

# Proyecto Modelado, Simulación y Optimización

## Segunda Entrega

Mauricio Martinez 202314461

Nicolas Gonzalez 202310041

Samuel Rodríguez Torres 202310140

### 1. Implementación del Modelo Matemático (10%):

Para la implementación del modelo matemático en Pyomo, no fue necesario realizar ningún cambio en comparación con nuestra entrega pasada, por lo que el modelado se mantiene igual a excepción de la función objetivo, la cual fue dada en la guía del enunciado y es la siguiente:

$$\min Z = \sum_{v \in V} (C_{fixed_v} * y_v) + \sum_{u \in V} (C_{dist_v} * d_v) + \sum_{u \in V} (C_{time_v} * t_v) + C_{fuel} + C_{special}$$

Para mayor referencia, ver la entrega 1 del proyecto.

### 2. Procesamiento de Datos (5%):

Para procesar los datos que se encuentran en los archivos .csv de cada caso se aplicaron las siguientes estandarizaciones para los 3 casos:

- Se crearon las clases y objetos para vehículos, clientes y depósitos, de manera que al leer cada archivo se crean los objetos asociados a las clases y los parámetros como el precio de la gasolina son atributos de la clase main.
- Para calcular las distancias, se propuso usar Google Maps Directions API, esto con el fin de reflejar la realidad y el contexto del problema, usar fórmulas como haversine violaría las calles y restricciones particulares de movilidad de cada ciudad. Esta API nos permite ingresar las 2 coordenadas en (lat,long) y nos retorna la distancia en metros, entre otras cosas de menor importancia.

### 3. Resolución del Modelo para Casos de Estudio (60%):

#### a. Caso 1 (15%):

Para la realización del caso base, se definieron 4 clases de dominio: vehículo, que contiene un id, código, capacidad, rango máximo en km, eficiencia en km por galón de gasolina y costo de gasolina por km. Cliente que tiene un id, código, latitud, longitud y demanda. Depósito que tiene un id, código, latitud y longitud. Y Mainconfig, que agrupa las clases anteriores usando parámetros como distancia al cliente, tiempo al cliente y costo de gasolina, además de los diccionarios de vehículos, clientes y depósitos. Además, contiene matrices de distancias/tiempo y los valores de los Big-M necesarios en el modelo. Se crearon funciones para cargar archivos CSV y almacenar los datos en las clases correspondientes dentro de MainConfig. Además se implementó la API de Google Maps para obtener las distancias y tiempo hasta los clientes. Se crean variables de decisión para determinar si un vehículo recorrió el nodo, si se utilizó un vehículo, la asignación del depósito al vehículo, etc.

Las restricciones aseguraron que cada cliente sea visitado exactamente una vez, que haya conservación de flujo, que cada vehículo salga y regrese a un único depósito, que solo existan arcos si el vehículo está activo, que la distancia total no exceda la autonomía, y que la demanda total de los clientes atendidos por cada vehículo no supere su capacidad.

También se incluyó la restricción MTZ para evitar subtours.

Luego de finalizar el modelo, se crea el solver GLPK, se reconstruyen las rutas por vehículo y se muestra un mapa interactivo con todos los resultados; finalmente se crea el archivo de validación.

Datos de verificación:

| VehicleId | DepotId | InitialLoad | RouteSequence | ClientsServed                                   | DemandsSatisfied | TotalDistance                                     | TotalTime | FuelCost   |              |
|-----------|---------|-------------|---------------|---|------------------|---|-----------|------------|--------------|
| 0         | V001    | CD01        | 126.0         | CD01-C004-C008-C005-C011-C017-C006-C018-CD01    | 7                | 15.0-20.0-20.0-17.0-25.0-17.0-12.0                | 94.648    | 229.216667 | 51425.413333 |
| 1         | V002    | CD01        | 138.0         | CD01-C001-C010-C024-C009-C016-C019-C003-C002-C_ | 10               | 13.0-15.0-11.0-20.0-10.0-11.0-12.0-15.0-17.0-14.0 | 136.768   | 338.550000 | 74310.613333 |
| 2         | V008    | CD01        | 113.0         | CD01-C015-C022-C013-C012-CD020-CD023-C014-CD01  | 7                | 17.0-18.0-21.0-12.0-15.0-15.0-15.0                | 75.622    | 168.366667 | 41087.953333 |

