Ruteo de entrega de productos para una empresa de e-commerce

Echeagaray-Gonzalez, J. P. [1] Méndez-Cruz, E. R. [1] Garcia-De-La-Fuente, V. V. [1] Longoria-Lozano, C. [1]

[1]*Instituto Tecnológico y de Estudios Superiores de Monterrey*

Resumen

Se formula un modelo de CVRP para satisfacer las necesidades de una empresa de E-commerce utilizando bases de datos proporcionadas por la misma, este modelo fue embebido y optimizado para minimizar las distancias recorridas por los camiones de entrega. El producto final despliega un reporte de las rutas así como una visualización de estas.

Introducción

El problema de Ruteo de Vehículos (VRP) se centra en la generación de un conjunto de rutas para su flotilla que minimice el costo total de las entregas, tomando en cuenta las demandas de los clientes, que todos los vehículos de este deben salir y regresar al mismo centro de distribución, cada cliente es visitado una sola vez, y a cada vehículo no se le pueden asignar entregas que sobrepasen su capacidad de carga máxima [2]. A nivel industrial, estas aproximaciones fallan debido al tiempo y poder computacional requerido para su uso en gran escala. Por esto, es recomendable introducir heurísticos al proceso de optimización, que si bien, no llegarían al conjunto de rutas óptimas, estarían cercanos al mismo. El problema propuesto se plantea de la siguiente manera:

Sean los conjuntos:

$$i \in \{1, \dots, n\}$$
$$j \in \{1, \dots, n\}$$
$$k \in \{1, \dots, V\}$$

Sean los parámetros:

V : Número de vehículos disponibles
Q : Capacidad de carga máxima del vehículo en m^3
n : Número de clientes a visitar
c_{ij} : Distancia geodésica entre el punto i y el punto j
d_j : Demanda en m^3 del cliente j

Sean las variables:

x_{ijk} : Variable binaria que toma el valor de 1
si el arco del punto i al punto j
forma parte de la ruta óptima recorrida por el vehículo k
$x_{ijk} \in \{0, 1\}$

Modelamos el problema de optimización como:

$$\min \quad z = \sum_{k=1}^{V} \sum_{j=1}^{n} \sum_{i=1}^{n} c_{ij} x_{ijk}$$

sujeto a:

$$\sum_{i=1}^{n} x_{ijk} = \sum_{i=1}^{n} x_{jik} \qquad \forall j, k$$
$$\sum_{k=1}^{V} \sum_{i=1}^{n} x_{ijk} = 1 \qquad \forall j$$
$$\sum_{j=2}^{n} x_{1jk} = 1 \qquad \forall k$$
$$\sum_{i=1}^{n} \sum_{j=2}^{n} d_j x_{ijk} \leq Q \qquad \forall k$$
$$u_j - u_i \geq d_j - Q(1 - x_{ijk}) \qquad \forall i, j, k$$
$$d_i \leq u_i \leq Q \qquad \forall i$$
$$x_{ijk} \in \{0, 1\} \qquad \forall i, j, k$$

Las restricciones anteriores hacen referencia a:

1. Número de veces que un camión entra por un punto debe de ser igual que el número de veces que las que sale
2. Cada camión solamente puede entrar una vez a un punto
3. Todos los vehículos salen del CEDIS
4. Límite a la capacidad de carga máxima del vehículo
5. Las últimas 2 restricciones eliminan los *sub-tours*

Para la problemática actual, se realizarán 192 entregas (n), se dispondrán de 9 vehículos (V), cada uno de ellos tendrá una capacidad máxima de carga de 18 m^3 (Q) y se utilizará la distancia geodésica. Los valores numéricos para las demandas y distancias son calculados dentro de la aplicación. El heurístico que usamos en nuestro modelo es el costo asociado entre 2 puntos de entrega, no se toma en cuenta un límite de distancia recorrida por vehículo ni tampoco se consideran ventanas de tiempo para realizar las entregas. [3].

Para realizar el trabajo debemos comprender el TSP. El *Travelling Salesman Problem* es un problema encargado de buscar la ruta más corta y eficiente para llegar a un destino. Este es clasificado como un problema de NP-Hard de acuerdo con la teoría de la complejidad computacional [5]. Entre los métodos de solución se encuentran los siguientes:

- Vecino más cercano [5]
- Ramificación y atadura: [5]
- Fuerza bruta [5]

El objetivo principal de nuestro trabajo es generar una metodología para resolver el problema de generación de rutas de entrega para una empresa de e-commerce que se acerque a la optimalidad. De forma específica, se desea encontrar un conjunto de n rutas que realicen todas las entregas de un día para la empresa, estas rutas deben de minimizar la distancia total recorrida así como respetar las diversas restricciones propias del problema y de la empresa.

Como hipótesis se propone una implementación que es capaz de obtener resultados cercanos a la solución óptima en tiempos aceptables, con un costo computacional bajo, en relación a otras implementaciones.

