Student Name: Ali Muhammad Asad Student ID: aa07190

19/20

CS 451 – Computational Intelligence
Spring' 2024
Quiz 01

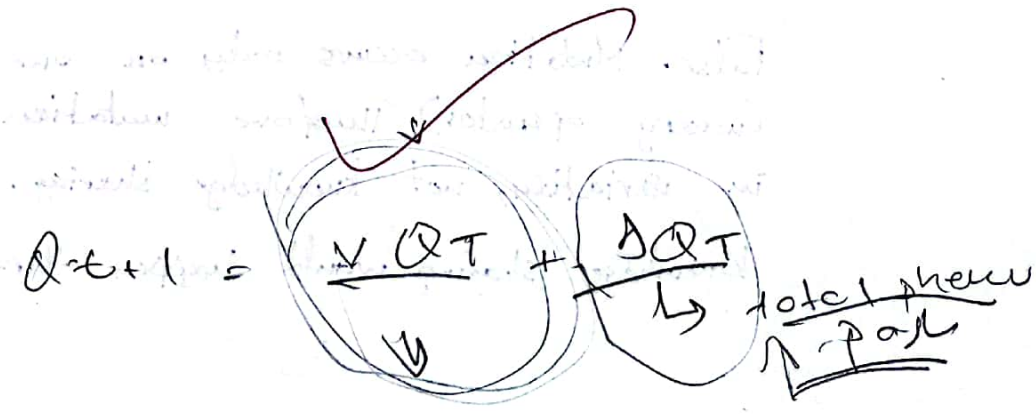
1. Identify the following statements as True/False. Give brief, to-the-point reason to justify your answer:

   a. Truncation is more exploitative than Rank based selection. True.

Truncation selects the best or the fittest survivors everytime for the next generation, therefore, the fitness remains clustered to a certain area. In Rank based, absolute value of fitness is not used, therefore, it still has some diversity. Hence, Truncation is more exploitative.

   b. Increasing the size of population increases exploitation.

False. This would lead to more variety, hence more exploration rather than exploitation.



Tournament Selection

c. ~~Binary tournament~~ becomes similar to truncation as the size of tournament approaches population size. True

In tournament selection, the best out of the tournament size is used. Therefore, on increasing the size of selection, ~~the chances of~~ we pick the best candidates more often. Therefore it becomes more similar to truncation.

d. High selective pressure may result in slow convergence of the evolution process.

False. If selective pressure is high, the best candidate takes over the whole population relatively faster, hence convergence happens faster. Can also lead to premature convergence.

e. Knowledge sharing is an important part of evolution that is achieved via mutation operator.

False. Mutation occurs only on one organism (unary operator). Therefore mutation results in variation, not knowledge sharing.

Knowledge sharing would happen through crossover.

2. What is Combinatorial Optimization (CO)? Why is it generally considered difficult to solve than continuous optimization? Give three examples of problems that are
   a) combinatorial optimization problems,
   b) not combinatorial optimization problems.

CO is the process of searching for a local or global peak or a maxima / minima when the sample space is discrete but large enough. combination / permutation
It is difficult to solve such problems due to the sheer number of computations required to solve the problem. Such problems might not have a good strategy.

(a) Such as the traveling salesman problem, the knapsacking problem, or hill climbing problem. Even the Minimum Spanning Tree Problem

(b) Functional Derivatives, neural networks solving a system of equations.

-1

3. Why is it important to maintain diversity in an evolutionary process. Identify factors/steps that contribute to maintaining diversity during evolution.

Diversity is important as in some situations, there might not be one single best answer, such as a gameplay strategy. Or there might be multiple peaks that can count as good answers, or there might even be a best peak.
Having diversity means more of those opportunities will be explored. If there is less diversity, the population might converge on a local maxima/minima rather than a global one, or even a better one.
→ having a large population, introducing mutation, and replacement schemes a good selection scheme, they all contribute to maintaing diversity.

4. What would you do to make sure that your evolutionary process eventually converges?

Based on some fixed criteria, the worst of the population can be discarded, or only the best may be picked for the next generation.

Student Name: Ali Muhammad Asad Student ID: 07190

9/10

CS 451 – Computational Intelligence
Spring' 2024
Quiz 02

Q1 – In Ant Colony Optimization (ACO),

a) It is important to initialize the pheromone table T_{ij} with some initial concentration of pheromone. (True/False)? Why?

~~True, it has~~ ~~True, since~~ ~~True, we~~

False, since we want the ants to explore the space

b) What is the role of n_{ij}, desirability, in the context of TSP and any other problem?

If we are talking about TSP, ~~then~~ then in TSP we aim to find the shortest possible route such that each city is visited exactly once. Consider the TSP route as a graph, where a city is a node & its edges are the roads to other cities. Then n_{ij} represents the inverse of the edge distance for nodes i & j. We aim to ~~minimize~~ maximize this, since we want to minimize our distance travelled.

what does this represent in general?

c) How does ACO algorithm (not intuition) ensures that better solutions are reinforced more than other solutions?

