Tecnológico Nacional de México, Campus Mexicali

Ingeniería Sistemas Computacionales

IA - Inteligencia Artificial

Tema:

“Tarea - Problema del Agente Viajero”

Estudiante:

Mac Callum Merino Fatima Berenice

No. De control:

C21490774

Profesor:

Oscar Rubén Batista

Mexicali, B.C., 17 de octubre de 2025.

Tabla de contenido

Tabla de imágenes

Implementación del Problema del Agente Viajero (TSP)

Introducción

El Problema del Agente Viajero (TSP, por sus siglas en inglés Traveling Salesman Problem) es uno de los desafíos clásicos de la Inteligencia Artificial y la optimización combinatoria. Consiste en encontrar la ruta más corta que permita a un viajero visitar un conjunto de ciudades exactamente una vez y regresar al punto de partida.

Este trabajo tiene como finalidad aplicar técnicas de búsqueda y optimización para resolver el TSP mediante programación en Python, generando ubicaciones aleatorias para representar casas y mostrando la ruta óptima de manera gráfica.

Objetivo

El objetivo de este trabajo es implementar un programa en Python que resuelva el Problema del Agente Viajero utilizando una estrategia heurística o de búsqueda, generando ubicaciones aleatorias para representar casas y mostrando la ruta óptima mediante una gráfica.

Fundamento teórico

El Problema del Agente Viajero pertenece a la clase de problemas NP-difíciles, lo que significa que no existe un algoritmo conocido que lo resuelva de manera eficiente para todos los casos. Su formulación es sencilla: dado un conjunto de nodos, que representan ciudades o casas, y las distancias entre ellos, se busca un recorrido cerrado de mínima longitud que pase exactamente una vez por cada nodo.

Se puede pensar en el TSP como un grafo completo, donde cada nodo representa una ciudad y cada conexión entre nodos tiene un costo asociado, que corresponde a la distancia entre las ciudades. El objetivo es encontrar un orden de visita de los nodos que minimice la suma total de las distancias recorridas, asegurando que se visite cada ciudad una sola vez y que se regrese al punto de partida.

Para resolverlo, pueden utilizarse métodos exactos, como backtracking o programación dinámica, que garantizan encontrar la solución óptima, o métodos heurísticos, como algoritmos genéticos, recocido simulado o búsqueda local, que aproximan soluciones eficientes de manera más rápida. En este trabajo se emplea un enfoque basado en el **recocido simulado** para aproximar la solución del TSP sobre un conjunto de casas generadas aleatoriamente.

Descripción de actividades

Para implementar la solución se realizaron los siguientes pasos:

1. **Generación de ubicaciones**
   Se generaron de manera aleatoria las coordenadas de las casas dentro de un plano cartesiano utilizando la librería random en Python. Cada casa se representa como un punto con coordenadas (x,y)(x, y)(x,y).
2. **Construcción de la matriz de distancias**
   Se calculó la distancia entre cada par de casas usando la distancia euclidiana, creando una matriz que contiene todas las distancias entre nodos.
3. **Implementación del TSP**
   Se resolvió el TSP usando el algoritmo de **recocido simulado**, que busca aproximar la ruta más corta evaluando iterativamente distintas permutaciones de los nodos y aceptando cambios según un criterio de mejora o probabilidad.
4. **Obtención de la ruta óptima**
   El algoritmo devuelve un orden de visita de los nodos que minimiza la distancia total recorrida y regresa al punto inicial para cerrar el recorrido.
5. **Visualización gráfica**
   Se utilizó la librería matplotlib para graficar las casas y la ruta obtenida. Cada punto representa una casa y las líneas conectan las casas en el orden que sigue el agente viajero, mostrando claramente el recorrido completo.

```python
# tspmain.py
import random
import numpy as np
import matplotlib.pyplot as plt
from python_tsp.heuristics import solve_tsp_simulated_annealing as solve_sa

# ----------------------------
# Funciones auxiliares
# ----------------------------
def generate_houses(n=10, x_range=(0,100), y_range=(0,100)):
    """Genera coordenadas aleatorias de casas para TSP"""
    houses = []
    for _ in range(n):
        x = random.uniform(*x_range)
        y = random.uniform(*y_range)
        houses.append((x, y))
    return houses
```

