AI Programming Assignment-2
Report

1. TicTacToe Game:

Using MiniMax:

In MiniMax algorithm, Max(User) vs Min(Computer) will play against each other.
Initially user has 9 moves,as game progress user have 7,5,3,1 moves.
Likewise Initially Computer has 8 moves , latest 6,4,2 moves.

The utility values are according to the Max like - +1 for Win
-1 for Loss and
0 for Draw

Here Computer always tries to reduce the winning probability of Max(User), at the same time tries to win .
Max will choose the path i.e., maximum utility value. Max Value=max(+1,-1,0)
Min will choose the path i.e., minimum utility value. Min Value=min(+1,-1,0)

Min will make a move accordingly
case 1: if winning move(which is equivalent to reducing Max win probability) exists then pick that move.
Case 2: if no winning move exists take random move.(may be Draw or Loss)

Game flow: (see) **minimax.txt**

Using AlphaBeta pruning:

Here Alpha(User) is equivalent to Max and Beta is equivalent to Min(Computer).

The utility values with respect to Max(User) - +1 for Win, -1 for Loss , 0 for Draw

The moves are made by User and Computer alternatively. But while Computer's turn it will generate the children by taking Alpha and Beta values into consideration and prune the edges if that move is useless or losing state.

Initially at node - Alpha value=-Infinity, Beta value=+ Infinity
The computer will pick its best move with respect to Min by calculating Alpha and Beta values and prune the useless state. So that picking the move is fast.

Pruning Condition : **ALPHA value >= BETA value**

case 1: if winning move(which is equivalent to reducing Max win probability) exists then pick that move by evaluating Alpha,Beta values.
Case 2: if no winning move exists then takes random move.(may be Draw or Loss)

Game flow: (see) **alphabeta.txt**

picking the best move by computer in MiniMax algorithm and in AlphaBetaPruning is a little different.

For Comparison on outputs: (see) minimax.txt and alphabeta.txt

2. Time table Scheduling:

Logical Constraints:

- Same course should not be allocated more than once in same slots.
- All courses should be exactly as specified frequency(fixed course frequency).
- Same course should not be scheduled more than once on same day.
- Only one course should be allocated to one room in one slot.

Using Genetic Algorithm:

Program Flow:

- Step1: randomly generating initial population.
- Step2: giving initial population to genetic algorithm.
- Step3: calculating fitness values and store them in chromosome.

  Fitness Function (F)= 1 / (1+C)

  where C= number of conflicts in that chromosome w.r.t constraints

  Desired state is when C=0 i.e., F=1
- Step4 : take weighted chromosomes from population according to fitness values and generate children using one of the cross over methods.

  Single Point CrossOver: randomly generates one value to partition the chromosome and exchange those partitions each other(parent 1 and parent 2).

  Multi Point CrossOver: (Two point CrossOver): divide the chromosome into three parts and exchange them with each other.

  Uniform CrossOver: exchange genes alternatively.

  The two parents will produce two children by using these methods.
- Step 5: Mutation takes place – randomly choose one gene and change its value.
- step6: add out newly generated children to population by deleting some parents.
- If desired state encounted then return it.

Using Memetic Algorithm: is an extended version of Genetic Algorithm.

Program Flow:

- Step1: randomly generating initial population.
- Step2: giving initial population to genetic algorithm.
- Step3: calculating fitness values and store them in chromosome

  Fitness Function (F)= 1 / (1+C)

  where C= number of conflicts in that chromosome w.r.t constraints

  Desired state is when C=0 i.e., F=1

- Step4 : take weighted chromosomes from population according to fitness values and generate children using one of the cross over methods.
  Single Point CrossOver: randomly generates one value to partition the chromosome and exchange those partitions each other(parent 1 and parent 2).
  Multi Point CrossOver: (Two point CrossOver): divide the chromosome into three parts and exchange them with each other.
  Uniform CrossOver: exchange genes alternatively.

