PRÁCTICA 3 – ALGORITMO DE PLANIFICACIÓN

Algoritmo para el TSP

Rubén Morante González
Ruben.morante@urjc.es

Tabla de contenido

Tabla de ilustraciones

INTRODUCCIÓN

Se requiere implementar un algoritmo metaheurístico para solucionar el problema del viajero de comercio TSP. Para ello se implementará un algoritmo GRASP cuyo constructor será un algoritmo RANDOM y un algoritmo de mejora basado en una búsqueda local de intercambiar nodos.

Se resolverá una instancia de nodos de la zona de Móstoles (Madrid).

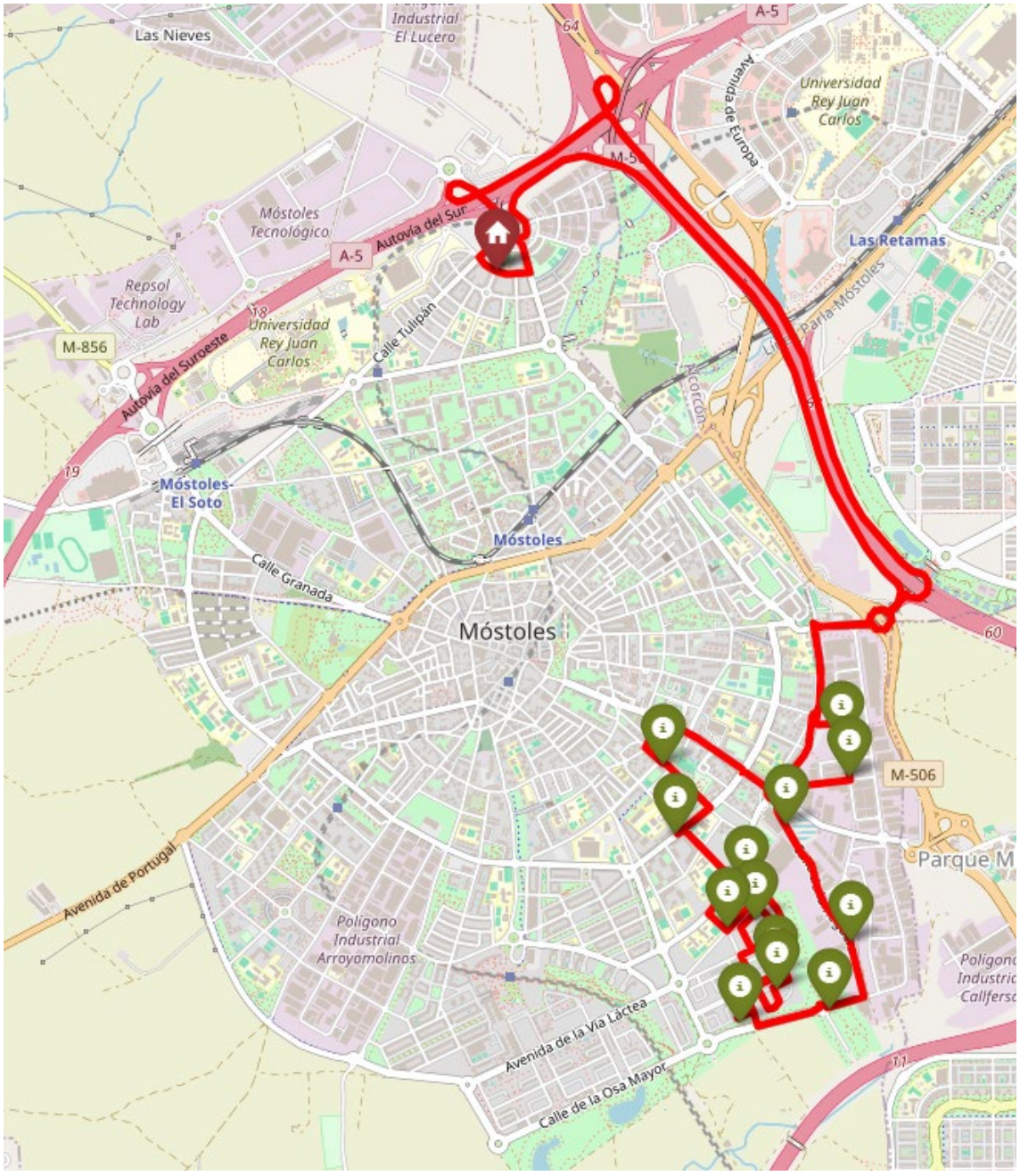


Ilustración 1 Ruta con pocos nodos

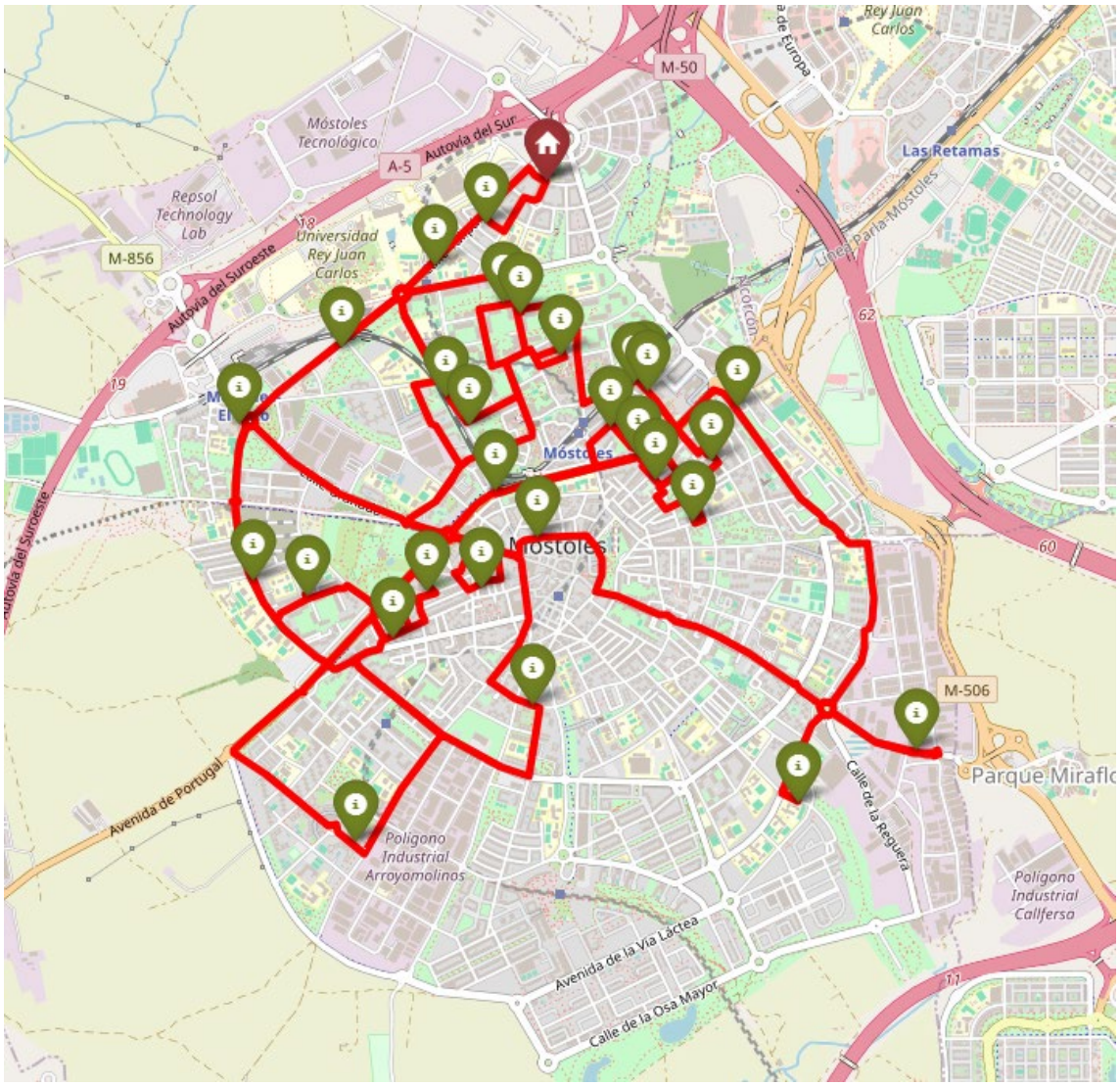


