**COMSATS University Islamabad (Lahore Campus)**

**□Midterm Exam □Terminal Examination –** **Fall 2024**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Course Title: | Computational Intelligence | Course Code: | CSC 347 | Credit Hours: | 3(3, 1) |
| Course Instructor/s: | Asia Maqsood | Programme Name: | BSCS | | |
| **Time Allowed:** | **90 minutes** | **Maximum Marks:** | | **25** | |
| **Important Instructions / Guidelines:**  Please write your Name and Roll Number on the question paper and return it with your answer sheet. | | | | | |

**Question No 1. Marks: 5**

***CLO: <1>; Bloom Taxonomy Level: <Understanding>***

Explain the historical evolution of computational intelligence and how its paradigms (e.g., Artificial Neural Networks and Fuzzy Systems) have evolved to address complex real-world problems.

**Answer:**

The historical evolution of *Computational Intelligence (CI)* began in the mid-20th century, inspired by mimicking human-like decision-making and learning. Initially, it focused on three primary paradigms: **Artificial Neural Networks (ANNs), Fuzzy Systems,** and **Evolutionary Computation (EC)**.

1. **Artificial Neural Networks (ANNs)**: Inspired by the human brain, ANNs emerged in the 1940s with McCulloch and Pitts' model. However, due to computational limitations, they didn’t gain momentum until the 1980s when the backpropagation algorithm was developed, enabling multi-layered networks. Over time, advancements led to *Deep Learning*, allowing ANNs to solve complex tasks like image recognition and natural language processing.
2. **Fuzzy Systems**: Introduced by Lotfi Zadeh in the 1960s, *Fuzzy Logic* allowed handling uncertainty and imprecision in decision-making. This was crucial for problems where binary logic failed. Fuzzy systems have been widely used in control systems (e.g., washing machines, and air conditioning) and decision-making processes that require approximate reasoning.
3. **Evolutionary Computation (EC)**: Inspired by natural evolution, this paradigm includes algorithms like Genetic Algorithms (GA), introduced in the 1970s. EC mimics biological processes like selection, crossover, and mutation to solve optimization problems.

Together, these paradigms evolved to address increasingly complex real-world issues by combining strengths (e.g., *neuro-fuzzy systems* or hybrid *GA-ANN models*). This integration allows CI systems to better handle uncertainty, learn from data, and optimize complex processes in fields like healthcare, robotics, and finance.

**Question No 2. Marks: 10**

***CLO: <2>; Bloom Taxonomy Level: <Applying>***

You are part of a research team developing a Genetic Algorithm (GA) to optimize delivery truck routes in a metropolitan area. Your objective is to minimize fuel consumption and delivery time while maximizing customer satisfaction. To test your algorithm, consider the following scenario:

**Scenario Details:**

* There are 5 delivery trucks and 15 delivery locations.
* The average fuel consumption rate for each truck is 12 km/liter.
* Each delivery location has a specific priority level on a scale of 1 to 5 (5 being the highest priority).
* The average delivery time for each location is recorded, with distances between locations ranging from 2 km to 10 km.
* The company aims to achieve a customer satisfaction score of at least 85%.

1. Select an appropriate encoding strategy to represent the delivery routes of the trucks, considering the constraints and objectives of the problem. Justify your choice of encoding based on the characteristics of the scenario, including route constraints and the number of delivery locations.

**Answer:**

**Encoding Strategy: Permutation Encoding**

**Reason for Choosing Permutation Encoding:** Permutation encoding is the most suitable approach for this problem. In permutation encoding, each chromosome represents a sequence of delivery locations, which aligns well with the route optimization problem. The sequence specifies the order in which the delivery locations will be visited by each truck.

For example, if we have 5 trucks and 15 delivery locations, a possible chromosome could be represented as:

* **Chromosome Representation:** [Truck 1: (3, 7, 1), Truck 2: (5, 12, 9), Truck 3: (2, 8, 6), Truck 4: (4, 13, 10), Truck 5: (11, 14, 15)]

In this example:

* Truck 1 visits delivery locations 3, 7, and 1 in that order.
* Truck 2 visits delivery locations 5, 12, and 9, and so on.

This encoding method ensures that all delivery locations are covered and visited in a specific order, which is essential for minimizing fuel consumption, delivery time, and maximizing customer satisfaction.