  The two parents will produce two children by using these methods.

- Step 5: do LocalSearch using Hill climbing to generate the children which have fitness more than its parents and add them to population. So that our converge will occur faster.
- Step 6: Mutation takes place – randomly choose one gene and change its value.
- Step7: add out newly generated children to population by deleting some parents.
- If desired state encounted then return it.

Using CSP:(Backtracking search CSP)

It uses Backtracking search algorithm like DFS, Depth-Limited DFS, Iterative Deepening DLS. Here timetable scheduling is done by using backtraking search algorithm DFS.

Program Flow:

- Step1: generated one initial solution.
- Step2: called backtrackingCSP by giving input as out initial solution.
- Step3: check for desired solution, if it is true then return that state
- Step4: generate children according to specified constraints above and calculate fitness values for each generated children.

  Fitness Function (F)= 1 / (1+C)

  where C= number of conflicts in that chromosome w.r.t constraints

  Desired state is when C=0 i.e., F=1
- Step5: store them in queue and do recursively same step for each until we find goal state.

Here we are following Backtracking CSP i.e, it will take one state and go to till depth by applying our constraints. So, it will take more time relative to GA and MA because of DFS but here we will find our solution.

Sample test cases:

GA and MA: Input1:

days: 5 slots: 8 rooms: 5
for each room:totalSlots: 40
for all rooms: total slots: 200
--> length of chromosome: 200
Population Size: 100

output for GA: (see) **ga.txt**
output for MA: (see) **ma.txt**

CSP Input1:

days: 3 slots: 3 rooms: 2
for each room:totalSlots: 9
courses count: 3
for all rooms: total slots: 18

--> length of chromosome: 18

Input 2:
days: 5 slots: 8 rooms: 5
for each room:totalSlots: 40
for all rooms: total slots: 200
--> length of chromosome: 200

output 2: Terminated after Max Iterations

output: (see) **csp.txt**

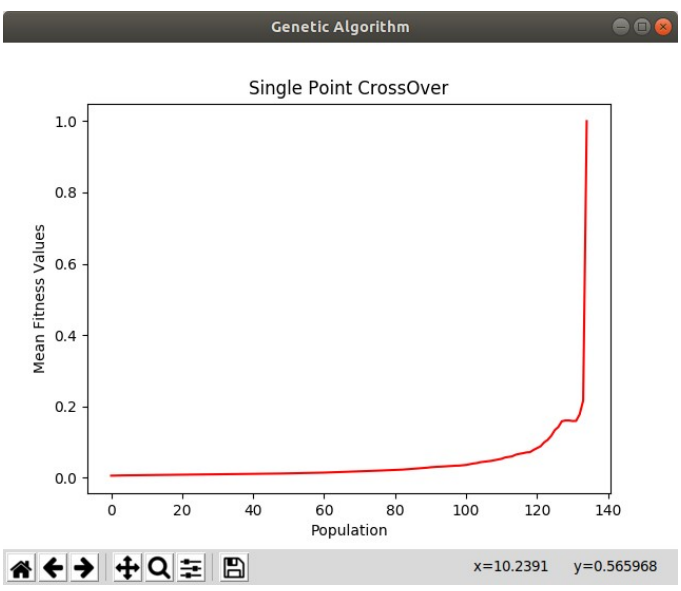
		Genetic Algorithm			Memetic Algoithm			Backtracking search CSP
		Single Point CrossOver	Multi Point CrossOver	Uniform CrossOver	Single Point CrossOver	Multi Point CrossOver	Uniform CrossOver	
Input 1	Iterations	133	234	152	125	176	145	22283
	Time	0:00:19 sec	0:00:31 sec	0:00:25 sec	0:00:46 sec	0:00:48 sec	0:00:54 sec	0:00:55 sec

