

IA_PRACTICA_01

Problema del viajante mediante el algoritmo Hill Climbing

IA

Esta práctica consiste en resolver el problema del viajante de negocios, que debe visitar 'n' ciudades una y sólo una vez, buscando la ruta más corta, para invertir menos tiempo y gastar menos combustible.



Para resolver el problema se realiza la optimización del camino más corto utilizando el algoritmo Hill Climbing.

SOLUCIÓN

Definir las librerías a utilizar

```
# Importar librerías
import random
from PIL import Image, ImageDraw, ImageFont
from math import sqrt
```

Función para generar de forma aleatoria todos los pares i,j desde 0 hasta size

```
def all_pairs(size, shuffle=random.shuffle):
    r1=range(size)
    r2=range(size)
    if shuffle:
        shuffle(list(r1))
        shuffle(list(r2))
    for i in r1:
        for j in r2:
            yield (i,j)
```

Función para generar todas las posibles variaciones cuando se produce el intercambio entre dos ciudades

```
def reversed_sections(tour):
    for i,j in all_pairs(len(tour)):
        if i != j:
            copy=tour[:]
            if i < j:
```

```
        copy[i:j+1]=reversed(tour[i:j+1])
    else:
        copy[i+1:]=reversed(tour[:j])
        copy[:j]=reversed(tour[i+1:])
    if copy != tour: # ningún punto devuelve el mismo tour
        yield copy
```

Función para crear todas las posibles variaciones cuando dos ciudades se han intercambiado

```
def swapped_cities(tour):
    for i,j in all_pairs(len(tour)):
        if i < j:
            copy=tour[:]
            copy[i],copy[j]=tour[j],tour[i]
            yield copy
```

Función para crear la matriz de distancias (considerando la línea recta entre ciudades)

```
def cartesian_matrix(coords):
    matrix={}
    for i,(x1,y1) in enumerate(coords):
        for j,(x2,y2) in enumerate(coords):
            dx,dy=x1-x2,y1-y2
            dist=sqrt(dx*dx + dy*dy)
            matrix[i,j]=dist
    return matrix
```

Función para leer las coordenadas del fichero de entrada

```
# Las coordenadas deben estar en pares x,y por línea y separadas por una coma
def read_coords(coord_file):
    coords=[]
    with open(coord_file) as f:
        content = f.readlines()
    for line in content:
        x,y=line.strip().split(',')
        coords.append((float(x),float(y)))
    return coords
```

Función para crear la longitud total del tour basado en la matriz de distancias

```
def tour_length(matrix,tour):
    total=0
    num_cities=len(tour)
    for i in range(num_cities):
        j=(i+1)%num_cities
        city_i=tour[i]
        city_j=tour[j]
        total+=matrix[city_i,city_j]
    return total
```

Función para crear una imagen de representación de las ciudades y el tour seleccionado

```
def write_tour_to_img(coords,tour,title,img_file):
    padding=20
    coords=[(x+padding,y+padding) for (x,y) in coords]
    maxx,maxy=0,0
    for x,y in coords:
        maxx=max(x,maxx)
        maxy=max(y,maxy)
    maxx+=padding
    maxy+=padding
    img=Image.new("RGB", (int(maxx),int(maxy)),color=(255,255,255))
```

```
font=ImageFont.load_default()
d=ImageDraw.Draw(img);
num_cities=len(tour)
for i in range(num_cities):
    j=(i+1)%num_cities
    city_i=tour[i]
    city_j=tour[j]
    x1,y1=coords[city_i]
    x2,y2=coords[city_j]
    d.line((int(x1),int(y1),int(x2),int(y2)),fill=(0,0,0))
    d.text((int(x1)+7,int(y1)-5),str(i),font=font,fill=(32,32,32))

for x,y in coords:
    x,y=int(x),int(y)
    d.ellipse((x-5,y-5,x+5,y+5),outline=(0,0,0),fill=(196,196,196))

d.text((1,1),title,font=font,fill=(0,0,0))

del d
img.save(img_file, "PNG")
```

Función para inicializar el tour de forma aleatoria

```
def init_random_tour(tour_length):
    tour=list(range(tour_length))
    random.shuffle(tour)
    return tour
```

Función para ejecutar el algoritmo de Hill Climbing

```
def run_hillclimb(init_function,move_operator,objective_function,max_iterations):
    from hillclimb import hillclimb_and_restart

iterations,score,best=hillclimb_and_restart(init_function,move_operator,objective_function,max_ite
erations)
    return iterations,score,best
```

PROGRAMA PRINCIPAL

```
out_file_name="ruta_resultado.png"
max_iterations=10000
verbose=True
move_operator=reversed_sections
arg="swapped_cities"
#arg="reversed_sections"
```

Leer el fichero con las coordenadas de las ciudades

```
# Fichero con las ccordenadas de las ciudades
city_file="city100.txt"

# Lectura de las coordenadas de las ciudades
coords=read_coords((city_file))
```

Inicializar de forma aleatoria la ruta

```
# Inicializar de forma aleatoria la ruta
init_function=lambda: init_random_tour(len(coords))
```

Calcular la matriz de distancias

```
# Calcular la matriz de distancias
```

```
matrix=cartesian_matrix(coords)
```

Calcular la longitud de las rutas

```
# Calcular la longitud de las rutas
objective_function=lambda tour: -tour_length(matrix,tour)
```

Ejecutar el algoritmo Hill Climbing

```
# ejecutar el algoritmo de Hill Climbing
iterations,score,best=run_hillclimb(init_function,move_operator,objective_function,max_iterations
)
```

Visualizar los resultados

```
print("Iteraciones: ",iterations)
print("Puntuación: ",score)
print("Ruta seleccionada entre ciudades:")
print(best)
```

Guardar una imagen con la mejor ruta seleccionada

```
# Guardar una imagen de la ruta seleccionada
write_tour_to_img(coords,best,'%s: %f %s'%(city_file,score,arg),out_file_name)
```

```
2018-12-18 12:17:27,021 INFO (re)iniciando hillclimb 10000/10000 restante
2018-12-18 12:17:27,021 INFO Hillclimb comenzado: score=-26192.714184
Reloaded modules: hillclimb
2018-12-18 12:17:27,411 INFO hillclimb finalizado: num_evaluations=10000, best_score=-
15604.256046
Iteraciones: 10000
Puntuación: -15604.25604616874
Ruta seleccionada entre ciudades:
[24, 17, 29, 89, 40, 7, 42, 85, 80, 1, 75, 58, 6, 74, 11, 26, 8, 19, 61, 35, 5, 56, 88, 36, 49,
32, 76, 72, 12, 95, 55, 43, 13, 64, 53, 71, 47, 25, 98, 4, 94, 90, 48, 63, 97, 41, 52, 34, 10,
14, 31, 50, 77, 66, 60, 39, 9, 99, 69, 30, 82, 87, 86, 62, 3, 18, 51, 68, 65, 15, 57, 78, 23, 67,
96, 79, 20, 83, 38, 22, 28, 16, 21, 45, 54, 59, 92, 0, 44, 73, 84, 37, 2, 93, 91, 27, 70, 46, 81,
33]
```

city100.txt: -15604.256046