Ilustración 2 Ruta con muchos nodos

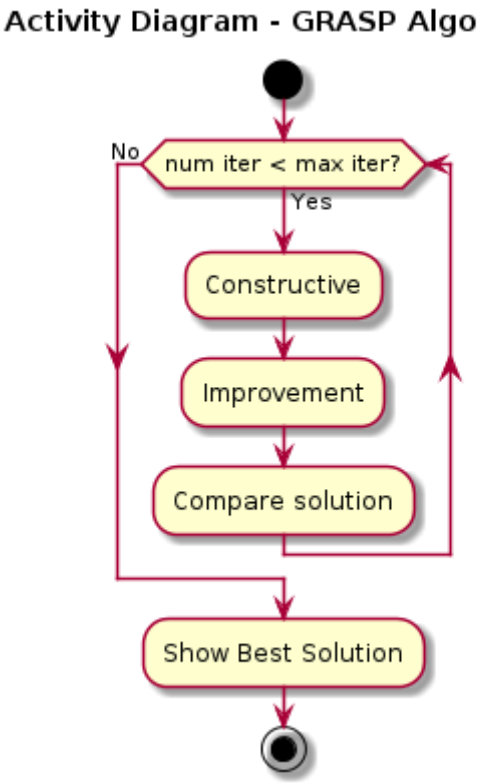
Diagramas de actividad

El flujo del algoritmo es el siguiente:

Se itera N veces. Por cada iteración se construye una solución mediante el algoritmo RANDOM. Esta solución se mejora mediante el algoritmo de búsqueda local y por último se compara con la mejor solución conseguida hasta el momento. Siempre mantendremos la mejor solución almacenada.

Una vez iterado N veces se muestra la mejor solución encontrada.

Lo descrito anteriormente se muestra en el diagrama de actividad:

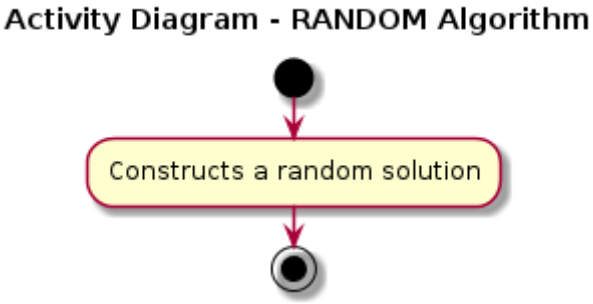


La parte del constructor se realizará mediante un algoritmo RANDOM que lo único que hará sera construir una solución aleatoria:

Ej:

Nodos en la instancia: [1, 2, 3, 4, 5]

Nodos en la solución: [4, 3, 5, 1, 2]



La parte de mejora se realizará mediante una búsqueda local intercambiando dos nodos en la solución.

Ej:

Nodos en la solución inicial: [3, 4, 1, 7, 5, 2, 8, 6]

Nodos en la solución resultante: [3, 5, 1, 7, 4, 2, 8, 6]

Cuando encuentre un movimiento que mejore la solución volverá a empezar desde el principio intentando intercambiar nodos. Si compara todos los movimientos sin que mejore la solución se detendrá.