In the ACO algorithm, the shorter paths will have more pheromones as compared to longer paths. Since any single ant will deposit more pheromones on that trail. So other ants will pick up on it more. The evaporation aspect of the algorithm thus evaporates older trails, & prevents ants from converging on a sub optimal solution. In addition, setting a min, max on pheromone levels ensures that ants best utilize the limited pheromones, so shorter solutions are explored.

Q2 – In the CI Assignment 2,

a) What was the representation of your chromosome in JSSP? How was it initialized?

~~A chromosome in JSSP~~

A single chromosome in JSSP represents a sequence of operations, where each operation was represented as a tuple of a job & a machine. It was initialized by creating random permutations of the operations. ~~& the sequence of positions~~ Each permutation was a chromosome, & collectively they make up the population.

How is ordering constraint ensured?

b) Assume that, instead of JSSP, you are solving a variant of JSSP known as Flexible JSSP (FJSSP). In the FJSSP, there are a set of machines $A = M_1,\ldots, M_m$, and a set of jobs, $J = J_1,\ldots, J_n$ so that each job J_i consists of a given sequence of n_i operations, $O_{i,1}, O_{i,2},\ldots, O_{i,ni}$. Each operation $O_{i,j}$ can be processed on any machine of a subset $A_{i,j} \subseteq A$.

What changes (if any) will you do in you representation of the problem or in any of your functions (fitness, crossover, mutation)?

~~Initially, the JSSP was run on a single set of machines. In FJSSP, we have~~

I don't think I will make much changes in my fitness function, ~~however,~~ nor in how the problem was represented. However, in my mutation function, I would now mutate the tuple itself, such that a job is run on a different machine $A_{i,j}$, ~~so that~~ ~~similarly the crossover function~~ permutations can be tried. The crossover would work as before.

I would only change the mutation function, to explore more possibilities by changing the machine associated with the job. Thus, a ~~random~~ completely random population would also be initialized.

Student Name: Ali Muhammad Asad Student ID: 07190

7.5/15

CS 451 – Computational Intelligence
Spring' 2024
Quiz 03

Q1- Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (have smaller distance) to each other than to those in other groups (clusters). KMeans is a simple and popular method of data clustering. However, you have to specify your number of clusters (K) to KMeans. After taking CI course, you are interested in applying PSO to perform data clustering without specifying number of clusters. The objective is to achieve optimal clustering with as few clusters as possible. Discuss the formulation of PSO to cluster given datapoints. Your formulation should cover the following:

a) [3 points] Representation of particle

We can model each particle to act as a neuron, where the number of particles have to be significantly large, & not too small. Each particle would be initialized randomly, & would have the desired traits as attributes. Particles with similar attributes upto a certain range would cluster together as our iterations move forward.

clustering information?

particle ... SOM ... BMU is found.

b) [2 points] Fitness function

The fitness function would calculate of the particles fitness with respect to the different clusters, or different particles in the swarm. Then whatever cluster / group of particles / group our particle has fitness value closest to, that would be selected. Basically it would give of an average of the attributes / values it has in the given dataset which would be our fitness.

fitness function,

c) [2 points] Mechanism to update velocity

Whatever fitness value our function gives off will be compared to the concentrations that most particles are centered around. Then the closest one will be selected, & the velocity function would apply a vector towards that data points by difference in that data point & the particle's position.

Q2- [3 points] In PSO, all particles move towards the global best and eventually converge there. How does exploration happen in PSO?

Initially, all particles are generated randomly, and don't know where the best solution is, an example could be of a food source. But they know how far away the food source is. Each particle maintains its local best out of its fitness values, while the swarm as a whole keeps track of the global best which is taken out of the best of all the local best values. The particles then move towards that global best value based on a velocity vector that is applied to each particle (uniquely), which updates not only their velocity but also position. Then the fitness values are computed again & the whole process repeats. This way, eventually all particles converge towards the global best.

Q3 - a) [3 points] What is the role of learning rate and neighborhood function in Self-Organizing Maps (SOM)?

- Learning rate

The learning rate is a value between 0 & 1 which basically tells the rate at which the competitive learning of the model takes place. Usually it is set to a fixed value. The learning rate affects the convergence / finding an optimal solution for the data set. Sometimes a higher learning rate can lead to worse fitness values, therefore, it should be adjusted through trial & error.

- Neighborhood function

The neighborhood function is a function which ~~affects~~ tells how a neuron affects the neurons in its neighborhood. When the winning neuron is found, its neighborhood is calculated using the neighborhood function, & all neurons lying within the radius of that neuron's neighborhood belong to a certain group/cluster or possess similar properties.

b) [2 points] How does SOM differ from conventional clustering techniques like K-means.

1.5

~~SOM~~ SOM differs from conventional clustering techniques such that in SOM each neuron has a specific radius based on which its cluster / neighboring neurons are found. This reduces / shrinks on each iteration due to which the neurons cluster towards a specific group / neuron. Thus, we only select the nuber of neurons whereas in K-means clustering we set the nuber of & clusters, K, to K-means, across the ~~dataset~~ data set.