1. Design a fitness function that considers fuel consumption, delivery time, and customer satisfaction. Discuss how you would address issues such as premature convergence and slow finishing in your GA implementation.

**Answer:**

**Fitness Function Calculation**

The fitness function will consider the following three factors:

1. **Fuel Consumption**:

Calculated as:

1. **Delivery Time**: Estimated using the average speed of the truck. Suppose the truck's average speed is 40 km/h, and the delivery time for a location is calculated as:
2. **Customer Satisfaction**: Prioritize high-priority locations in the route. Assume that delivering to the highest priority location (priority level 5) within the specified time window increases the satisfaction score.

**Example Calculation:**

Assume the delivery truck travels 40 km to visit 5 locations.

* **Fuel Consumption**:
* Fuel Consumption=40 km/12 km/liter=3.33 liters
* **Delivery Time** for each location:
* Location 1 (10 km away): 10 km/40 km/h=0.25 hours
* Total delivery time for all locations = Sum of individual delivery times.
* The **Customer Satisfaction Score** is calculated based on delivering high-priority orders promptly. For example, if 80% of the high-priority deliveries (priority level 4 or 5) are on time, we set the satisfaction score accordingly.

1. Select and explain at least two basic GA operators that you would use in your algorithm. Justify why these operators are suitable for your problem scenario.

**Answer:**

* 1. **Crossover (Recombination)**

**Purpose:** The crossover operator is used to combine information from two parent solutions to produce offspring solutions. This mimics the process of genetic recombination in nature, where offspring inherit traits from both parents.

**Justification:**

* **In Route Optimization:** Crossover is essential in our scenario because it allows combining efficient routes from different parent solutions, potentially creating a more optimal delivery route. For instance, if one parent has an efficient set of routes for high-priority locations, and the other optimizes fuel consumption, combining these routes can lead to a solution that balances both objectives.
* **Creating Diversity:** Crossover introduces diversity to the population by combining routes differently, preventing the algorithm from getting stuck in local optima and exploring a broader range of potential solutions.

A suitable crossover technique here could be *Ordered Crossover (OX)*, which ensures that offspring inherit valid routes without repeating locations.

* 1. **Mutation**

**Purpose:** Mutation introduces small random changes in an individual solution to maintain diversity and explore new potential solutions that may not arise through crossover alone.

**Justification:**

* **Exploring New Routes:** In the context of delivery routes, mutation can help in exploring new paths or slight deviations in a truck’s route that may result in lower delivery time or reduced fuel consumption. For example, swapping two delivery locations, reversing a segment of a route, or inserting a high-priority location earlier in the route can lead to discovering more efficient delivery paths.
* **Avoiding Premature Convergence:** Mutation prevents the algorithm from converging too early on suboptimal solutions. Given the importance of balancing multiple objectives (delivery time, fuel efficiency, and customer satisfaction), mutation can help the GA explore alternative routes that improve overall performance.

1. Propose initial values for the control parameters of your GA, including population size, crossover rate, and mutation rate. Demonstrate how these parameters might influence the performance of your algorithm using the scenario data.

**Answer:**

**Control Parameters Calculation**

* **Population Size**: Initial value = 50
* **Crossover Rate**: 0.8 (80% of the population undergoes crossover)
* **Mutation Rate**: 0.1 (10% of the population undergoes mutation)

We can calculate the number of individuals affected:

* Crossover: 50×0.8=40 individuals
* Mutation: 50×0.1=5 individuals

These values help maintain diversity in the population, preventing premature convergence to suboptimal solutions.

**Question No 3. Marks: 10**

***CLO: <2>; Bloom Taxonomy Level: <Applying>***

You are part of a software development team tasked with creating an intelligent recommendation system for an e-commerce platform. The system aims to personalize product recommendations for users based on their browsing history, preferences, and similar user behaviors. Your team has decided to use Particle Swarm Optimization (PSO) to optimize the parameters of the recommendation algorithm. Consider the following scenario:

**Scenario Details:**

* The recommendation system includes 10 tunable parameters (e.g., weight factors for browsing history, similarity scores, product ratings).
* The objective is to maximize a recommendation accuracy score that ranges from 0 to 100.
* Each parameter in the system can have a value between 0 and 1.
* The PSO algorithm will have a swarm of 20 particles with an initial velocity range of [-0.5, 0.5].

