Clase heurística

Marcelino Sánchez Rodríguez 191654 Samuel Méndez 196659

2024-02-29

Clase Heurística

Contexto

El problema del agente viajero (TSP) es un problema de optimización combinatoria que consiste en encontrar el recorrido más corto que visita un conjunto de ciudades exactamente una vez y al finalizar regresa a la ciudad de origen. Este problema es NP-duro.

En esta clase decidimos encontrar un algoritmo heurístico para resolver el problema del agente viajero, utilizando la idea de empezar en un punto y recorrer el punto más cercano, y así sucesivamente hasta regresar al punto de inicio.

Para utilizar el algoritmo preparamos los datos de las coordenadas de las ciudades en un DataFrame de pandas, y utilizamos la librería numpy para calcular la matriz de distancias entre las ciudades.

```
import numpy as np
import pandas as pd

data = {
    'Posición': [1, 2, 3, 4, 5, 6, 7],
    'x': [1, 9, 5, 3, 2, 11, 6],
    'y': [5, 15, 7, 1, 2, 20, 21]
}

df = pd.DataFrame(data)

print(df)

# Assuming 'df' is your DataFrame
x = df['x'].values
```

```
y = df['y'].values

# Create a 2D array of coordinates
coordinates = np.column_stack((x, y))

# Calculate the distance matrix
dist_matrix = np.sqrt(((coordinates[:, None, :] - coordinates[None, :, :]) ** 2).sum(-1))

# Assuming 'dist_matrix' is your distance matrix
df_dist_matrix = pd.DataFrame(dist_matrix)

# Display the DataFrame
df_dist_matrix
```

```
   Posición   x   y
0         1   1   5
1         2   9  15
2         3   5   7
3         4   3   1
4         5   2   2
5         6  11  20
6         7   6  21
```

	0	1	2	3	4	5	6
0	0.000000	12.806248	4.472136	4.472136	3.162278	18.027756	16.763055
1	12.806248	0.000000	8.944272	15.231546	14.764823	5.385165	6.708204
2	4.472136	8.944272	0.000000	6.324555	5.830952	14.317821	14.035669
3	4.472136	15.231546	6.324555	0.000000	1.414214	20.615528	20.223748
4	3.162278	14.764823	5.830952	1.414214	0.000000	20.124612	19.416488
5	18.027756	5.385165	14.317821	20.615528	20.124612	0.000000	5.099020
6	16.763055	6.708204	14.035669	20.223748	19.416488	5.099020	0.000000

Algoritmo Vecino más cercano

El algoritmo del vecino más cercano es un algoritmo heurístico que empieza en un punto y en cada paso visita el punto más cercano que no ha sido visitado. Este algoritmo es muy sencillo y rápido, pero no garantiza encontrar la solución óptima.

```python
import numpy as np

def vecino_mas_cercano(matriz_distancias, punto_inicio):
    # Número de puntos
    n_puntos = matriz_distancias.shape[0]

    # Lista para guardar el orden de los puntos visitados
    camino = [punto_inicio]

    # Conjunto de puntos no visitados
    no_visitados = set(range(n_puntos))
    no_visitados.remove(punto_inicio)

    # Punto actual
    punto_actual = punto_inicio

    # Recorrer todos los puntos
    while no_visitados:
        # Encontrar el vecino más cercano no visitado
        siguiente_punto = min(no_visitados, key=lambda x: matriz_distancias[punto_actual,

        # Moverse al siguiente punto
        camino.append(siguiente_punto)
        no_visitados.remove(siguiente_punto)
        punto_actual = siguiente_punto

    # Regresar al punto de inicio para completar el ciclo
    camino.append(punto_inicio)

    # Calcular la distancia total del viaje
    distancia_total = sum(matriz_distancias[camino[i], camino[i+1]] for i in range(len(cam

    return camino, distancia_total

# Ejemplo de uso

punto_inicio = 0
camino, distancia_total = vecino_mas_cercano(dist_matrix, punto_inicio)

print(f"Camino: {camino}")
print(f"Distancia total: {distancia_total}")
```

```
Camino: [0, 4, 3, 2, 1, 5, 6, 0]
Distancia total: 47.09255738784489
```

Ahora que tenemos el camino y la distancia total, podemos graficar el recorrido del agente viajero.