Spatial clustering

Thus, in SOM, clusters aren't specified, but found / calculated based on the neurons with similar properties.

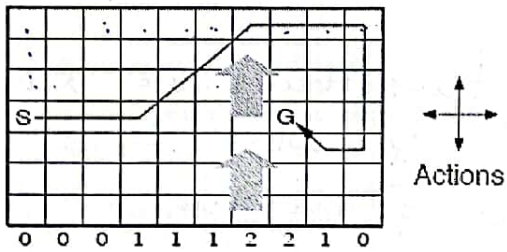
Student Name: Ali Muhammad Student ID: 07190

13/20

CS 451 – Computational Intelligence
Spring' 2024
Quiz 04

Q1- You have learnt how to use RL to learn an optimal policy for an agent to navigate in a gridworld. Your grid is now changed to a *windy Gridworld*, shown in the figure below. This is a standard gridworld, with start and goal states but with one difference: there is a crosswind upward through the middle of the grid. The actions are the standard four (up, down, right, and left) but in the middle region, the resultant next states are shifted upward by a "wind" the strength of which varies from column to column. The strength of the wind is given below each column, in number of cells shifted upward. For example, if you are one cell to the right of the goal, then the action left takes you to the cell just above the goal. The actions otherwise are deterministic and there is no stochasticity in the environment.

You need to apply RL to determine the optimal policy for an agent to reach the goal state.



38
15

a) Discuss the formulation of problem including the MDP representation and reward function.

States: Our starting state can be S_0, & each subsequent state can be labelled as S_1, S_2, S_3 --- where we can move from our current state to either the state at the right, left, up or down.

$S = \{S_0, S_1, S_2, \ldots S_n\}$ where 'n' are total cells in our grid world.

Actions:

We have 4 standard actions,

up → causes agent to go up (state above current state)

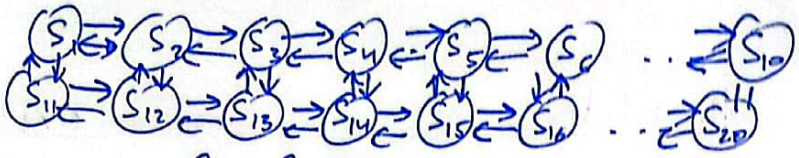
down → causes agent to go down (state below current state)

left → agent goes to state at left of current state.

right → agent goest to state at right of current state.

$A = \{u, d, l, r\}$

→ Consider states formulated as shown:



State Transitions: $A = \{a_0, a_1, a_2, a_3\} = \{u, d, r, l\}$

↓ Grid World as per figure

Then: $Q(S_1, d) = S_{11}$

$Q(S_1, r) = S_2$

$Q(S_2, l) = S_1$

$Q(S_{12}, u) = S_2$

and so on.

$Q(S_t, u) = S_{t-10}$

$Q(S_t, d) = S_{t+10}$

$Q(S_t, r) = S_{t+1}$

$Q(S_t, l) = S_{t-1}$ in General.

affect of wind?

Transition Probabilities:

Since our transitions are dependent on the wind, if wind is 0, we have 100% chance of reaching our destination/state.

~~If wind is 1, if we move up, we have 0% chance of reaching intended state,~~ If wind is 1, we have 100% probability of reaching state S_{t-9} if 'r', if 'l' S_{t-11}, if 'u' S_{t-10} if 'd', S_{t-30} if 'u', S_{t-19} if 'r', S_{t-21} if 'l'

If wind is 2, 100% chance to reach: S_{t+20}, if 'd' S_{t+30}, if 'u' S_{t+19} if 'r' S_{t+21} if 'l' S_t.

How do you generalize this?

Reward Function:

b) What would you do to ensure that the resultant policy gives shortest path to the goal?

~~More reward would be given~~

Keep a higher reward if the state is closer to the goal state, if the robot achieves the state it intended to go to. The values would be updated accordingly for each state.

c) You want your agent to avoid getting closer to the boundary (wherever possible). How would you handle this requirement?

For all the states that are ~~closer to~~ at the boundary, we will either lessen the reward at those states by a certain factor, or introduce a penalty. Since we want to avoid boundaries, a penalty would be more suited as this would lessen the reward value not only at that state, but also to the states near the boundary. So this in turn would cause the robot to prefer cells towards the center of the grid more ~~rather than~~ and avoid the boundary.

Q2 - What is the core idea of temporal difference learning? How is it different from value iteration or conventional episodic learning? Explain the following equation in the context of TD learning.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

In value iteration or conventional episodic learning, the value for reward is calculated & set once an episode has been completed, & set accordingly to the max value. In temporal difference learning, however, the reward value is calculated at each step & set. This leads to a much faster convergence, but also has the risk of getting stuck at a suboptimal solution or a local optima which might not be good or the best.

In the above equation, value of the updated state becomes value of current state + plus the learning rate 'α' times the R value added to the value of the next state with some discount factor with the difference of the value of the current state which gives us the net change in value for a particular state.

9/6