1. Implement how you would apply the original PSO algorithm to optimize the parameters of your recommendation system using the given scenario values. Illustrate how the particles represent potential solutions and how their movement is influenced by their personal best positions and the global best position.

**Answer:**

* Each particle in the swarm represents a possible solution (parameter values).
* Particle positions and velocities are updated using the equations:

A screenshot of a computer

Description automatically generated

* where:
  + is the velocity of particle i.
  + is the position of particle iii.
  + ω is the inertia weight (controls the exploration).
  + ​ and are acceleration coefficients (usually set to 2).
  + and are random numbers in the range [0, 1].
  + pBest is the best-known position of the individual particle.
  + gBest is the global best position found by the swarm.

**Example Calculation for One Iteration:**

* Assume ω=0.7, =1.5, =1.5.
* Current position of the particle: =0.3
* Current velocity of the particle: = 0.1
* Personal best position: pBesti = 0.4
* Global best position: gBest = 0.5

(t+1) = 0.7×0.1+1.5×0.4 × (0.4−0.3) + 1.5×0.5 × (0.5−0.3)

​(t+1) = 0.07+0.06+0.15=0.28

(t+1) = 0.3+0.28=0.58

1. Apply the constriction factor approach to your PSO implementation using the scenario values.

**Answer:**

**Step 1:** Define the PSO Parameters

**Scenario Details:**

* **Number of Particles (N):** 20
* **Number of Tunable Parameters (D):** 10
* **Velocity Range:** [-0.5, 0.5]
* **Constriction Factor (φ):** Commonly used values for the constriction factor are derived from the formula:

A screenshot of a computer

Description automatically generated

where ϕ1, ϕ2​ are cognitive and social parameters, respectively. Typical values are ϕ1=2.0., ϕ2​=2.0, which results in:

A screenshot of a computer

Description automatically generated

Instead, a commonly accepted fixed value for φ is around 0.72984.

**Step 2: Velocity Update Equation with Constriction Factor**

The standard velocity update equation in PSO can be expressed as:

A screenshot of a computer

Description automatically generated

**Step 3: Position Update Equation**

The position update equation remains unchanged:

A screenshot of a computer

Description automatically generated

**Step 4: Initialize Particles**

1. **Initialization of Particles:**

* Initialize 20 particles, each with 10 tunable parameters.
* Each parameter is randomly initialized in the range [0, 1].
* Each particle’s velocity is randomly initialized in the range [-0.5, 0.5].

1. **Personal Best (p\_best) and Global Best (g\_best):**

* Each particle retains its personal best position (p\_best) and the global best position (g\_best) is updated based on the accuracy scores.

**Step 5: Iteration of PSO with Constriction Factor**

The PSO algorithm iteratively updates the velocity and position of each particle based on the constriction factor:

1. **For each iteration:**

* Calculate the fitness (recommendation accuracy) for each particle’s current position.
* Update p\_best if the current fitness is better than the previous p\_best.
* Update g\_best based on the best fitness among all particles.
* Update the velocity using the constriction factor equation.
* Update the position using the position update equation.
* Ensure that each parameter value stays within the bounds [0, 1].

**Example Calculation**

Let’s illustrate the calculations for one iteration of one particle.

1. **Particle Initialization:**

* Random initial position: x1,j=[0.2,0.5,0.3,0.6,0.1,0.7,0.4,0.9,0.8,0.0]
  + Random initial velocity: v1,j=[0.1,−0.2,0.3,0.0,−0.1,0.2,0.1,−0.3,0.0,0.5]

1. **Personal Best (p\_best):**

* Assume p\_best = [0.2,0.5,0.3,0.6,0.1,0.7,0.4,0.9,0.8,0.0]

1. **Global Best (g\_best):**

* Assume g\_best = [0.3,0.5,0.5,0.4,0.2,0.8,0.6,0.7,0.9,0.1]and its corresponding fitness score is 80.

