Comparación de dos heurísticas para el problema de enrutamiento de vehículos con restricción por capacidad

Esteban Sánchez Durán

Agosto 2023

Introducción

El problema de enrutamiento de vehículos (VRP por sus siglas en inglés) es un problema computacional en el que se toma una flota de vehículos, un almacén y un número de clientes y se busca encontrar la ruta óptima para llegar a estos clientes y regresar al almacén con los vehículos disponibles. También existen diferentes restricciones adicionales que se pueden aplicar, como ventanas de horarios, capacidad, distancia, etc. Este problema y sus soluciones tienen muchas aplicaciones prácticas en el área de administración de sistemas de transportes y entrega de productos.

En este problema se representan los sitios de entrega como los vértices de un grafo y las rutas que las conectan entre sí como sus aristas. Usualmente se realiza en grafos dirigidos y pesados, pero existen variantes para grafos no dirigidos o no pesados.

El objetivo de este trabajo es resolver la variante restringida por capacidad del VRP (CVRP) en instancias que utilicen datos geográficos de la ciudad de Mérida. Para resolverlo se utilizará Programación Lineal para encontrar la solución óptima para cada instancia; posteriormente se aplicarán dos heurísticas

distintas a las instancias y se compararán las soluciones encontradas, así como el tiempo usado por cada método.

Métodos de Solución

Los tres métodos de solución que se utilizarán serán Programación Lineal, el algoritmo de Clarke y Wright y el algoritmo de Clarke y Wright con la mejora propuesta por Gaskell y Yellow.

Programación Lineal

Para obtener la solución óptima de un problema de VRP se plantea como un problema de programación lineal. Este está conformado por una función objetivo y restricciones.

El modelo de programación lineal que se utilizó para este trabajo es el siguiente:

Con p vehículos con r capacidad y $n+1$ vértices siendo el vértice 0 el almacén. Sea c_{ij} el costo de la arista que va del vértice i al vértice j, d_i la demanda de el vértice i y x_{ijk} la variable binaria que es igual a 1 si la arista que va del vértice i al vértice j es parte de la ruta del vehículo k en la solución óptima, e igual a 0 si no lo es.

Función objetivo:

$$Min \quad \sum_{k=1}^{p}\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ijk}$$

Sujeta a:

$$x_{iik} = 0 \quad \forall\, k, i$$

$$\sum_{k=1}^{p}\sum_{i=0}^{n} x_{ijk} \;\; \forall\, j > 0$$

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik} \quad \forall \;\; j, k$$

$$\sum_{j=1}^{n} x_{0jk} = 1 \quad \forall \;\; k$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} d_i x_{ijk} \leq r \;\; \forall\, k$$

$$\sum_{i\in S}\sum_{j\notin S} x_{ijk} \geq 2 \quad S \subset V \setminus \{0\} \quad 2 \leq |S| \leq n-2$$

Algoritmo de Clarke y Wright

El algoritmo de Clarke y Wright es una heurística para el VRP que también puede aplicarse a su variante restringida por capacidad. La base de este algoritmo son los ahorros. Funciona de la siguiente forma:

Considerando c_{ij} como el costo de la arista que conecta del vértice i al vértice j. Entonces el ahorro $s_{ij} = c_{i0} + c_{oj} - c_{ij}$ es el costo ahorrado de unir dos rutas por los vértices i a j. Existen varias formulaciones del algoritmo pero la que fue utilizada para este proyecto fue:

1. Se calculan los ahorros s_{ij} para todos los vértices i y j tal que $i \neq j$. Estos se ordenan de manera no creciente.

2. Se inicializan rutas para todos los vértices i de la forma $(0, i, 0)$

3. Empezando desde el inicio de la lista se elige el ahorro s_{ij} de mayor valor que sea factible, es decir, que existan dos rutas distintas, una que contenga la arista $(0, j)$ y otra la arista $(i, 0)$, y que el unir las rutas no supere la capacidad máxima del vehículo. Si se cumplen estas condiciones, unir las rutas de la forma $(0, ...i, j, ...0)$. Repetir este proceso hasta recorrer todos los ahorros (mientras estos sigan siendo de valor positivo).

Como podemos observar, este es un algoritmo relativamente simple y fácil de implementar, por lo que lo hace de las heurísticas más utilizadas para resolver el VRP en situaciones prácticas.