Metodología

Se realizó una limpieza de las cuatro bases de datos proporcionadas utilizando `Python` versión 3.8.10., las bases de datos consisten en las direcciones de los centros de distribución de la empresa, listado de los productos que venden y propiedades, tipos de vehículos que la empresa utiliza, y listado de las entregas que se deben de realizar. Con estas se determinaron las entregas a hacer en un día y se obtuvieron los domicilios de los clientes, aquellos que cumplieran con la información suficiente requerida fueron geo-codificados utilizando la librería `Photon` [7] para calcular una matriz de distancias geodésicas todos contra todos, se utilizó la función implementada en el módulo `distance` de la librería `Geopy` [4] y a través de la función `cdist` implementada en el sub-módulo `spatial` de la librería de cómputo científico `SciPy` [11]. Con la máquina utilizada, este proceso toma 13 minutos y agrega una matriz que incluye los datos de los domicilios de los compradores.

La red con la que tratamos consta de 193 nodos, 192 referentes a los clientes, y 1 nodo más para el Centro de Distribución. El número de conexiones (aristas) dentro del grafo es $\binom{193}{2} = 18528$. En conjunto con la matriz de distancias y el volumen de los pedidos de cada cliente se realizó un proceso de optimización con el `solver` implementado en la librería `ortools` [6] de libre distribución. Finalmente se desplegó un reporte con el conjunto de rutas encontradas en un mapa en adición a datos generales de la solución propuesta, tales como el volumen muerto y el consumo de gasolina.

En la solución del problema se ha utilizado la librería `ortools` [8] de Google. Con la matriz de distancias calculada en el paso anterior, se hace un listado de cuánto volumen debe de ser entregado a cada uno de los clientes, un número de vehículos a mandar, y su capacidad máxima de carga en metros cúbicos. La solución consiste de una lista de las rutas a seguir por cada uno de los vehículos, el volumen total que llevarán, cuál fue la distancia total recorrida por todos los vehículos y cuánta carga llevaron.Para determinar el mejor tiempo límite, se obtuvieron 10 muestras de resultados para tiempos máximos de cómputo de entre 1 y 10 segundos; de esta se recuperó la mejor distancia encontrada, así como la media, el máximo, la desviación estándar y la varianza. Con la API de la librería `folium` [9] se generó la visualización, este generó un mapa de la ciudad en la que se observan las rutas de los vehículos; además despliega información del domicilio al que se realiza la entrega, y el volumen total de los productos que debe de recibir.

Resultados

En la figura 1 se presentan las curvas de optimización generadas, se grafica la distancia mínima encontrada, la distancia media y la máxima del conjunto de 10 muestras.

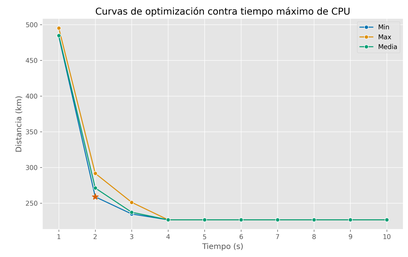

Figure 1. Curva de optimización con diferentes tiempos máximos de cómputo

También hemos generado una curva para varianza de cada una de las observaciones en la figura 2.

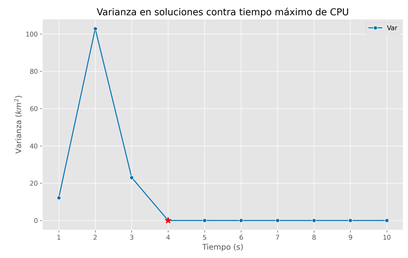

Figure 2. Varianza de soluciones encontradas para diferentes tiempos máximos de cómputo

Con estas curvas, se utilizó la librería de `Python`, `kneed` [1]. Esta implementa el algoritmo propuesto por Satopaa en Finding a “Kneedle” in a Haystack: Detecting Knee Points in System Behavior [10]. En la figura 1 y en la figura 2 se visualiza el punto encontrado.

El punto rodilla de la distancia mínima sugeriría que a partir de 2 segundos de cómputo, esta métrica no cambiará de forma drástica; sin embargo, en la curva de la varianza este punto se localiza en 4 segundos. Con el fin de asegurar la mayor consistencia, sugerimos que para la generación de rutas se realicen 10 simulaciones con un tiempo máximo de cómputo de 4 segundos por simulación. Después se escoge la que tenga la menor distancia asociada entre las 10 simulaciones generadas.

Una vez se corre la aplicación, se desplegará en el navegador el conjunto de rutas de todos los vehículos, como se pueden observar dos de ellas en la figura 3.