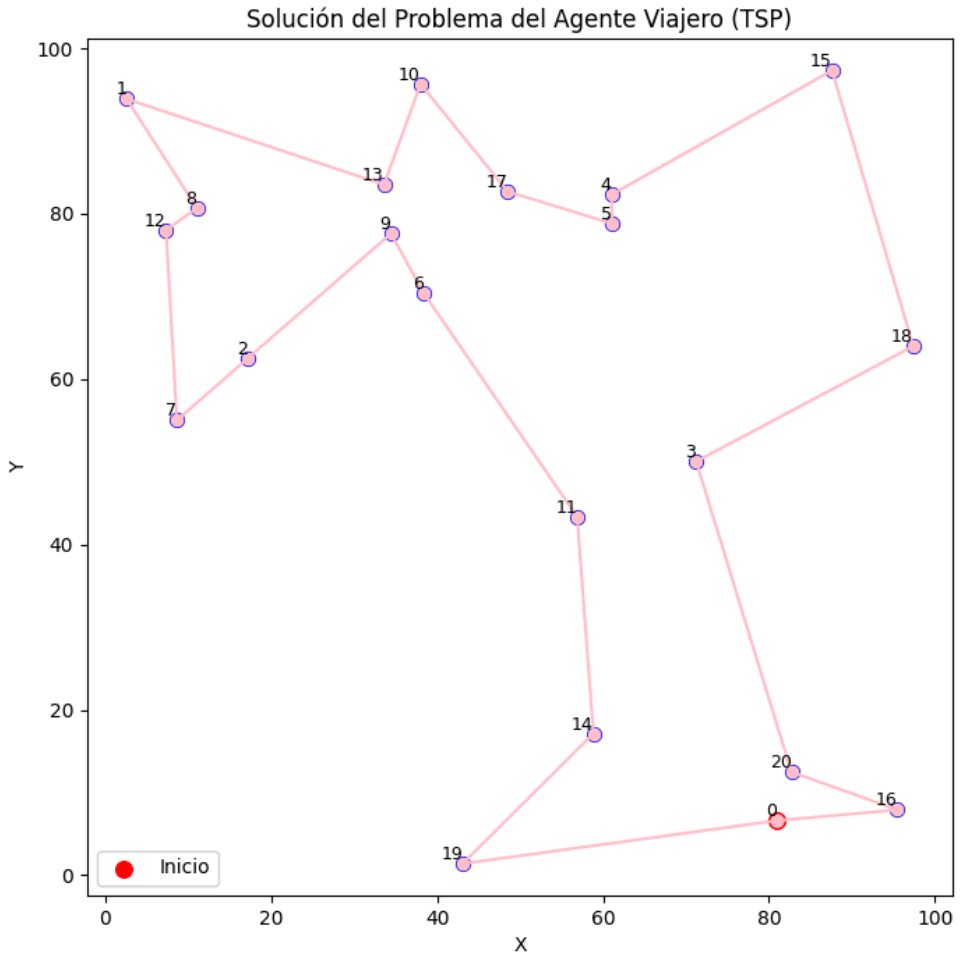
```python
def plot_solution(points, tour):
    """Dibuja las casas y la ruta del TSP"""
    xs = [points[i][0] for i in tour] + [points[tour[0]][0]]
    ys = [points[i][1] for i in tour] + [points[tour[0]][1]]

    plt.figure(figsize=(8,8))
    plt.scatter(*zip(*points), color='blue', s=50)
    plt.scatter(points[tour[0]][0], points[tour[0]][1], color='red', s=70, label="Inicio")

    for i, (x,y) in enumerate(points):
        plt.text(x, y, str(i), fontsize=9, ha='right', va='bottom')

    plt.plot(xs, ys, linestyle='-', marker='o', color='pink')
    plt.title("Solución del Problema del Agente Viajero (TSP)")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.legend()
    plt.axis("equal")
    plt.show()

def build_distance_matrix(points):
    """Construye matriz de distancias euclidianas"""
    n = len(points)
    matrix = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            if i != j:
                matrix[i][j] = np.hypot(points[i][0] - points[j][0],
                                        points[i][1] - points[j][1])
    return matrix
```

```python
# ----------------------------
# Función principal
# ----------------------------
def main():
    # 1. Generar casas
    num_houses = 10
    houses = generate_houses(num_houses, x_range=(0,100), y_range=(0,100))
    print("Casas generadas (x,y):")
    for i, h in enumerate(houses):
        print(f"{i}: {h}")

    # 2. Construir matriz de distancias
    dist_matrix = build_distance_matrix(houses)

    # 3. Resolver TSP
    order, distance = solve_sa(dist_matrix)
    print("\nEl mejor orden de visita es:", order)
    print("Distancia total:", distance)

    # 4. Graficar solución
    plot_solution(houses, order)

# ----------------------------
# Ejecutar main
# ----------------------------
if __name__ == "__main__":
    main()
```

Resultados

- Número de nodos: 21
- Método utilizado: Recocido simulado
- Distancia total de la ruta: Mostrada en la ejecución del programa
- Representación gráfica: Los puntos indican las ubicaciones de las casas, y las líneas muestran la secuencia de visita del agente viajero.

Se generaron 21 casas con posiciones aleatorias, y el algoritmo encontró una ruta cerrada que recorre todas las casas y regresa al punto inicial. La gráfica permite visualizar la ruta y analizar su eficiencia visualmente.



Conclusiones

Este ejercicio permitió comprender la naturaleza del Problema del Agente Viajero y su importancia en la Inteligencia Artificial, la logística y la optimización de rutas. Se comprobó que mediante programación en Python y el uso de librerías como matplotlib, es posible representar gráficamente la solución y analizar su eficiencia visualmente.

El trabajo también reforzó conceptos de representación espacial, heurísticas de búsqueda y optimización, mostrando cómo los algoritmos pueden abordar problemas complejos de planificación. Aunque el TSP no puede resolverse de forma exacta en tiempo razonable para grandes cantidades de nodos, el uso de métodos aproximados permite obtener soluciones útiles en contextos reales como transporte, distribución y planificación de recursos.

Bibliografía

(N.d.). Edu.Mx. Retrieved October 18, 2025, from https://moodle.itmexicali.edu.mx/pluginfile.php/272064/mod_resource/content/1/3%20-%20Optimizacion%20-%20Intro%20IA%20Python%20-CS50.pdf [1]