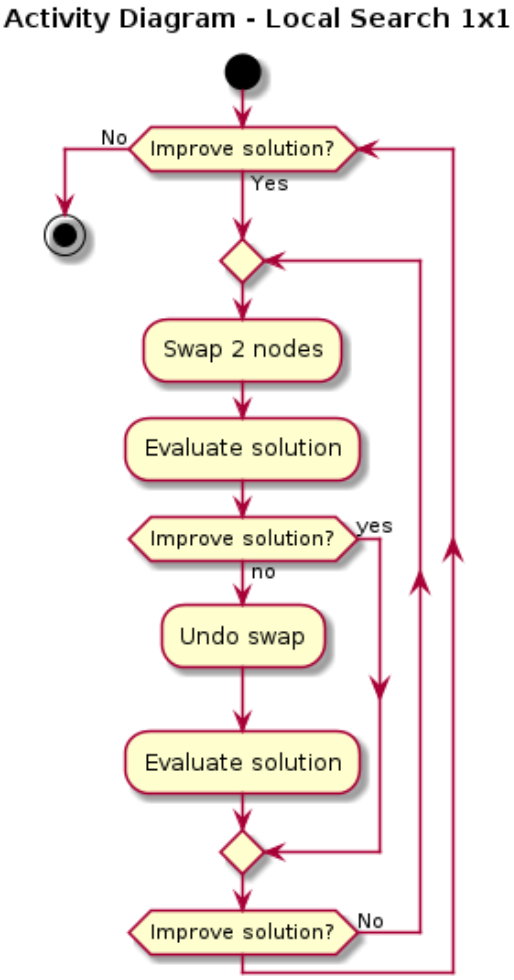
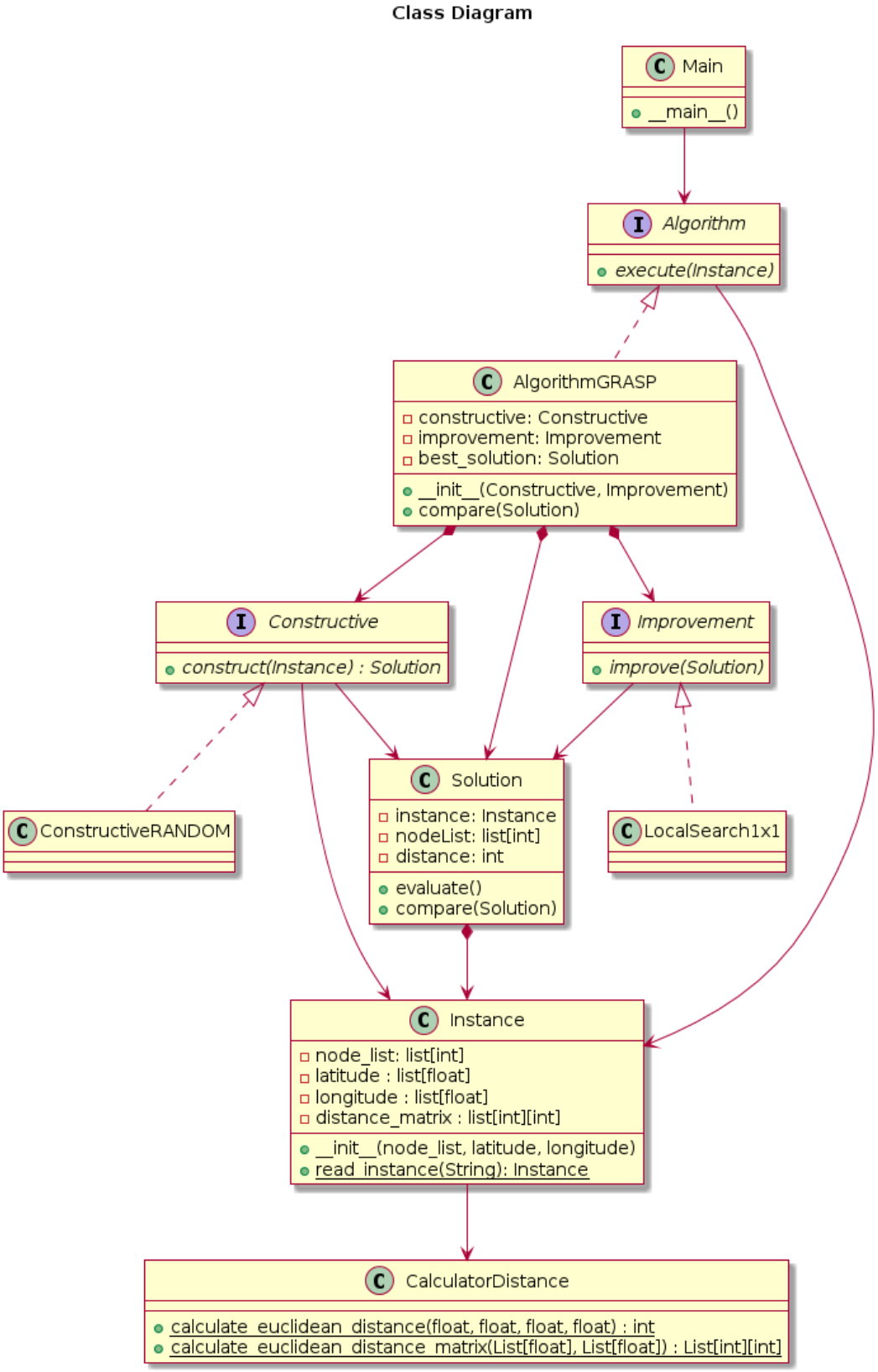


