day and assign to the course
* Check whether the all the constrained are satisfied or not
* If yes repeat the above points for the selecting the next course
* If constrained are violated repeat from point 4
* If all possible hall,slot and day values are giving violations backtrack to the previously selected course and select a different hall,slot and day and perform the steps 4
* This will countinue until all the courses and lection hours per course have been selected

**RESULTS(MEMETIC):**

enter the number 0f cources:

10

enter the number of lecture halls:

5

enter the number of professors:

4

enter the min number of slots for a course in a week:

2

enter the max number of hours for a proffesor in a week:

6

SAMPLE CHROMOSOME:

[['c0', 'p4', 2, 6, 1], ['c0', 'p4', 5, 2, 1], ['c1', 'p3', 2, 8, 4], ['c1', 'p3', 3, 3, 1], ['c2', 'p2', 4, 3, 4], ['c2', 'p2', 3, 5, 4], ['c3', 'p1', 4, 3, 2], ['c3', 'p1', 2, 7, 5], ['c4', 'p4', 3, 1, 4], ['c4', 'p4', 2, 2, 3], ['c5', 'p3', 5, 8, 1], ['c5', 'p3', 4, 7, 4], ['c6', 'p2', 2, 4, 5], ['c6', 'p2', 1, 1, 1], ['c7', 'p1', 1, 3, 1], ['c7', 'p1', 5, 1, 3], ['c8', 'p4', 4, 1, 1], ['c8', 'p4', 5, 7, 5], ['c9', 'p3', 5, 5, 2], ['c9', 'p3', 2, 2, 1]]

FITNESS VALUE : 100.0

No of iterations: 0

No of iterations: 1

No of iterations: 2

No of iterations: 3

No of iterations: 4

==========================================================

Best schedule :

([['c8', 'p4', 1, 5, 4], ['c4', 'p4', 1, 7, 3], ['c5', 'p3', 3, 7, 1], ['c5', 'p3', 4, 5, 4], ['c6', 'p2', 5, 3, 4], ['c6', 'p2', 2, 7, 2], ['c7', 'p1', 4, 3, 4], ['c7', 'p1', 4, 1, 2], ['c8', 'p4', 3, 3, 5], ['c8', 'p4', 3, 2, 1], ['c9', 'p3', 2, 7, 5], ['c9', 'p3', 3, 6, 3], ['c0', 'p4', 3, 8, 5], ['c0', 'p4', 4, 4, 2], ['c1', 'p3', 2, 2, 5], ['c1', 'p3', 5, 5, 2], ['c2', 'p2', 1, 1, 3], ['c2', 'p2', 3, 6, 1], ['c3', 'p1', 4, 4, 1], ['c3', 'p1', 4, 4, 3]], 0.0)

2

**RESULTS(MEMETIC):**

Number of generations: 200

Best fitness value: 20.0

OPTIMAL SCHEDEULE

[['c2', 'p2', 4, 2, 1], ['c2', 'p2', 1, 8, 5], ['c3', 'p1', 3, 5, 2], ['c9', 'p3', 4, 7, 5], ['c9', 'p3', 4, 1, 5], ['c0', 'p4', 2, 1, 2], ['c0', 'p4', 1, 6, 5], ['c1', 'p3', 5, 2, 4], ['c1', 'p3', 5, 3, 3], ['c2', 'p2', 1, 2, 5], ['c2', 'p2', 3, 5, 3], ['c3', 'p1', 5, 5, 5], ['c3', 'p1', 5, 8, 1], ['c4', 'p4', 3, 4, 1], ['c4', 'p4', 2, 4, 3], ['c5', 'p3', 1, 2, 1], ['c5', 'p3', 2, 8, 5], ['c2', 'p2', 3, 3, 4], ['c3', 'p1', 4, 6, 4], ['c3', 'p1', 2, 3, 3]]

**RESULT(CSP)**

enter the number 0f cources:

10

enter the number of lecture halls:

6

enter the number of professors:

4

enter the min number of slots for a course in a week:

2

enter the max number of hours for a proffesor in a week:

6

['c1', 'p3', 5, 6, 4]

['c4', 'p4', 1, 5, 5]

['c4', 'p4', 2, 5, 2]

['c0', 'p4', 3, 1, 3]

['c6', 'p2', 3, 1, 5]

['c0', 'p4', 6, 5, 5]

['c0', 'p4', 6, 2, 3]

['c2', 'p2', 6, 4, 4]

['c7', 'p1', 1, 3, 1]

['c6', 'p2', 3, 4, 4]

['c6', 'p2', 1, 1, 3]

['c2', 'p2', 1, 6, 4]

['c7', 'p1', 3, 1, 1]

['c8', 'p4', 1, 7, 5]

['c9', 'p3', 2, 7, 4]

['c9', 'p3', 6, 5, 2]

['c5', 'p3', 5, 7, 3]

['c3', 'p1', 3, 8, 3]

['c3', 'p1', 4, 4, 4]

['c1', 'p3', 5, 8, 4]

['c8', 'p4', 3, 2, 5]

['c5', 'p3', 1, 3, 2]

Best schedule=============

[['c1', 'p3', 5, 6, 4], ['c4', 'p4', 1, 5, 5], ['c4', 'p4', 2, 5, 2], ['c0', 'p4', 3, 1, 3], ['c6', 'p2', 3, 1, 5], ['c0', 'p4', 6, 2, 3], ['c2', 'p2', 6, 4, 4], ['c7', 'p1', 1, 3, 1], ['c6', 'p2', 1, 1, 3], ['c2', 'p2', 1, 6, 4], ['c7', 'p1', 3, 1, 1], ['c8', 'p4', 1, 7, 5], ['c9', 'p3', 2, 7, 4], ['c9', 'p3', 6, 5, 2], ['c5', 'p3', 5, 7, 3], ['c3', 'p1', 3, 8, 3], ['c3', 'p1', 4, 4, 4], ['c1', 'p3', 5, 8, 4], ['c8', 'p4', 3, 2, 5], ['c5', 'p3', 1, 3, 2]]

**OBSERVATIONS AND INFERENCE:**

Memetic Algorithm sometimes converges early to the solution than genetic alogorithm as it is performing the local optimization on the best parent selection