1. **Random Numbers:**

* r1=0.3and r2=0.6

1. **Velocity Update:** Using φ=0.72984:

v1,j(t+1)=0.72984⋅(0.1+2.0⋅0.3⋅(0.2−0.2)+2.0⋅0.6⋅(0.3−0.2))

For j=1:

v1,1(t+1)=0.72984⋅(0.1+0+0.12)=0.72984⋅0.22≈0.1606

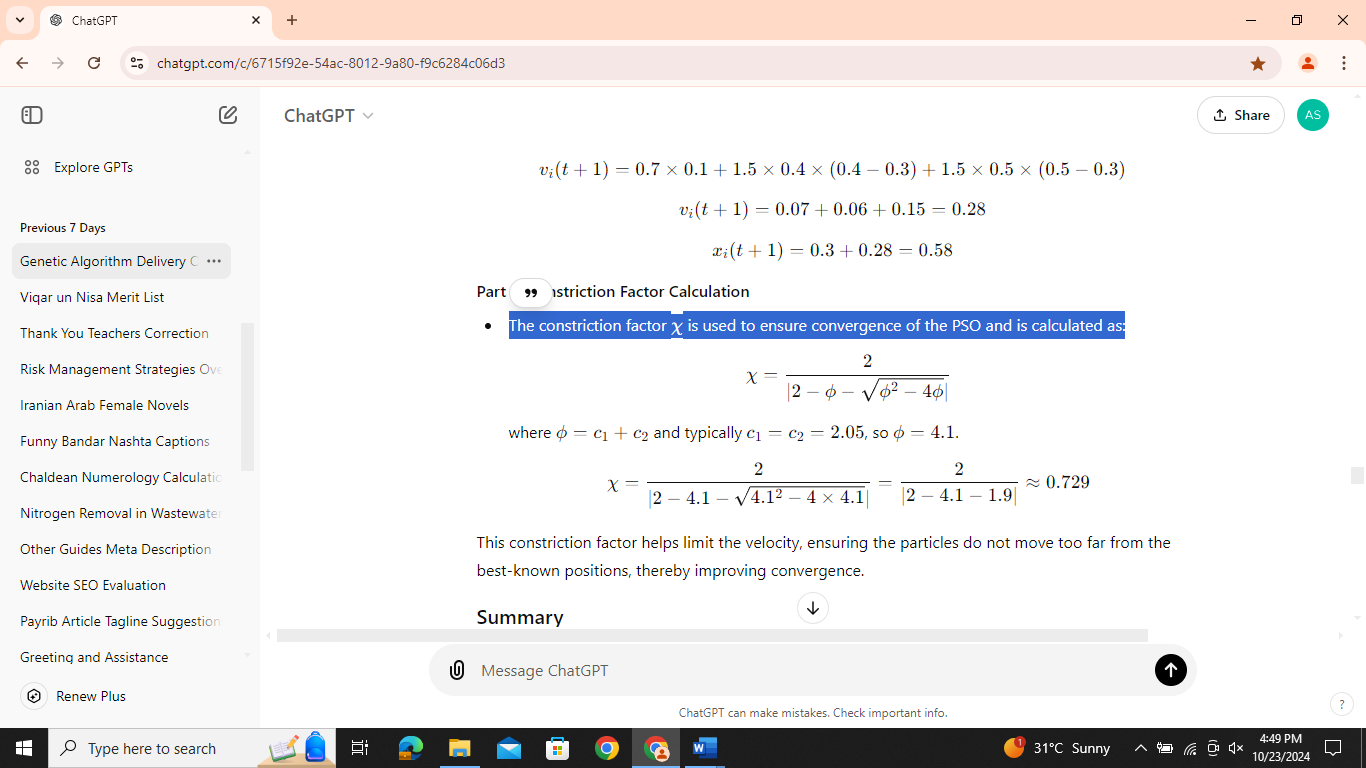
1. **Position Update:**

x1,1(t+1)=0.2+0.1606≈0.3606

1. Calculate how the constriction factor would affect the convergence of the algorithm and demonstrate its benefits in balancing exploration and exploitation of the search space.

**Answer:**

The constriction factor i is used to ensure convergence of the PSO and is calculated as:



where ϕ=c1+c2​ and typically c1=c2=2., so ϕ=4.1.

This constriction factor helps limit the velocity, ensuring the particles do not move too far from the best-known positions, thereby improving convergence.