Informe de práctica del Problema de Rutas de Vehículos (VRP)

ENRIQUE VIÑA ALONSO

Mayo 2022

1. Introducción

El problema de enrutamiento de vehículos (VRP, por sus siglas en inglés) es un problema de optimización combinatoria que pregunta cuál es el conjunto óptimo de rutas para una flota de vehículos que debe de satisfacer las demandas de un conjunto dado de clientes.

En nuestro caso, añadiremos la restricción de que la ruta para cada vehículo tendrá un número máximo de clientes visitados dado por la siguiente expresión:

$(N/K) + 0,1N$

Dónde N corresponde al número de clientes y K corresponde a el número de vehículos disponibles.

2. Descripción del código

2.1. Problem

Hemos creado una clase 'Problem', que será la encargada de extraer la información almacenada en los ficheros de datos. Se almacena el nombre del fichero, así como el número de clientes y la matriz de distancias entre cada par de clientes.

2.2. Solution

Para representar las soluciones, se hace uso de una estructura de clases simple, con una clase abstracta 'Solution' que extenderán cada una de las soluciones para los diferentes algoritmos.

Las soluciones se almacenarán en una lista, que tendrá K elementos, dónde K es el número de vehículos, cada uno de estos elementos será otra fila, correspondiendo a la ruta para cada uno de los vehículos.

Por ejemplo, 'GraspSolution' añade un nuevo atributo 'rclSize' para almacenar el tamaño de la Restricted Candidate List empleado para resolver el problema.

2.3. Algoritmos desarrollados

Se han implementado tres algoritmos y cuatro búsquedas de entorno, que se detallarán a continuación:

2.3.1. GreedyWithRCL

'GreedyWithRCL' es una clase que implementa un algoritmo voraz con Restricted Candidate List. Para resolver el problema de forma puramente voraz, el tamaño de la RCL está asignado a 1 por defecto.

2.3.2. GRASP

La clase 'GRASP' implementa un algoritmo GRASP, el método 'Solve' toma como parámetro el tamaño de la RCL que tendrá la fase constructiva y un enum 'GraspType' usado para identificar la búsqueda de entorno que se usará en la fase de búsqueda local.

2.3.3. GVNS

La clase 'GVNS' implementa un algoritmo GVNS, el método 'Solve' toma como parámetro el tamaño de la RCL que se usará para inicializar la primera solución. El proceso de VND se realiza con las 4 búsquedas de entorno que se han implementado, el orden es el siguiente:

- Reinserción multirruta El movimiento realizado es tomar elementos de una ruta uno a uno e insertarlo en cada posición posible en el resto de rutas.
- Reinserción intrarruta El movimiento realizado consiste en, para cada ruta tomar elementos de uno a uno e insertarlo en cada posición posible dentro de la misma ruta.
- Intercambio multirruta El movimiento consiste en realizar todas las posibles combinaciones de intercambios de dos elementos que están en rutas diferentes.
- Intercambio intrarruta El movimiento consiste en, para cada ruta, hacer todos los intercambios posibles de pares de elementos dentro de la misma ruta.

3. Resultados

A continuación se presentan las mejores soluciones encontradas por los algoritmos implementados:

3.1. Algoritmo voraz

	Tamaño	Coste	Tiempo
40-2	40 (2)	227	3 ms
40-4	40 (4)	281	0 ms
40-6	40 (6)	352	0 ms
40-8	40 (8)	484	0 ms

Cuadro 1: Mejores resultados del algoritmo voraz.

3.2. Algoritmos GRASP

Resultados de los algoritmos GRASP:

	Tamaño	RCL	Coste	Tiempo
40-2	40 (2)	2	116	270 ms
40-4	40 (4)	2	155	310 ms
40-6	40 (6)	2	190	295 ms
40-8	40 (8)	2	240	960 ms

Cuadro 2: Mejores resultados del algoritmo GRASP con reinserción multirruta como búsqueda local.

	Tamaño	RCL	Coste	Tiempo
40-2	40 (2)	2	128	249 ms
40-4	40 (4)	2	164	163 ms
40-6	40 (6)	2	221	70 ms
40-8	40 (8)	2	305	74 ms

Cuadro 3: Mejores resultados del algoritmo GRASP con reinserción intrarruta como búsqueda local.

	Tamaño	RCL	Coste	Tiempo
40-2	40 (2)	2	149	99 ms
40-4	40 (4)	2	179	140 ms
40-6	40 (6)	2	248	49 ms
40-8	40 (8)	2	307	79 ms

Cuadro 4: Mejores resultados del algoritmo GRASP con intercambio multirruta como búsqueda local.

	Tamaño	RCL	Coste	Tiempo
40-2	40 (2)	2	141	77 ms
40-4	40 (4)	2	183	99 ms
40-6	40 (6)	2	244	115 ms
40-8	40 (8)	2	314	43 ms

Cuadro 5: Mejores resultados del algoritmo GRASP con intercambio intrarruta como búsqueda local.

3.3. Algoritmo GVNS

Resultados del Algoritmo GVNS:

	Tamaño	Coste	Tiempo
40-2	40 (2)	101	206 ms
40-4	40 (4)	134	359 ms
40-6	40 (6)	171	158 ms
40-8	40 (8)	175	395 ms

Cuadro 6: Mejores resultados del algoritmo GVNS.

4. Conclusiones

Comparando los resultados obtenidos, podemos ver claramente que el algoritmo GVNS es mucho más eficiente que el resto, ya que es capaz de hallar las soluciones de mayor calidad entre los algoritmos implementados sin comprometer enormemente el tiempo de ejecución.