Diagrama de clases

El diagrama de clases del proyecto será el siguiente:



Clases:

Main: Clase lanzadora que tendrá la función __main__ desde donde se ejecutará el algoritmo.

Algorithm: Clase abstracta o interfaz que declara el método execute, el cuál es la única forma de interactuar con el algoritmo.

AlgorithmGRASP: Clase hija que hereda de Algorithm. Implementa el método execute, definiendo la lógica de la metaheurística GRASP.

Constructive: Clase abstracta o interfaz que declara el método construct, donde irá la lógica del algoritmo constructivo.

ConstructiveRANDOM: Clase hija que hereda de Constructive. Implementa el método construct con la lógica del algoritmo de construcción.

Improvement: Clase abstracta o interfaz que declara el método improve, donde irá la lógica del algoritmo de mejora.

LocalSearch1x1: Clase hija que hereda de Improvement. Implementa el método improve con la lógica del algoritmo de mejora de intercambiar un elemento por otro de la solución.

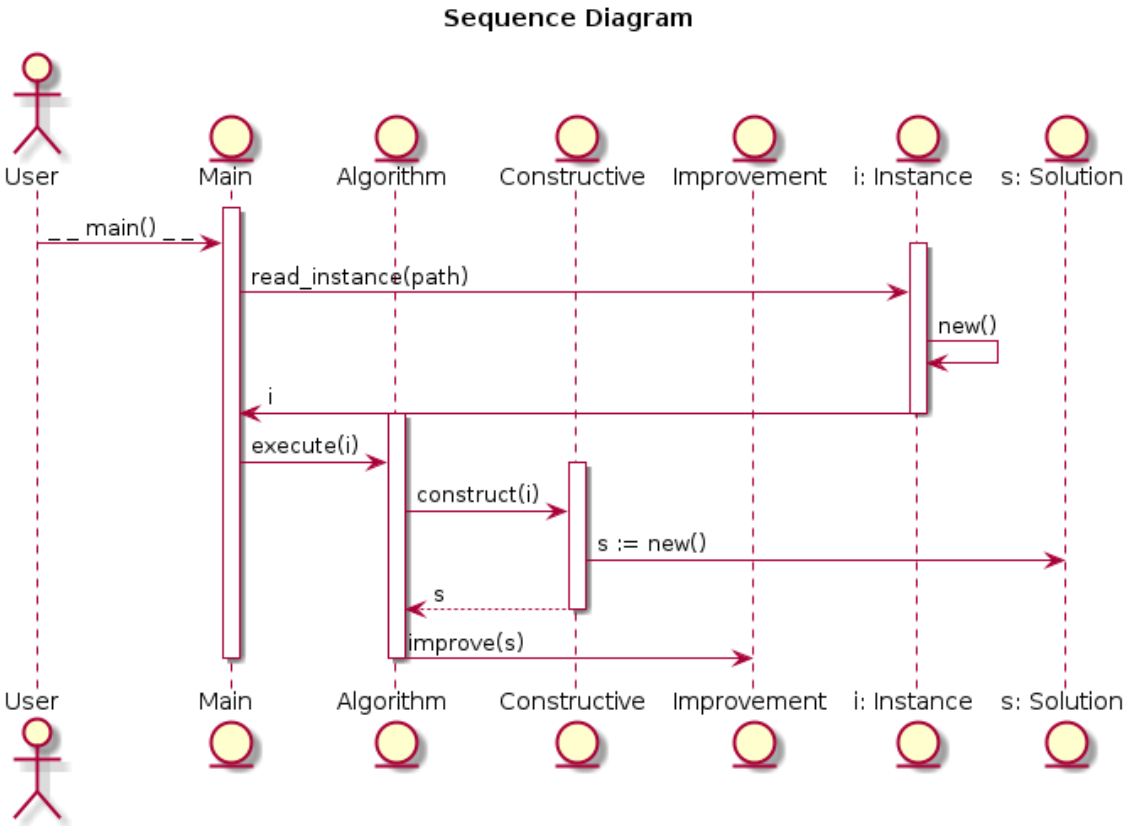
Solution: Clase que almacena la secuencia de nodos final de la ruta y la distancia recorrida.

Instance: Clase que almacena los datos del problema, los nodos por los que debe pasar la ruta, la latitud y la longitud de los nodos y la matriz de distancias.

CalculatorDistance: Clase funcional con las utilidades relacionadas al cálculo de distancias.

Diagrama de secuencia

La secuencia completa del algoritmo viene representada por el siguiente diagrama:



Explicación del flujo del algoritmo

En la clase Main, en la función __init__ se instanciará el objeto Instance con los datos necesarios para el problema, para ello llamaremos a Instance.read_instancer() pasándole como parámetro la ruta del fichero donde vienen los nodos con sus coordenadas. Esta función llamará al constructor de Instance, al que le pasará los nodos, las latitudes y la longitudes, con esta información llamará a la clase CalculatorDistance para calcular la matriz de distancias.

Una vez instanciado el objeto Instance, instanciaremos un objeto Algorithm, en concreto de AlgorithmGRASP y le pasaremos como parámetros al constructor un objeto ConstructiveRANDOM y otro LocalSearch1x1.

Teniendo el objeto de tipo Algorithm llamaremos al método execute() pasándole como parámetro la instancia.

Dentro de este método estará la lógica del algoritmo, de forma simplificada, un bucle de N iteraciones que crea una solución mediante el algoritmo ConstructiveRANDOM, y luego mediante el algoritmo LocalSearch1x1 se mejora, y por último se llama a un método compare() que almacena la mejor solución encontrada hasta el momento.

Por último, se debe mostrar la mejor solución encontrada.

NORMAS DE LA ENTREGA

Las prácticas se realizarán en grupos de 3 o 4 personas.

La entrega se realizará en Aula Virtual, y sólo la entregará un miembro del grupo.

Se deben entregar dos cosas:

1. El proyecto con el código del algoritmo.
2. Un documento donde se especifiquen los miembros del grupo.