### b. Caso 2 (25%):

Para la implementación del caso 2 se definió primero las clases de domino. Vehicle guarda el id, tipo, capacidad, autonomía y combustible por Km; Client tiene id, coordenadas y demanda; Depot tiene id, coordenadas y capacidad. MainConfig es la clase en la que se centrará todas las clases guardando los vehículos, clientes y depósitos, los parámetros de costos globales, las matrices de distancia/tiempo y los Big-M que serán usados en el modelo.

Después de esto se crearon funciones para la carga de los csv para que posteriormente se guarden en las clases ya creadas y queden agrupadas en el MainConfig. Para tener en cuenta los valores de distancia, se creó una API de googlemaps para que en cada nodo se extraiga la distancia y la duración del trayecto.

Con todo cargado, se procede a crear el modelo donde se definen los conjuntos pertinentes en Pyom, vehículos, clientes, depósitos, nodos y un conjunto de arcos. Se crean los parámetros demanda por cliente, capacidad por depósito, capacidad y autonomía por vehículo, costo de combustible por km, y las matrices  $dist[i,j]$  y  $time[i,j]$ , además de los costos globales  $C_{fixed}$ ,  $C_{dist}$  y  $C_{time}$ . Luego de esto se procedió a crear las variables de decisión para ver si un vehículo recorrió cierto nodo, si el vehículo se usó, la asignación del depósito con el vehículo, el orden de vista y más. Luego de esto se declararon las restricciones que aseguran que cada cliente se visita exactamente una vez; conservación de flujo en cada cliente y vehículo; que cada vehículo salga y regrese a un único depósito; que solo haya arcos si el vehículo está activo que la distancia total no supere la autonomía; que  $Q[v]$  sea la suma de demandas de los clientes atendidos por  $v$  y no supere su capacidad, métricas de las variables de decisión y que no haya subtours gracias a las restricciones MTZ.

Luego de tener listo el modelo, se crea el solver GLPK, se reconstruyen las rutas por vehículo y se muestra un mapa interactivo con todos los resultados; finalmente se crea el archivo de validación.

Los datos del archivo de verificación fueron los siguientes:

|   | VehicleId | DepotId | InitialLoad | RouteSequence                 | ClientsServed | DemandsSatisfied   | TotalDistance | TotalTime  | FuelCost     |
|---|-----------|---------|-------------|-------------------------------|---------------|--------------------|---------------|------------|--------------|
| 0 | V001      | CD09    | 42.0        | CD09-C008-C007-C005-C003-CD09 | 4             | 10.0-12.0-5.0-15.0 | 42.417        | 117.583333 | 17284.927500 |
| 1 | V002      | CD05    | 26.0        | CD05-C006-C002-CD05           | 2             | 11.0-15.0          | 25.727        | 66.916667  | 13978.336667 |
| 2 | V003      | CD12    | 18.0        | CD12-C004-C001-CD12           | 2             | 6.0-12.0           | 29.149        | 79.066667  | 15837.623333 |
| 3 | V005      | CD11    | 15.0        | CD11-C009-CD11                | 1             | 15.0               | 10.427        | 25.666667  | 5665.336667  |

La implementación de este caso queda cargado en los entregables de la actividad

### c. Caso 3 (20%):

Para la construcción de la matriz de distancias y tiempos en el grafo, inicialmente se planificó utilizar el servicio de Google Distance Matrix API como en los otros casos, con el fin de obtener mediciones realistas basadas en la red vial de Bogotá. Sin embargo, durante



el desarrollo del proyecto se agotaron los créditos disponibles en GCP, lo cual imposibilitó continuar realizando consultas masivas a la API.

