Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
C12693 - Laboratorio de Algoritmos y Estructuras III
Trimestre septiembre-diciembre 2024

Proyecto #2

Fecha de entrega: semana 12
miércoles 11-nov-2024, 08:30 am
Valor: 35%

INTRODUCCIÓN / PLANTEAMIENTO

Se ha hablado en el laboratorio de las metaheurísticas para resolver el problema del TSP, y una de ellas, los algoritmos genéticos (AG), será el tema subyacente a este proyecto.

Sobre un banco de prueba de SIETE tests (5, 10, 15, 42, 52, 100 y 200 ciudades), la idea es comparar los resultados de las heurísticas de mejor vecino, inserción 1 e inserción 2 con una implementación "quick and dirty" de un algoritmo genético.

Para ello se le proveerá de un programa Kotlin monolítico (en un solo archivo) Main.kt que tiene una pequeña interfaz de evaluación. **Recuerde que el propósito del proyecto es los algoritmos subyacentes a AG, no la interfaz**.

El esquema de programación sigue las pautas de IntelliJ Idea, que es un entorno de desarrollo mucho más sofisticado que Visual Studio Code, y ofrece más herramientas para el desarrollo en Kotlin. No obstante, el programa corre en Visual Studio Code, sin mayores inconvenientes, solo hay que ajustar las dependencias en build.gradle.kts para enlazar el uso de ***id("org.jetbrains.kotlinx.dataframe") version "0.14.1"*** que es un utilitario que permite el uso de dataframes para reportes (similar al Pandas en Python).

Se describe puntos importantes del archivo ***Main.kt*** que permita su fácil comprensión:

((1)) En la parte superior hay una constante ***TSP_DIR*** que indica el directorio donde están los tests (archivos TXT), ajustar conforme se requiera.

((2)) ***Clase App***. Es una clase que es implementada por IntelliJ Idea cuando selecciona un proyecto tipo aplicación. Se aprovechó dicha clase para generar una interfaz para seleccionar los archivos de tests y preguntar si la ejecución del programa se acompañará de "gráficos".

((3)) ***Clase MyChart***. Es una interfaz para poder presentar gráficos sencillos en ventanas.

((4)) ***Clase Point***. Es una interfaz para representar puntos (X,Y) en un hipotético primer cuadrante de un plano cartesiano 2D, y permite el calculo de distancia entre dos puntos (distancia euclidiana "clásica").

((5)) ***Clase TspTest***. Es una clase que lee los archivos de tests que tiene un formato fijo, indicando las coordenadas de las ciudades (primer cuadrante de un plano cartesiano 2D), tiene la información del óptimo, del número de ciudades, etc. Construye la matriz de distancia y un vector de puntos que son requeridos para las operaciones con los diferentes algoritmos y para las interfaces gráfica.

((6)) ***Clase Tour***. Es una clase que soporta el concepto de "ruta" en el TSP, representada por una lista de enteros. Tiene atributos de distancia y fitness.

((7)) ***Alias Individuo***. Un typealias de Tour, que permite que usemos la terminología de Individuo en los algoritmos genéticos.

((8)) ***Clase TspToolBox***. Aquí es donde se implementan las heurísticas (fuerza bruta, mejor vecino, inserción 1 e inserción 2) y la metaheurística del algoritmo genético.

((9) **fun main()**. Se enlazan los diferentes elementos para construir un simple mini entorno de ejecución de experimentos.

PETITORIO

Ud. deberá completar el código en las partes señaladas para que algoritmo genético sea funcional, no se preocupe si los resultados al compararlos con las heurísticas son desalentadores. Los algoritmos genéticos requieren mucha entonación, pruebas, ajustes, etc.

Ud. deberá entregar el archivo ***Main.kt***, y se le pide que las programaciones vayan en los sitios donde se solicita. En el código encontrará directrices a ser tomadas en cuenta, como comentarios.