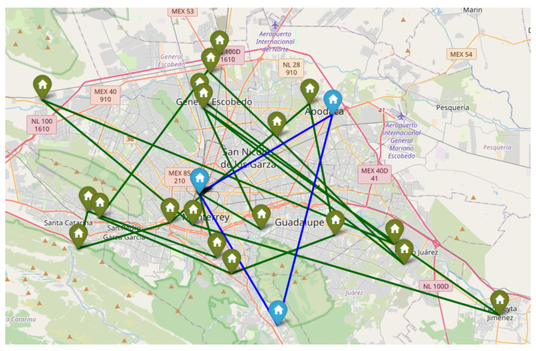

Figure 3. Ejemplo de rutas generadas

Para esta simulación, la aplicación encontró un conjunto de rutas que minimizan la distancia que recorren los camiones en total hasta 226.79 km. En esta solución el vehículo con la ruta más corta viaja 12.86 km y el que tiene la ruta más larga viaja 42.21 km, se tiene una huella de carbono de 55 kg de CO_2; en este tabulado cada una de las filas representa una ruta encontrada por la aplicación.

A pesar de que el producto que hemos desarrollado ya es funcional, hemos encontrado algunas áreas de oportunidad que de ser resueltas incrementarían aún más el valor de la aplicación.

La aplicación hace uso del motor de búsqueda de domicilios `Photon` [7] para geo-codificar los domicilios disponibles en la base de datos; consideramos que la adquisición de una licencia comercial podría aumentar aún más el número de domicilios que son geo-codificados con éxito. Esta también utiliza la distancia geodésica como métrica de distancia, lo cual está relativamente lejos de ser una aproximación rigurosa para determinar la distancia entre dos puntos. Sugerimos que se adquiera una licencia comercial que calcule esta distancia, su uso le daría una mayor validez a los resultados encontrados.

Tenemos que el límite de tiempo máximo impuesto al proceso de optimización en nuestra aplicación está ligado al número máximo de cálculos por segundo del equipo usado; en caso de que nuestro sistema sea implementado en un ordenador con un mayor poder de cómputo, el tiempo necesario para encontrar una solución óptima se verá reducido.

El proceso de la aplicación podría ser condensado en 4 etapas, limpieza de datos, geo-codificación de domicilios y cálculo de matriz de distancias, optimización y despliegue de resultados. Se han realizado 10 simulaciones de las que se calculó el tiempo promedio de ejecución, dichos resultados se presentan en la tabla 1:

Proceso	Tiempo (s)
Limpieza de datos	7.7
Geo-codificación y matriz de distancias	777
Optimización	42.3
Despliegue de rutas	1.18

Table 1. Tiempos de cómputo de cada proceso

El tiempo total de ejecución es de 13 minutos con 48 segundos, tiempo del cual el proceso de geo-codificación representa el 93.8% de la operación completa. Pensamos que el uso de un software más avanzado podría reducir drásticamente este tiempo de ejecución. Sin embargo, hay que destacar que el tiempo de geo-codificación puede ser mitigado con una base de datos en la que se lleve un registro de los domicilios de los clientes en conjunto con sus coordenadas asociadas.

Conclusiones

El ruteo de vehículos es uno de los desafíos más relevantes para la Investigación de Operaciones, ya que el diseño de algoritmos eficientes que logren encontrar rutas de entrega óptimas con cantidades más grandes de puntos de entrega es un imperativo para cualquier departamento de logística. Dado el gran número de puntos de entrega y la enorme cantidad de posibles rutas a explorar, el uso de algoritmos heurísticos que logren encontrar rutas sub-óptimas es una alternativa a considerar para cualquier empresa; el que estos métodos puedan encontrar buenas soluciones en tiempos razonables, hace que sean más sencillos de implementar en producción.

La aplicación desarrollada es capaz de producir resultados cercanos a la optimalidad con recursos computacionales limitados, con un tiempo de operación aceptable. Esta solamente se vería beneficiada si es que se ejecutase en un ordenador de carácter industrial, volviendo sencillo encontrar mejores soluciones en ventanas de tiempo más pequeñas. Además, que siguiendo la metodología y algoritmos que proponemos, hemos encontrado que el mejor tiempo de procesamiento es de cuatro segundos, obteniendo resultados favorables y una baja demanda en costo temporal y computacional.

Referencias

[1] Kevin Arvai. kneed. `https://github.com/arvkevi/kneed`, 2022.

[2] Raafat Elshaer and Hadeer Awad. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, 140:106242, 2020.

[3] Güneş Erdoğan. An open source spreadsheet solver for vehicle routing problems. *Computers & Operations Research*, 84:62–72, 2017.

[4] Kostya Esmukov. Geopy. `https://github.com/geopy/geopy`, 2022.

[5] Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[6] Google. Routing Options | OR-Tools |, 2021.

[7] Christoph Lingg. photon. `https://github.com/komoot/photon`, 2022.

[8] Laurent Perron and Vincent Furnon. Or-tools.

[9] python visualization. Folium.

[10] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a” kneedle” in a haystack: Detecting knee points in system behavior. In *2011 31st international conference on distributed computing systems workshops*, pages 166–171. IEEE, 2011.

[11] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

Código QR del video