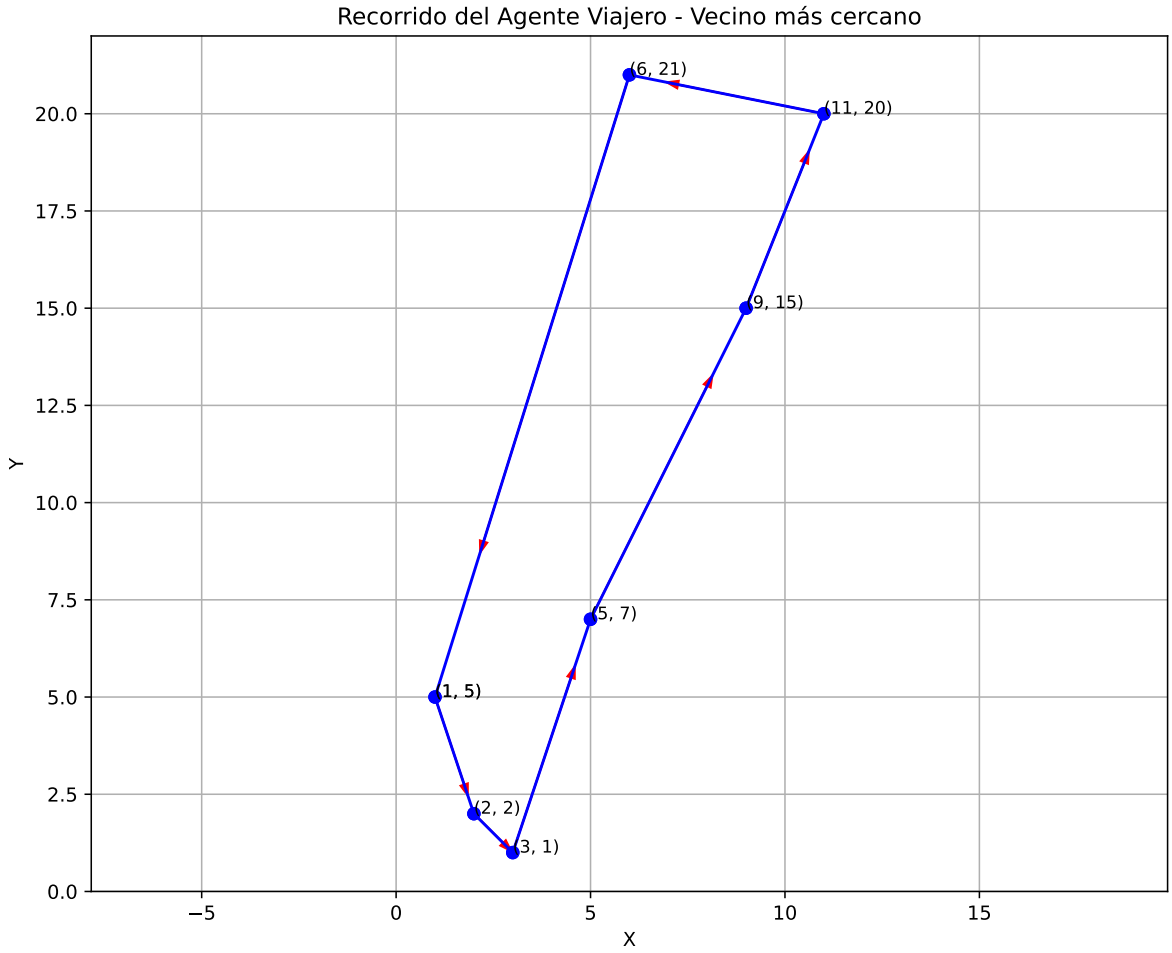
```python
import matplotlib.pyplot as plt

# Coordenadas de los puntos (ejemplo)
coordenadas = coordinates

# Graficar el recorrido
plt.figure(figsize=(10, 8))
for i in range(len(camino)-1):
    punto_a = coordenadas[camino[i]]
    punto_b = coordenadas[camino[i+1]]
    plt.plot([punto_a[0], punto_b[0]], [punto_a[1], punto_b[1]], 'bo-') # Líneas entre pun
    # Mostrar coordenadas exactas del punto A
    plt.text(punto_a[0], punto_a[1], f'({punto_a[0]}, {punto_a[1]})', fontsize=9)
    # Dibujar flecha más grande de A a B
    dx = punto_b[0] - punto_a[0]
    dy = punto_b[1] - punto_a[1]
    plt.arrow(punto_a[0], punto_a[1], dx * 0.75, dy * 0.75, head_width=0.2, head_length=0.

# Mostrar último punto con sus coordenadas exactas
ultimo_punto = coordenadas[camino[-1]]
plt.text(ultimo_punto[0], ultimo_punto[1],
         f'({ultimo_punto[0]}, {ultimo_punto[1]})', fontsize=9)

plt.title('Recorrido del Agente Viajero - Vecino más cercano')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.axis('equal')  # Esto asegura que se muestre a escala
plt.show()
```


Recorrido del Agente Viajero - Vecino más cercano
(6, 21)
(11, 20)
(9, 15)
(5, 7)
(1, 5)
(2, 2)
(3, 1)
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Y
−5
0
5
10
15
X