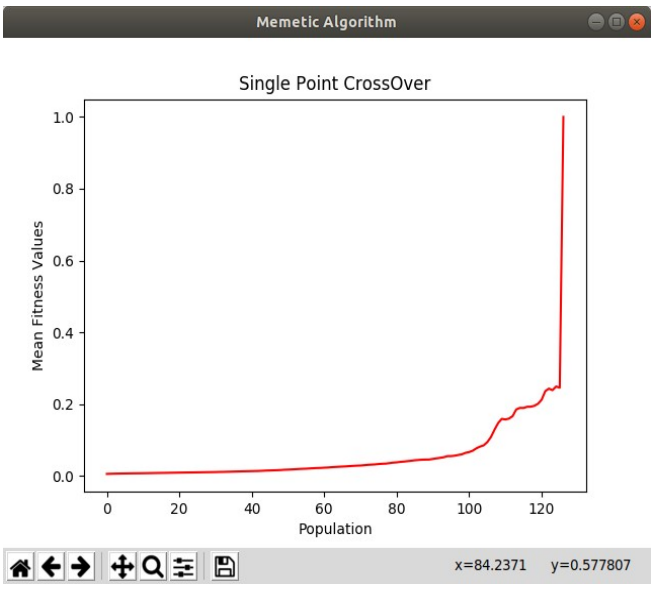
Notice that Memetic algorithm reduces the iteration w.r.t to Genetic algorithm. This is due to we implemented local search(Hill climbing) to children and add our best children to population. So, our convergence will be faster than GA.

Plots for Genetic and Memetic Algorithms Analysis :

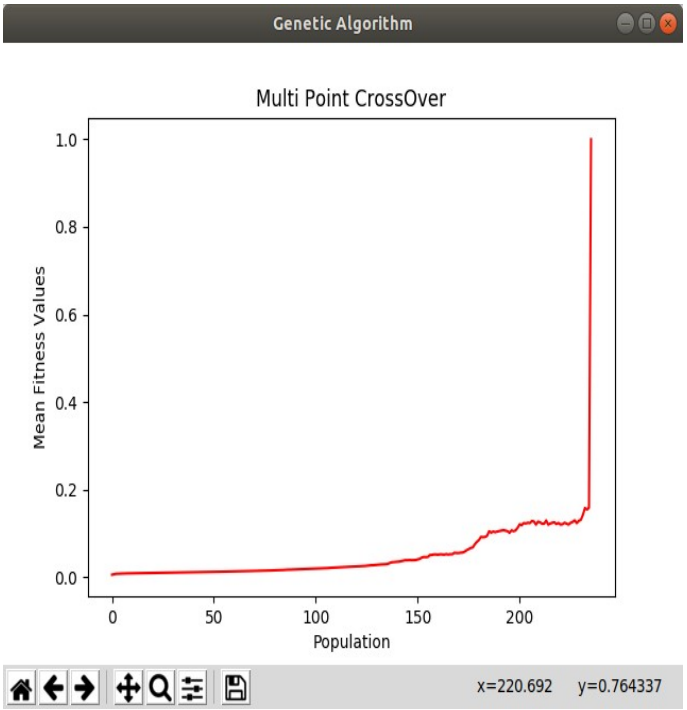
On same initial population done the GA and MA and ploted the graph by taking ‘Mean of all fitness values of each population’ in every iteration against ‘Population size’.At last added final chromosome fitness value, so it is a sudden peak in graph.