Ante este escenario, se optó por generar una aproximación razonable utilizando la distancia geodésica Haversine entre cada par de nodos. Esta distancia fue posteriormente escalada mediante un factor de holgura (1.25), diseñado para compensar la diferencia entre desplazamientos en línea recta y trayectorias reales por la red urbana. Por su parte, el tiempo de viaje se asignó mediante un valor sintético generado aleatoriamente dentro de un rango calibrado con observaciones previas de la ciudad, de modo que mantuviera coherencia con los patrones de movilidad sin depender de la API.

El tamaño del problema completo planteaba un desafío computacional significativo: el grafo original incluía decenas de depósitos, más de un centenar de clientes y un número muy grande de arcos potenciales. Resolver el modelo completo con Pyomo y el solver HiGHS requería manejar decenas de miles de variables binarias y restricciones, lo que rápidamente llevó a problemas de tiempo y memoria. Para mitigar esto, se adoptó una estrategia híbrida que prioriza la calidad de la solución sin sacrificar la factibilidad computacional.

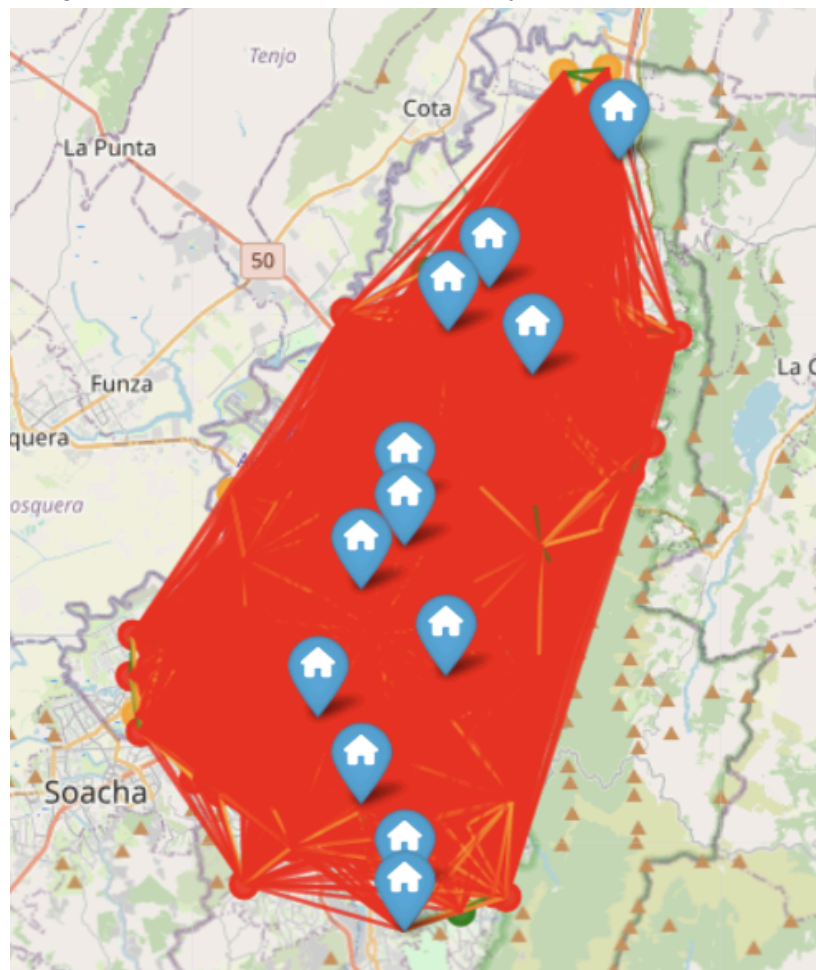
Primero, se dividió el conjunto de clientes en dos regiones según su latitud: Sur y Norte. Esta partición permite trabajar con subgrafos más pequeños y mejor condicionados. Para la

