Problema de rutas de vehiiculos (VRP)

Ayob asrout Vargas

2 de mayo de 2022

Resumen

Universidad de La Laguna.
Diseño y Analisis de Algoritmos.

1. Introduccion

En el problema de rutas de vehiculos (VRP) hay un conjunto de clientes V que hay que visitar y un conjunto de vehiculos K para realizar las tareas. Los clientes tienen que ser visitados todos, una sola vez. El numero de vehiculos disponibles será dado y se generarán tantas rutas como vehiculos haya disponibles. El objetivo de este problema es visitar a todos los clientes minimizando la distancia total recorrida. En el ejemplo de la figura 1, la distancia total recorrida de la ruta 1 es igual a 25, la de la ruta 2 es 22 y la de la ruta 3 es 19. Por tanto, el valor objetivo de esta solucion es la suma de estos tres valores (25 + 22 + 19 = 66).

2. Algoritmos diseñados

A continuacion pasamos a revisar la eficacia y eficiencia de los distintos algoritmos utilizados para resolver este problema. La implementacion de estos se ha realizado en c++ 17.

2.1. Estructuras de datos

Para la implementacion de los algoritmos se han usado las siguientes estructuras de datos:

- Clase problema

  Esta clase contiene un vector de vectores de enteros, que representa la matriz de distancias que define el problema. Cuenta con un metodo para mostrar por pantalla la matriz de distancias si fuera necesario. Cuenta tambien con un vector de objetos de la clase Car, que a su vez contienen vectores de enteros en los que se calcularan las soluciones.

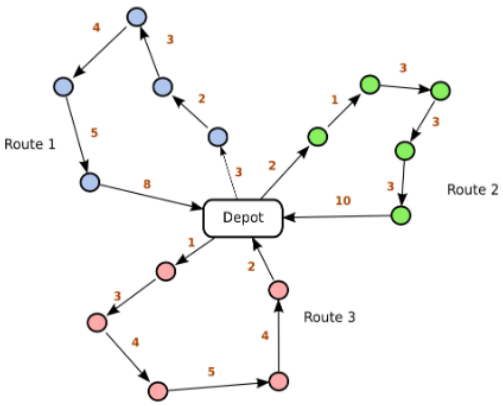

Figura 1: Ejemplo de posible solucion para un problema de VRP.

- Clase solucion

  Esta clase contiene un vector de vectores de enteros. Cada ruta es un vector de enteros que empieza y termina por 0, siendo este nodo el almacen. Cuenta con metodos para calcular el coste de una ruta, el coste total de la solucion y mostrar la solucion por pantalla.

- Clase Greedy

  Esta clase contiene un objeto de la clase problema y metodos para resolver dicho problema usando el algoritmo greedy diseñado y obtener el nodo mas cercano al ultimo nodo de alguna ruta indicada por parametro.

- Clase EnviromentStructure

  Clase abstracta que proporciona metodos para aplicar una busqueda local.

- Clase ReinsertionEntre

  Clase derivada de EnviromentStructure que define la estructura de entorno de reinsercion entre rutas

- Clase ReinsertionIntra

  Clase derivada de EnviromentStructure que define la estructura de entorno de reinsercion intra rutas

- Clase SwapEntre

  Clase derivada de EnviromentStructure que define la estructura de entorno de intercambio entre rutas

- Clase SwapIntra

  Clase derivada de EnviromentStructure que define la estructura de entorno de intercambio intra rutas

- Clase Grasp

  Esta clase contiene un objeto de la clase problema, un puntero que apunta a un objeto de la clase EnviromentStructure que nos permitira hacer las busquedas locales y un entero que indica el numero maximo de clientes por ruta. Cuenta con metodos para obtener la solucion al problema usando el algoritmo GRASP diseñado, construir soluciones iniciales, calcular el nodo mas cercano al ultimo visitado en una ruta y para escoger un nodo de la lista de candidatos restringida.

- Clase Gvns

  Esta clase contiene un objeto de la clase problema, un vector de punteros que apuntan a objetos de la clase EnviromentStructure que nos permitiran hacer las busquedas locales, y dos enteros que indican el numero de clientes maximos permitidos por ruta y el numero kMax usado en el algoritmo GVNS. Cuenta con metodos para obtener la solucion al problema usando el algoritmo GVNS, realizar shaking de una solucion dada, intensificar una solucion dada, construir soluciones iniciales y al igual que la clase Grasp calcular el nodo mas cercano al ultimo visitado en una ruta y para escoger un nodo de la lista de candidatos restringida. Estos ultimos se usan al calcular las soluciones iniciales pues estas serán caslculadas al igual que en el algoritmo GRASP.