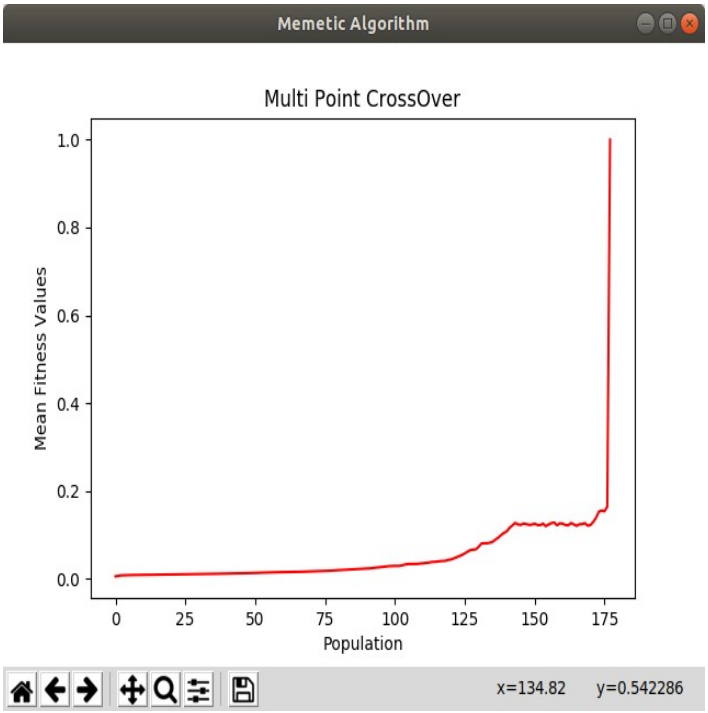
Single Point CrossOver in GA vs MA:



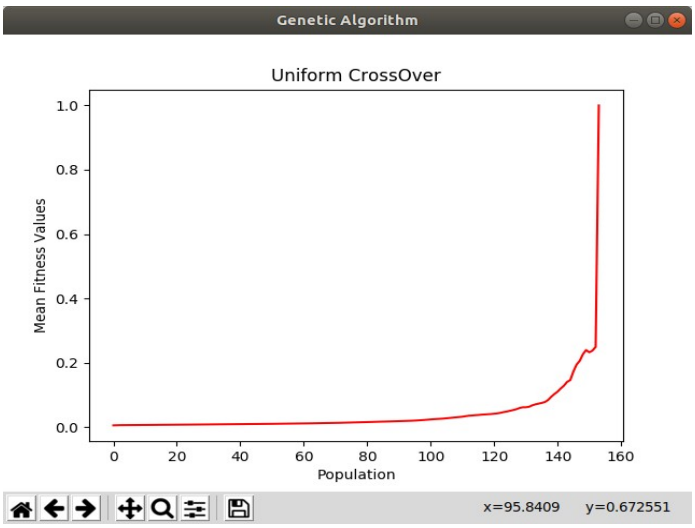


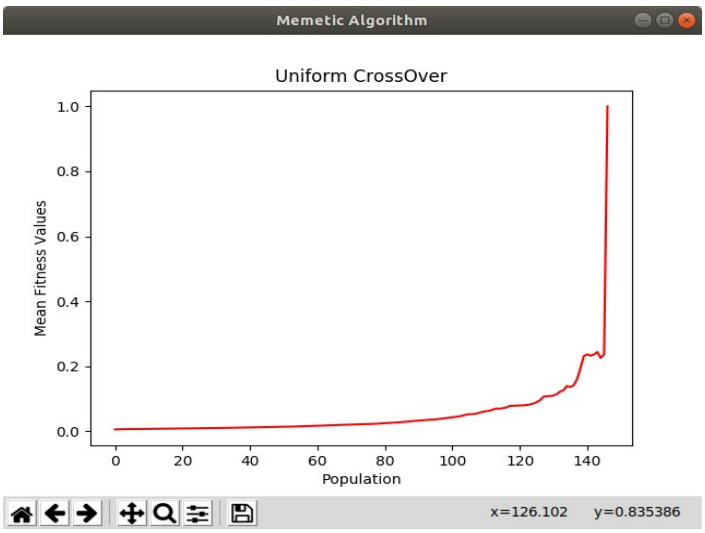
Multi Point CrossOver in GA vs MA:





Uniform CrossOver in GA vs MA:





<u>Assumptions</u>:

- Giving input logically i.e., there exits solution.
- Only one professor to each course (not more than one professor).
- Only one course can be handled by professor (not multiple courses).