Algoritmo de Clarke y Wright (mejora de Gaskell y Yellow)

Desde la creación del algoritmo de Clarke y Wright se han hecho varias mejoras a este, tanto en el ámbito de la eficiencia como en la exactitud. En las computadoras modernas el tiempo que toma el algoritmo de Clarke y Wright es despreciable, por lo que no nos interesan mejoras en la eficiencia en tiempo. La mejora que se usará para este proyecto es una propuesta por Gaskell y Yellow.

Esta es una modificación muy simple al algoritmo original, cambia la forma en la que se calculan los ahorros. Esta mejora plantea los ahorros como $s_{ij} = c_{i0} + c_{oj} - \lambda c_{ij}$ donde λ es un parámetro de forma de ruta. Usualmente se utiliza un $\lambda = 0.4$ o $\lambda = 1$. Para este trabajo se utilizara $\lambda = 0.4$

Instancias

Para la obtención de datos geográficos se utilizará la herramienta OpenStreetMap, una herramienta gratuita para obtener datos de calles, carreteras, tráfico, etc. Para ello se descarga un archivo osm que después será manejado con el software SUMO, específicamente el editor gráfico de redes incluido llamado netedit.

Una vez que se tienen los datos de red de carreteras de la zona geográfica elegida, se crearán las instancias de los problemas. Se crearán instancias con 5, 8, 10 y 12 clientes. Cada cliente tendrá asignada aleatoriamente una demanda;

también se asignarán arbitrariamente la cantidad de vehículos para cada instancia y la capacidad de cada vehículo.

Usando la librería de sumolib, un set de módulos para python dedicados a trabajar con redes de SUMO, se obtendrán los datos para cada instancia. Para ello se requiere obtener un arreglo de demandas para los vértices (los clientes del problema) y una matriz de costos que representen los costos de las aristas que conectan a los vértices.

Cada instancia es generada aleatoriamente con parámetros definidos: número de clientes, número de vehículos, capacidad de vehículos. Para generar esta instancia se situará el almacén en el centro de la red. Luego se generarán puntos con coordenadas aleatorias definidas dentro del tamaño de la red, estos serán los clientes. Estos puntos se asignarán a la calle más cercana en la red de SUMO. A estos puntos se les asignará aleatoriamente una demanda entera en un intervalo entre 1 y 5. Con esto se puede generar el arreglo de demandas para los vértices. Por último, se necesita generar la matriz de costos de las aristas entre los vértices, para lo cual se requiere obtener las distancias que tienen todos los vértices entre sí. Pero la distancia euclidiana no es lo suficientemente correcta, ya que no toma en cuenta las direcciones y sentidos de las calles reales de una ciudad. Por este motivo se utilizará un método de sumolib para encontrar el camino más corto entre dos vértices, respetando las direcciones y sentidos de las calles utilizadas en el camino. Al obtener este valor, se utilizará para crear la matriz de costos de las aristas.

Una vez que se generan estos datos, se puede resolver el problema con los métodos elegidos.

Además, se crearán instancias usando distancias euclidianas para los costos de las aristas. De igual manera se crearán instancias para 5, 8, 10 y 12 clientes.

Resultados

Como ya se había mencionado, para comparar los métodos de solución se usarán cinco instancias para cada cantidad de clientes (5, 8, 10 y 12 clientes). Estos se resolverán utilizando los tres métodos y se compararán las soluciones y el tiempo que tomaron. Como es un problema de optimización que busca encontrar la solución con el menor costo, la mejor solución es aquella que tiene un menor valor. El método de programación lineal nos brinda con la solución óptima. Se presentarán dos tablas, una para las instancias creadas con los datos de distancias euclidianas y otra para las instancias creadas con datos geográficos. Por último, el tiempo de solución para el algoritmo de Clarke y Wright y su versión mejorada es igual, por lo que sólo se incluirá una vez por instancia.

