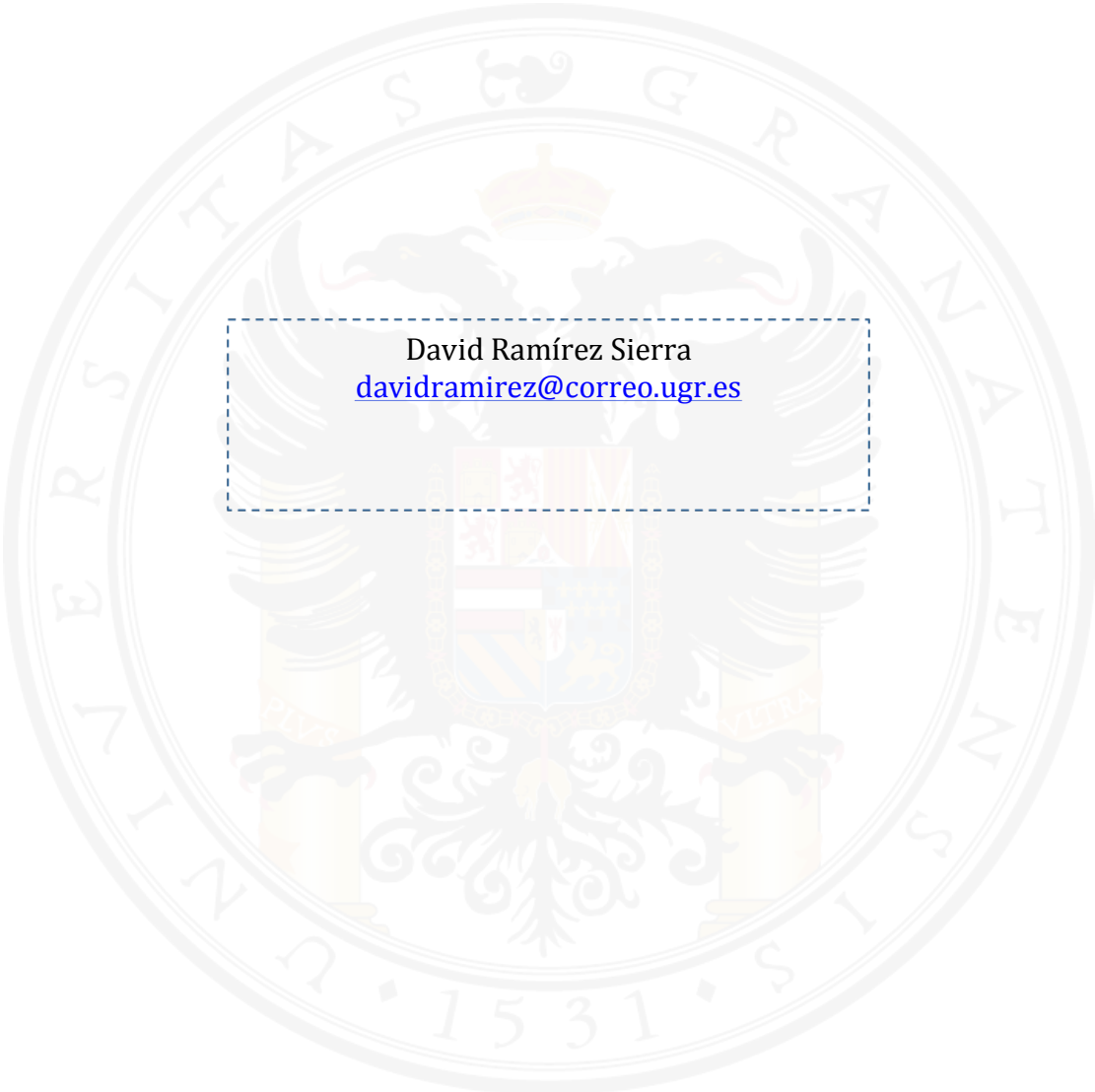
PRÁCTICA 4

EL PROBLEMA DEL VIAJANTE DE COMERCIO

INFORME.



David Ramírez Sierra
davidramirez@correo.ugr.es

Table of Contents

1. INTRODUCCIÓN

El problema del viajante de comercio:

El objetivo de esta práctica es que el alumno aplique los contenidos de los primeros temas de la asignatura (introducción a la programación dirigida a objetos, arrays y clases) para la resolución de un problema concreto: el problema del viajante de comercio (TSP, por Travelling Salesman Problem).