2.2. Greedy

El funcionamiento del algoritmo greedy es sencillo. Empezamos desde el nodo 0, es decir el almacen, y mientras queden nodos por visitar visitaremos el nodo mas cercano. Para evitar visitar todos los nodos con un solo vehiculo hacemos que los nodos a calcular se añadan en una ruta distinta por cada iteracion, es decir, primero añadimos un nodo a la primera ruta, luego a la segunda y en la iteracion n añadiremos un nodo a la ruta n % numero de rutas. Una vez hayamos vistado todos los nodos, volvemos al nodo 0.

2.3. Grasp

Para el algoritmo GRASP primero debemos construir una solucion inicial, para ello usaremos el algoritmo greedy anteriormente descrito pero con un ligero cambio. En cada iteracion en lugar de visitar al nodo mas cercano, creamos una lista restringida de candidatos que vendra compuesta por los n nodos mas cercanos y visitamos un nodo al azar de entre todos ellos. De esta manera somos capaces de generar varias soluciones.

Una vez generada la solucion inicial le aplicamos una de las 4 busquedas locales posibles(re-insercion intra rutas, re-insercion entre ruta, intercambio intra ruta o intercambio entre ruta) para hallar el minimo local en el entorno descrito por su estructura de entorno.

Todas las busquedas locales son simplemente bucles en los que se comprueban todos los posibles cambios haciendo un solo movimeinto a partir de la solucion inicial y se devuelve la solucion de menor coste.

Ahora solo falta repetir este proceso un numero arbitrario de veces y quedarnos con la mejor solucion encontrada, en este caso se ha optado por repetir el bucle en el que generamos soluciones y las mejoramos 2000 veces, y si llegamos a iterar 500 veces sin mejorar la mejor solucion obtenida hasta el momento saldremos del bucle.

2.4. GVNS

Para el algoritmo GVNS primero debemos hallar una solucion inicial, lo haremos de la misma manera que la explicada para el algoritmo GRASP. Una vez contemos con la solucion inicial debemso entrar en bucle en el que realizaremos una fase de shaking para diversificar nuestras soluciones y otra de intensificacion de la soluciones para obtener optimos locales.

En el bucle del shaking definiremos un valor de k max que denotara el entorno k-esimo maximo que vamos a utilizar. El bucle empezará por aplicar shaking para el primer entorno, la estructura de entorno a suar en la fase de shaking es intercambio entre rutas. Por tanto para generar una solucion en el entorno k-esimo debemos realizar k veces movimientos no repetidos de reinsercion de nodos entre rutas. Luego realizaremos la fase de intensificacion y si la solucion mejora esta pasará a ser nuestra nueva solucion actual y volvemos a fijar el valor de k a 1, en otro caso debemos sujmar 1 a k.

Para el proceso de intensificacion simplemente debemos aplicar las busquedas locales definidas para el VND. En nuestro caso estas son, en orden, re-insercion entre rutas ->re-insercion intra ruta ->intercambio entre rutas ->intercambio intra ruta. La manera de aplicar estas es la siguiente, entramos en un bucle mientras que l <lMax, lMax será el numero de busquedas locales de las que dsipongamos, en nuestro caso 4. Dentro del bucle aplicamos la busqueda local correspondiente al valor de l, y vemos si esta mejora la que teniamos hasta el momento. De ser asi, esta pasaría a ser nuestra nueva solucion actual y fijariamos el valor de l a 1, de lo contrario, aumentariamos en 1 el valor de l.

De vuelta al bucle principal, una vez realizados los procesos de diversificacion en intensificacion y obtenida una solucion optima local a partir de una solucion inicial, repetimos este proceso un numero arbitrario de veces para poder contar con un proceso multiarranque. En este caso se ha optado por hacer 4000 iteraciones y 2000 iteraciones sin mejora

2.5. Tablas de resultados

Greedy:

Problema	Distancia recorrida	Tiempo(s)
I40j 2m S1	227	2.9e-05
I40j 4m S1	281	3.1e-05
I40j 6m S1	352	3.1e-05

Grasp(re-insercion intra):
Grasp(re-insercion entre):
Grasp(intercambio intra):
Grasp(intercambio entre):
GVNS:

Problema	LRC	Ejecucion	Distancia recorrida	Tiempo(s)
I40j 2m S1	2	1	155	0.799117
I40j 2m S1	2	2	155	0.77799
I40j 2m S1	2	3	160	0.760323
I40j 4m S1	2	1	209	0.473022
I40j 4m S1	2	2	209	0.478612
I40j 4m S1	2	3	209	0.477019
I40j 6m S1	2	1	289	0.411399
I40j 6m S1	2	2	286	0.410477
I40j 6m S1	2	3	286	0.409356
I40j 2m S1	3	1	127	0.980002
I40j 2m S1	3	2	167	0.943502
I40j 2m S1	3	3	144	0.954734
I40j 4m S1	3	1	215	0.669631
I40j 4m S1	3	2	205	0.660002
I40j 4m S1	3	3	209	0.656226
I40j 6m S1	3	1	289	0.575072
I40j 6m S1	3	2	282	0.570202
I40j 6m S1	3	3	286	0.573776

Problema	LRC	Ejecucion	Distancia recorrida	Tiempo(s)
I40j 2m S1	2	1	149	0.515671
I40j 2m S1	2	2	149	0.514645
I40j 2m S1	2	3	164	0.522705
I40j 4m S1	2	1	204	0.844072
I40j 4m S1	2	2	204	0.860287
I40j 4m S1	2	3	216	0.854746
I40j 6m S1	2	1	263	1.14252
I40j 6m S1	2	2	252	1.15185
I40j 6m S1	2	3	263	1.14404
I40j 2m S1	3	1	156	0.676064
I40j 2m S1	3	2	149	0.680948
I40j 2m S1	3	3	160	0.666045
I40j 4m S1	3	1	193	1.06016
I40j 4m S1	3	2	190	1.04217
I40j 4m S1	3	3	190	1.04147
I40j 6m S1	3	1	252	1.35046
I40j 6m S1	3	2	261	1.36061
I40j 6m S1	3	3	252	1.35386

Problema	LRC	Ejecucion	Distancia recorrida	Tiempo(s)
I40j 2m S1	2	1	174	0.454476
I40j 2m S1	2	2	174	0.447551
I40j 2m S1	2	3	174	0.448171
I40j 4m S1	2	1	222	0.393807
I40j 4m S1	2	2	222	0.391801
I40j 4m S1	2	3	229	0.426272
I40j 6m S1	2	1	285	0.352518
I40j 6m S1	2	2	304	0.355154
I40j 6m S1	2	3	285	0.354715
I40j 2m S1	3	1	158	0.616279
I40j 2m S1	3	2	170	0.614773
I40j 2m S1	3	3	174	0.617899
I40j 4m S1	3	1	229	0.528939
I40j 4m S1	3	2	234	0.522859
I40j 4m S1	3	3	229	0.523286
I40j 6m S1	3	1	285	0.492206
I40j 6m S1	3	2	285	0.490145
I40j 6m S1	3	3	285	0.485477

Problema	LRC	Ejecucion	Distancia recorrida	Tiempo(s)
I40j 2m S1	2	1	155	0.68406
I40j 2m S1	2	2	155	0.68531
I40j 2m S1	2	3	168	0.683248
I40j 4m S1	2	1	196	0.936065
I40j 4m S1	2	2	199	0.944754
I40j 4m S1	2	3	196	0.946121
I40j 6m S1	2	1	217	1.13148
I40j 6m S1	2	2	231	1.07246
I40j 6m S1	2	3	217	1.11655
I40j 2m S1	3	1	155	0.844531
I40j 2m S1	3	2	167	0.842937
I40j 2m S1	3	3	163	0.842684
I40j 4m S1	3	1	197	1.22561
I40j 4m S1	3	2	199	1.22393
I40j 4m S1	3	3	197	1.20694
I40j 6m S1	3	1	233	1.49604
I40j 6m S1	3	2	235	1.51618
I40j 6m S1	3	3	233	1.46954

Problema	k-max	Ejecucion	Distancia recorrida	Tiempo(s)
I40j 2m S1	4	1	125	5.6237
I40j 2m S1	4	2	130	6.21388
I40j 2m S1	4	3	144	5.12654
I40j 4m S1	4	1	166	6.60634
I40j 4m S1	4	2	168	6.76581
I40j 4m S1	4	3	166	6.7791
I40j 6m S1	4	1	207	7.53413
I40j 6m S1	4	2	209	6.27995
I40j 6m S1	4	3	208	7.94247
I40j 2m S1	6	1	146	12.599
I40j 2m S1	6	2	130	12.8992
I40j 2m S1	6	3	138	12.4034
I40j 4m S1	6	1	166	13.8307
I40j 4m S1	6	2	163	13.2898
I40j 4m S1	6	3	174	11.9233
I40j 6m S1	6	1	209	15.6882
I40j 6m S1	6	2	209	12.3909
I40j 6m S1	6	3	209	14.8653