Instancia	C y W	C y W (E)	PL	C y W [s]	PL [s]
E5A	219.65	219.65	219.65	0.0020	0.2201
E5B	167.79	167.79	167.79	0.0030	0.0315
E5C	175.98	175.98	175.98	0.0019	0.0197
E5D	227.93	227.93	227.93	0.0030	0.1854
E5E	176.65	176.65	176.65	0.0029	0.0274
E8A	271.16	277.83	271.15	0.0040	2.7625
E8B	269.28	280.69	269.25	0.0040	1.0253
E8C	249.80	303.32	249.80	0.0030	0.8850
E8D	271.08	294.73	271.08	0.0039	11.1519
E8E	386.08	406.93	380.55	0.0060	9.7728
E10A	231.09	254.74	229.04	0.0049	59.7849
E10B	303.39	307.84	303.39	0.0050	52.9440
E10C	372.79	429.58	365.22	0.0069	19.4675
E10D	310.80	344.09	306.16	0.0049	8.7755
E10E	274.14	295.99	266.97	0.0059	46.5363
E12A	291.23	291.23	291.23	0.0082	74.2258
E12B	399.95	468.02	386.52	0.0059	5.3382
E12C	298.39	263.51	260.83	0.0069	30.0511
E12D	305.75	366.76	305.75	0.0059	833.0690
E12E	329.27	381.03	329.27	0.0090	455.0123

Columna 1: Nombre de la instancia, Columna 2-4: Resultados de los métodos, Columna 5-6: Tiempo de ejecución.

Table 1: Resultados con distancias euclidianas

Lo más aparente de esta tabla es el rápido crecimiento del tiempo de ejecución para obtener la solución con el método de Programación Lineal, llegando a tomar más de 10 minutos para resolver un problema de tamaño relativamente pequeño. Mientras tanto, los algoritmos de Clarke y Wright no pasan de 0.01 segundos en su tiempo de solución. El algoritmo original de Clarke y Wright tiene en promedio un porcentaje de error de 1.32%, mientras que para el algoritmo mejorado de Clarke y Wright es de 7.75%.

Instancia	C y W	C y W (E)	PL	C y W [s]	PL [s]
5A	11318.53	11744.78	11318.53	0.0029	0.1105
5B	13627.27	13647.54	13627.27	0.0019	0.0306
5C	11862.32	11990.52	11854.00	0.0020	0.0259
5D	12435.96	12468.37	12435.96	0.0029	0.0442
5E	13437.81	13135.22	13135.22	0.0020	0.0214
8A	19629.36	20105.45	19559.50	0.0030	6.1512
8B	15567.99	16145.91	15567.99	0.0040	8.6980
8C	21798.02	24901.62	21798.02	0.0059	20.0218
8D	18178.31	22291.22	18178.31	0.0030	2.6540
8E	18318.23	18888.62	18284.10	0.0050	6.0548
10A	14917.12	15947.01	14908.05	0.0059	4.9141
10B	14359.53	15196.02	14359.53	0.0044	6.4091
10C	15183.51	15183.51	15183.51	0.0040	182.7404
10D	15976.88	16420.79	15976.88	0.0059	0.3500
10E	22750.50	25202.30	22750.50	0.0049	542.2112
12A	17875.37	18542.74	17785.97	0.0070	1670.6887
12B	14914.22	15338.19	14734.90	0.0090	151.5116
12C	13426.76	14003.87	13426.76	0.0069	15.5383
12D	15813.99	16760.37	15813.99	0.0050	50.7040
12E	17458.04	18202.97	17458.04	0.0059	18.7892

Columna 1: Nombre de la instancia, Columna 2-4: Resultados de los métodos, Columna 5-6: Tiempo de ejecución.

Table 2: Resultados con datos geográficos

Se puede observar que el crecimiento del tiempo para el método de Programación Lineal es comparable, si no mayor, que con las anteriores instancias, incluso llega a tomar más de 25 minutos para una instancia. Pero lo más interesante es que los otros dos métodos de solución se comportaron significativamente mejor que con las instancias de distancias euclidianas. En este caso, el algoritmo original de Clarke y Wright tiene un promedio de porcentaje de error de tan sólo 0.23% y para el algoritmo mejorado de Clarke y Wright es de 5.06%.

En las instancias que usan datos geográficos podemos demostrar gráficamente las rutas que resultan de los distintos métodos de solución.

Figure 1: Programación Lineal

Figure 2: Clarke y Wright

Figure 3: Clarke y Wright mejorado

Discusión

De los resultados hay dos observaciones que resultan interesantes.