En términos sencillos, el TSP se define como sigue: dado un conjunto de ciudades y una matriz con las distancias entre todas ellas, un viajante debe recorrer todas las ciudades exactamente una vez, regresando al punto de partida, de forma tal que la cantidad de kilo´metros recorridos sea mínima. Más formalmente, dado un grafo G, conexo y ponderado, y dado uno de sus vértices v_0, encontrar el ciclo hamiltoniano3 de mínimo costo que comienza y termina en v_0.

Este problema es uno de los más relevantes en la clase de los NP-Completos y encuentra aplicación práctica en herramientas relacionadas con transporte, logística y también en la industria electrónica. Por ejemplo, consideremos el brazo de un robot encargado de soldar las conexiones de un circuito integrado: el camino más corto que permite visitar cada punto a soldar, es el recorrido más eficiente para el robot. Una aplicación similar aparece cuando se desea minimizar el tiempo que utiliza un plotter para dibujar una figura.

Se denomina “instancia” del TSP a un conjunto particular de ciudades. Una solución (una “ruta”) para una instancia es una permutación del con- junto de ciudades que indica el orden en que se deben recorrer. En el cálculo de la longitud del recorrido no se debe olvidar sumar la distancia que existe entre la última ciudad y la primera; es decir, se debe cerrar el ciclo.

Por su interés teórico y práctico, existe una variedad muy amplia de algoritmos para abordar la solución del TSP y sus variantes. Siendo un problema NP-Completo, el diseño y aplicación de algoritmos exactos para su resolución no es factible en todos los casos. En la actualidad, se pueden resolver problemas de aproximadamente 15000 ciudades de forma exacta, pero para llegar a estos valores, se han requerido años de estudio en las áreas de investigación operativa, estructura de datos, arquitecturas paralelas de cómputo, etc.

En este guión, nos centraremos en una serie de algoritmos aproximados de tipo Greedy y evaluaremos su rendimiento en un conjunto de instancias del TSP. Por lo general estos algoritmos no encuentran la solución óptima del problema, pero son capaces de obtener soluciones razonablemente buenas en tiempo reducido.

El vecino más cercano:

En este guión, se abordará la resolución del problema utilizando un método heurístico denominado “el vecino más cercano” cuyo funcionamiento es extremadamente simple: dada una ciudad inicial v0, se agrega como ciudad

siguiente aquella vi (no incluida en la ruta) que se encuentre más cercana a v0. El procedimiento se repite hasta que todas las ciudades se hayan incluido.

En la implementación solicitada, el método debe aplicarse desde cada posible ciudad inicial y el resultado final será el recorrido de menor longitud. Al final de la ejecución del programa, el programa debe haber calculado el orden en que se recorren las ciudades siguiendo la heurística del vecino más cercano. La solución obtenida podrá mostrarse en la salida estándar según se indica más adelante. El texto generado en la salida podría guardarse en un fichero de texto, y usarse para generar gráficos.

Para resolver el problema se propone la implementación de 4 clases cuyos nombres y funcionalidades básicas se describen a continuación.

2. DIAGRAMA DE CLASES EN UML

Enunciado:

Ciudad

Permite almacenar un par (x, y) (asociado a la posición de una ciudad). Cada ciudad tiene asociada una etiqueta (un número entero que va de 1 al número de ciudades). Además, dada una ciudad, tiene que permitir calcular la distancia euclídea a otra ciudad.

Ruta

Esta clase representa una solución del problema. La solución se define a través de una permutación de ciudades, o sea, un orden en el que se visitara´n las ciudades. Debe permitir la posibilidad de representar soluciones parciales al problema, es decir cuando no todas las ciudades este´n incluidas en la ruta todavía. La ciudad en el orden número 0 representa la ciudad de partida, la 1 la segunda en visitarse y así sucesivamente

Problema

Esta clase permitirá almacenar:

- El tamaño del problema
- La lista de ciudades
- La matriz de distancias entre ellas

La clase Problema debe disponer de un método que lea los datos de un problema de la entrada estándar.

Heurística

En esta clase se incluirán los métodos que resuelvan el problema, mediante el cálculo de un objeto Ruta. Dispondrá de un método que calcule un objeto Ruta con el resultado de resolverlo utilizando la heurística del vecino más cercano.
También tendrá un método para recuperar la solución calculada(objeto Ruta) por esta heurística.
La clase dispondrá también de un método que devuelva el coste de la solución obtenida.