TECNOLÓGICO NACIONAL DE MÉXICO

Nombre de la Institución: Instituto Tecnológico de Culiacán

Nombre de la Carrera: Ingeniería en Sistemas Computacionales

Materia: Inteligencia Artificial

Jesús Arnoldo Báez Sauceda

Juan Carlos Quiñonez Madrid

17 de Febrero de 2025

Agentes Deliberativos

Definición

Un **agente deliberativo** es una entidad que toma decisiones basadas en un modelo interno del entorno y de sus propias metas. Estos agentes operan en base a principios de planificación y deliberación, utilizando sus conocimientos previos y la información disponible para tomar decisiones informadas. Los agentes deliberativos son parte de la inteligencia artificial (IA) y se caracterizan por su capacidad para evaluar múltiples posibles acciones, predecir las consecuencias de estas acciones y elegir la que maximice la probabilidad de alcanzar sus objetivos.

Los agentes deliberativos suelen estar diseñados para actuar de manera racional y óptima dentro de su entorno, utilizando algoritmos de planificación y toma de decisiones. Estos algoritmos permiten a los agentes evaluar varias alternativas y seleccionar la más adecuada en función de sus metas y restricciones. Además, los agentes deliberativos pueden adaptarse a cambios en el entorno, ajustando sus planes y estrategias según sea necesario.

Ejemplo

Un ejemplo clásico de un agente deliberativo es un **robot de navegación autónoma**. Este robot utiliza un mapa del entorno, junto con algoritmos de planificación de rutas, para decidir la mejor manera de moverse de un punto a otro. El robot considera obstáculos, distancias, y otras restricciones para elegir su camino. A continuación, se detalla cómo un robot de navegación autónoma puede funcionar como un agente deliberativo:

1. **Percepción del entorno**: El robot recopila información sobre su entorno utilizando sensores, como cámaras, láseres, y ultrasonido. Esta información se utiliza para construir un mapa interno del entorno.
2. **Planificación de rutas**: Utilizando el mapa interno, el robot emplea algoritmos de planificación de rutas, como A* o Dijkstra, para calcular la mejor ruta desde su posición actual hasta su destino. Estos algoritmos consideran factores como la distancia, los obstáculos y las restricciones del entorno.
3. **Ejecución del plan**: Una vez que el plan de ruta está definido, el robot sigue la ruta planificada, ajustando su movimiento en tiempo real en respuesta a cambios en el entorno. Por ejemplo, si un obstáculo imprevisto aparece en el camino, el robot reevaluará su ruta y planificará un desvío adecuado.
4. **Adaptación y aprendizaje**: El robot puede aprender de sus experiencias y mejorar su rendimiento con el tiempo. Por ejemplo, si encuentra regularmente obstáculos en una determinada área, puede ajustar su planificación futura para evitar esa área.

Ejemplo de Código

Aquí hay un ejemplo en pseudocódigo de cómo podría funcionar un agente deliberativo para la planificación de rutas:

```
class DeliberativeAgent:
    def __init__(self, environment, start, goal):
        self.environment = environment
        self.current_position = start
        self.goal = goal
        self.plan = []

    def plan_route(self):
        # Implementar un algoritmo de planificación como A*
        open_list = [self.current_position]
        closed_list = []
        came_from = {}

        g_score = {self.current_position: 0}
        f_score = {self.current_position: self.heuristic(self.current_position, self.goal)}

        while open_list:
            current = self.get_lowest_f_score(open_list, f_score)
            if current == self.goal:
                return self.reconstruct_path(came_from, current)

            open_list.remove(current)
            closed_list.append(current)

            for neighbor in self.get_neighbors(current):
                if neighbor in closed_list:
                    continue

                tentative_g_score = g_score[current] + self.distance(current, neighbor)

                if neighbor not in open_list:
                    open_list.append(neighbor)
                elif tentative_g_score >= g_score[neighbor]:
                    continue

                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score[neighbor] = g_score[neighbor] + self.heuristic(neighbor, self.goal)

        return None
```

```
    def heuristic(self, start, goal):
        # Implementar una función heurística
        return abs(start.x - goal.x) + abs(start.y - goal.y)

    def get_lowest_f_score(self, open_list, f_score):
        # Encontrar el nodo con el f_score más bajo en open_list
        return min(open_list, key=lambda pos: f_score[pos])

    def get_neighbors(self, position):
        # Obtener vecinos del nodo actual
        pass

    def distance(self, start, end):
        # Calcular la distancia entre dos nodos
        pass

    def reconstruct_path(self, came_from, current):
        # Reconstruir el camino desde el nodo final hasta el inicio
        total_path = [current]
        while current in came_from:
            current = came_from[current]
            total_path.append(current)
        return total_path[::-1]

# Ejemplo de uso
environment = ...  # Definir el entorno
start = ...        # Definir la posición inicial
goal = ...         # Definir la posición objetivo
agent = DeliberativeAgent(environment, start, goal)
plan = agent.plan_route()
```