La primera es la diferencia en la exactitud que mostraron la versión original y la mejorada del algoritmo de Clarke y Wright. Se esperaba que la versión mejorada del algoritmo obtuviera resultados más exactos, pero en ambos casos demostró tener un promedio de porcentaje de error significativamente más alto que la versión original. Esto se puede deber a varios factores: la versión mejorada del algoritmo no es apta para este tipo específico de problemas de CVRP; el algoritmo se comporta significativamente mejor para instancias mucho más grandes que las elegidas o simplemente hubo un error en el momento de la implementación del algoritmo.

La segunda observación, y en mi opinión la más interesante, es la diferencia en exactitud para ambas heurísticas entre las instancias de distancias euclidianas y las realizadas con datos geográficos. Se esperaba que se comportaran de igual manera para ambos tipos de instancias o peor para las instancias con datos geográficos. Por lo contrario, ambos algoritmos heurísticos tuvieron un porcentaje de error significativamente menor en las instancias con datos geográficos. El algoritmo original de Clarke y Wright tuvo un porcentaje de error de 1.32% en las instancias euclidianas y de tan sólo 0.23% en las instancias con datos geográficos. De igual forma, el algoritmo mejorado tuvo un porcentaje de error de 7.75% en las instancias euclidianas, y de 5.06% en las instancias con datos geográficos. Además, el algoritmo original de Clarke y Wright obtuvo soluciones con errores significativos (más de 1% de porcentaje de error) en muchas menos ocasiones para las instancias con datos geográficos (2 de 20 instancias) que para las instancias euclidianas (6 de 20 instancias). Mi hipótesis para explicar este comportamiento es que en las instancias generadas utilizando datos geográficos, las limitaciones de las rutas posibles entre los diferentes vértices al tener que usar las calles y carreteras genera una matriz de costos en la que las soluciones más exactas son más fáciles de alcanzar con los métodos específicos de estas heurísticas.

Conclusión

La realización de este trabajo mostró varios retos y limitaciones, pero también brindó resultados interesantes y una apertura a subsecuentes investigaciones respecto al tema.

En cuanto a las limitaciones, el problema principal fue en el momento de generar instancias utilizando los datos geográficos. Estas instancias se generan con el objetivo de comparar los resultados de las heurísticas con el resultado óptimo. Esto resultó relativamente sencillo para instancias pequeñas, pero para instancias más grandes demostraría ser una gran limitación de los estudios realizados sobre este tema. Como los datos geográficos son específicos de la zona tomada, no existen instancias estándar que hayan sido estudiadas durante años a las que se les haya encontrado una solución óptima. Por esto mismo, tenemos que generar nuestras propias soluciones óptimas.

Por otro lado, las herramientas desarrolladas para este trabajo nos permitirían trabajar con datos geográficos de cualquier lugar en el mundo, y al ver el buen comportamiento del algoritmo de Clarke y Wright (que es probablemente la heurística más fácil de implementar para el CVRP) nos permitiría explorar las aplicaciones prácticas del uso de esta heurística.

También sería interesante para trabajos futuros explorar el comportamiento de otras heurísticas e incluso metaheurísticas y observar si el fenómeno que consiste en un mejor comportamiento para instancias de datos geográficos que en instancias euclidianas se sigue presentando.

Bibliografía

- Toth, P., Vigo, D. (2002). 1. An overview of vehicle routing problems. En *Society for Industrial and Applied Mathematics eBooks* (pp. 1–26). https://doi.org/10.1137/1.9780898718515.ch1
- Toth, P., Vigo, D. (2002b). 2. Branch-And-Bound algorithms for the capacitated VRP. En *Society for Industrial and Applied Mathematics eBooks* (pp. 29–51). https://doi.org/10.1137/1.9780898718515.ch2
- Laporte, G., Semet, F. (2002). 5. Classical heuristics for the capacitated VRP. En *Society for Industrial and Applied Mathematics eBooks* (pp. 109–128). https://doi.org/10.1137/1.9780898718515.ch5
- Cordeau, J., Gendreau, M., Laporte, G., Potvin, J., Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5), 512–522. https://doi.org/10.1057/palgrave.jors.2601319

Anexos

El código utilizado para generar y resolver las instancias así como las instancias generadas para los resultados de este trabajo pueden ser encontrados en:

https://github.com/EstSanchez/CVRP-Merida