7-Practice

Theme: Role Assignment Program

Object: concept of role assignment in multi-agent systems, with particular emphasis on understanding how roles are distributed among agents from both theoretical and practical perspectives.

Theory:

Role assignment in multi-agent systems is the process of distributing tasks and defining the roles of individual agents in order to achieve a common goal.

This process is essential for solving complex problems, as the system operates more efficiently when each agent performs tasks that align with its specific capabilities and assigned role.

Core Elements of Role Assignment in Multi-Agent Systems

1. Agents. Multiple agents exist in the system, each possessing specific capabilities and constraints. The role of each agent defines the type of task it is responsible for performing.

2. Tasks. A set of tasks that must be completed to achieve a common objective. For example, in traffic optimization, one agent may control traffic signals, while another agent directs vehicle routing.

3. Role Assignment Algorithm.

An algorithm is designed to determine the roles of agents.
Different approaches can be applied:

- **Priority-Based Assignment:** Each agent's task is determined according to its priority level.
- **Optimization-Based Assignment:** An optimization algorithm is used to assign roles in the most efficient and effective manner.

Main Approaches to Role Assignment in Multi-Agent Systems

1. Priority-Based Assignment
Each agent's task is determined according to its priority level.
For example, one agent may be designed to minimize energy consumption, while another agent focuses on reducing water usage.
2. Optimization-Based Assignment

An algorithm is used to assign roles to agents in the most optimal way.
For example, in **Distributed Constraint Optimization Problems (DCOP)**, agents' roles are determined optimally to solve complex coordination problems under constraints.

3. Contract Net Protocol (CNP)

One agent acts as a manager and announces a task, asking other agents whether they are capable of performing it.
For example, one agent may handle product placement, while another agent is responsible for transportation.

Role Assignment in Multi-Agent Systems (Python Implementation)

In multi-agent systems, role assignment ensures coordinated and efficient collaboration among agents. Below, we describe a simple Python-based role assignment program in which agents are automatically assigned as either a **leader** or a **worker**.

Program Description

- Each agent is created with an **identifier (ID)** and a **skill level**.
- The agent with the highest skill (or experience) level is designated as the **leader**.
- The remaining agents are assigned the role of **workers**.
- Agents communicate with each other while performing their assigned roles to achieve coordinated task execution.

Applications

- **Robotics:** Assigning leader and follower (worker) roles among drones or autonomous robots.
- **Collaborative AI:** Distributing roles among agents in multi-player games or cooperative artificial intelligence environments.
- **Organizational Systems:** Assigning roles such as managers and workers within structured organizational models.

To further enhance this approach, adaptive role switching mechanisms or **Nash equilibrium–based methods** can be incorporated to enable strategic and dynamic role allocation.

Practice

```
import random
```

```
class Agent:
    def __init__(self, name, skill_level):
        self.name = name
        self.skill_level = skill_level
        self.role = None

    def assign_role(self, role):
        self.role = role

    def perform_task(self):
        if self.role == "Leader":
            print(f"{self.name} ({self.role}): Defining the
strategy.")
        elif self.role == "Worker":
            print(f"{self.name} ({self.role}): Executing the task.")

# Creating agents
agents = [
    Agent("Agent_1", random.randint(1, 100)),
    Agent("Agent_2", random.randint(1, 100)),
    Agent("Agent_3", random.randint(1, 100)),
    Agent("Agent_4", random.randint(1, 100))
]

# Assign the most skilled agent as the leader
leader = max(agents, key=lambda agent: agent.skill_level)
leader.assign_role("Leader")

# Assign the remaining agents as workers
for agent in agents:
    if agent != leader:
        agent.assign_role("Worker")

# Display assigned roles and execute actions
print("=== Agent Roles Have Been Assigned ===")
for agent in agents:
    print(f"{agent.name}: {agent.role} (Skill Level:
{agent.skill_level})")

print("\n=== Agents Are Performing Their Tasks ===")
for agent in agents:
    agent.perform_task()
```

```
=== Agent Roles Have Been Assigned ===
Agent_1: Worker (Skill Level: 25)
Agent_2: Leader (Skill Level: 66)
Agent_3: Worker (Skill Level: 40)
Agent_4: Worker (Skill Level: 41)

=== Agents Are Performing Their Tasks ===
Agent_1 (Worker): Executing the task.
Agent_2 (Leader): Defining the strategy.
Agent_3 (Worker): Executing the task.
Agent_4 (Worker): Executing the task.
```

Below is a simple example of role assignment. In this example, two agents (Agent A and Agent B) collaborate to accomplish tasks in a given environment. The role of each agent is determined based on its capabilities and the current state of the environment.

```
import random

class Environment:
    def __init__(self):
        # Tasks in the environment: 'clean', 'move', 'inspect'
        self.tasks = ['clean', 'move', 'inspect']
        self.agents = []  # List of agents in the environment

    def add_agent(self, agent):
        """Add an agent to the environment"""
        self.agents.append(agent)

    def assign_roles(self):
        """Assign roles to agents"""
        roles = {}  # Mapping between roles and agents
        assigned_tasks = set()  # Set of already assigned tasks

        for agent in self.agents:
            # Select available tasks that are not yet assigned
            available_tasks = [task for task in self.tasks if task not
in assigned_tasks]
            if not available_tasks:
                roles[agent.name] = "idle"  # If no tasks remain, the
agent waits
                print(f"{agent.name} -> Waiting (idle).")
                continue

            # Select role based on agent capabilities
            role = agent.decide_role(available_tasks)
            roles[agent.name] = role
            assigned_tasks.add(role)
            print(f"{agent.name} -> Responsible for '{role}' task.")
```

```
        return roles


class Agent:
    def __init__(self, name, capabilities):
        self.name = name
        self.capabilities = capabilities  # Agent's capabilities

    def decide_role(self, available_tasks):
        """Select the most suitable task"""
        # Choose task based on agent capabilities
        possible_roles = [task for task in available_tasks if task in
self.capabilities]
        if possible_roles:
            return random.choice(possible_roles)  # Randomly select
among possible tasks
        else:
            return "idle"  # If no suitable capability exists, remain
idle


# Main program
if __name__ == "__main__":
    env = Environment()  # Create environment

    # Create agents
    agent_a = Agent("Agent A", ['clean', 'move'])
    agent_b = Agent("Agent B", ['move', 'inspect'])

    # Add agents to the environment
    env.add_agent(agent_a)
    env.add_agent(agent_b)

    print("Role assignment started:")
    roles = env.assign_roles()

    # Print results
    print("\nRoles have been assigned:")
    for agent_name, role in roles.items():
        print(f"{agent_name} -> {role}")
```

```
Role assignment started:
Agent A -> Responsible for 'clean' task.
Agent B -> Responsible for 'inspect' task.

Roles have been assigned:
Agent A -> clean
Agent B -> inspect
```