región Sur, se aplicó una heurística de reducción de arcos que elimina conexiones irrelevantes basadas en distancia Haversine, conservando únicamente arcos cercanos al depósito más probable y rutas geográficamente razonables. Esta reducción se evidencia en las imágenes comparativas del grafo original frente al grafo podado, donde se observa una disminución sustancial del número de aristas, manteniendo aún así la conectividad necesaria.

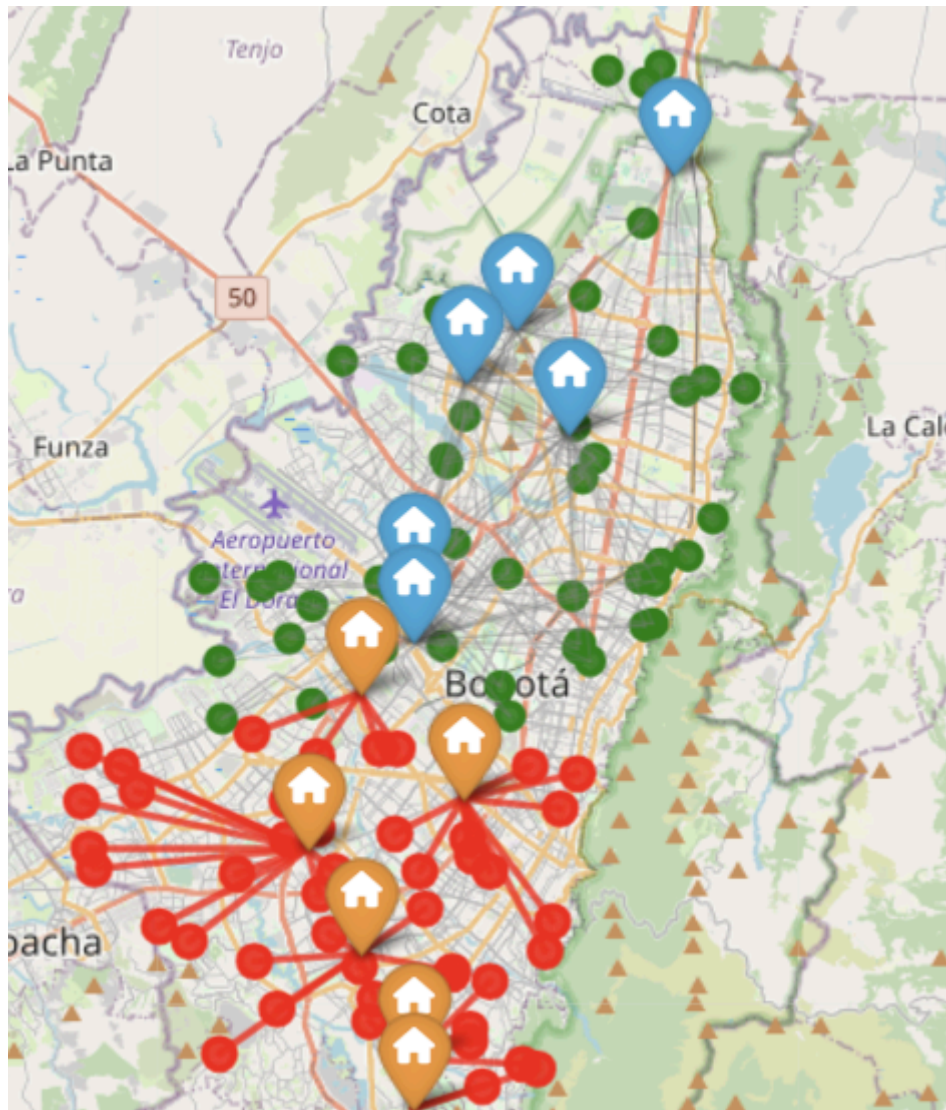
Sobre esta parte Sur, se utilizó una heurística estable de asignación de clientes a depósitos para determinar qué vehículos y recursos debían atender la demanda. Una vez cubierto el Sur, se tomaron los vehículos restantes y se construyó una subinstancia optimizable correspondiente únicamente a la región Norte. Esta subinstancia, al ser más pequeña y haber sido filtrada previamente, permitió ser resuelta con el solver HiGHS, logrando una solución óptima con un tiempo computacional razonable. Así, la solución global se obtiene como la combinación de:

1. Las rutas heurísticas del Sur.
2. Las rutas óptimas obtenidas por el solver en el Norte.

Este es el grafo originalmente, con todos los clientes y depósitos:



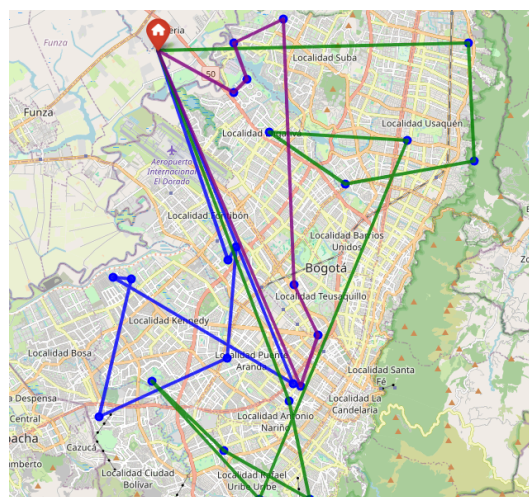
Luego de hacer la reducción de arcos y solucionar la parte sur:



Esto resulta ser un problema mucho más manejable para el solver el cual nos da los siguientes valores finales:

#### 4. Análisis de Resultados y Visualización (15%):

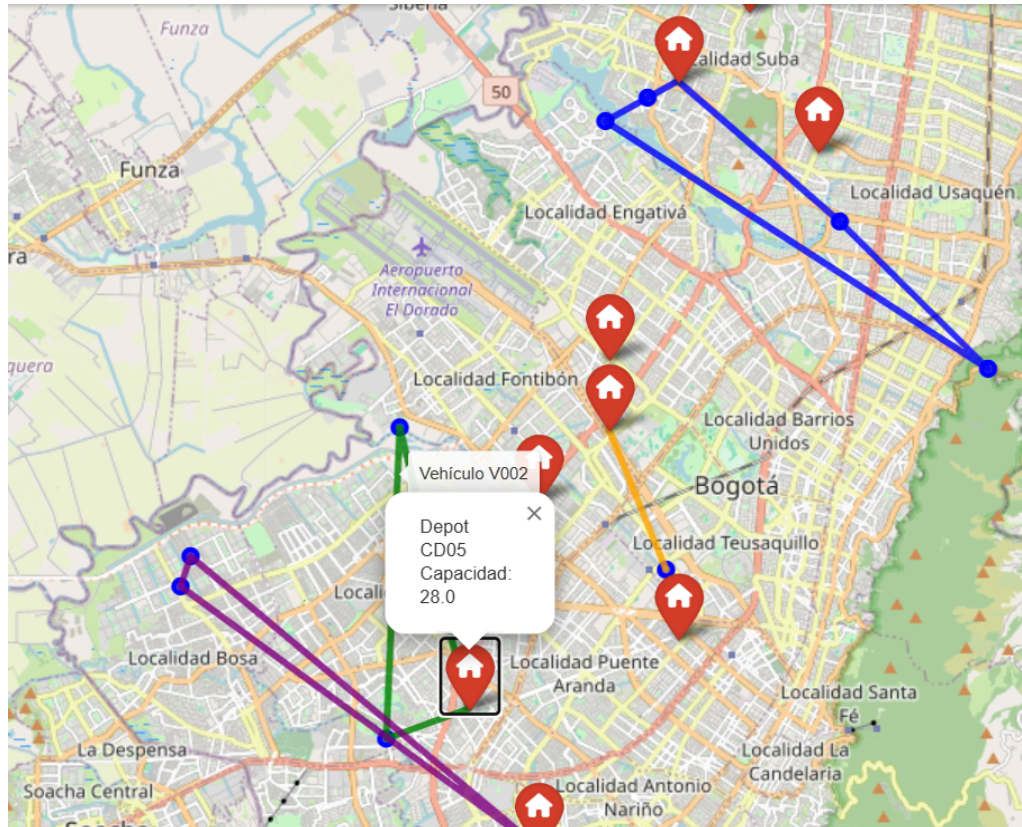
##### Caso 1:



En la solución se activaron los vehículos (V001, V002 y V008) como se puede ver en el mapa, los íconos de casa corresponden a los depósitos y las líneas de colores a las rutas de cada vehículo (azul para V001, verde para V002 y morado para V008). Cada ruta parte del mismo depósito, visita un subconjunto de clientes y regresa al mismo nodo, respetando la estructura clásica de un VRP con un solo depósito. Las rutas finales y más óptimas quedaron V001, atiende a los clientes C004, C008, C005, C011, C017, C006 y C018. V002 visita C001, C010, C024, C009, C016, C019, C003, C002, C007 y C021. Por último, V008 atiende C015, C022, C013, C012, C020, C023 y C014. El mapa confirma estos resultados del archivo de verificación, donde cada punto es un cliente que se ha visitado, además se ve que el camión se regresa siempre cumpliendo los requisitos. En términos de cargas, el archivo de verificación muestra que toda la demanda del sistema es atendida exactamente una vez. El vehículo V001 carga 126 unidades, V002 carga 138 y V008 113.

Finalmente, al combinar los costos fijos por vehículo utilizado, los costos proporcionales a la distancia, los costos por tiempo de operación y el costo de combustible, la función objetivo alcanza un valor mínimo de aproximadamente 436 865 pesos. Esto demuestra que con las condiciones reales de la ciudad y más variables del negocio, el modelo encontró una configuración en la que solo se usan 3 vehículos y para atender toda la demanda, esto ayuda a tomar decisiones estratégicas sobre el uso de los recursos de la empresa para atender todos los viajes necesarios para satisfacer toda la demanda.

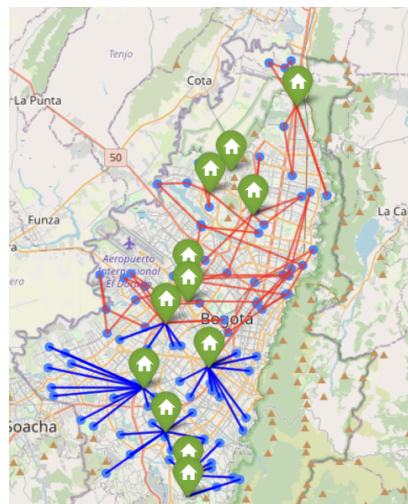
## Caso 2:



En la solución del modelo se activaron los vehículos (V001, V002, V003 y V005) como se puede ver en el mapa, los íconos de casa corresponden a los depósitos y las líneas de colores a las rutas de cada vehículo (azul para V001, verde para V002, morado para V003 y naranja para V005). Cada ruta parte de su depósito, visita un subconjunto de clientes y regresa al mismo nodo, respetando la estructura clásica de un VRP con múltiples depósitos. Las rutas finales y más óptimas quedaron V001 sale de CD09, atiende a los clientes C008, C007, C005 y C003 y regresa a CD09, mientras que V002 opera desde CD05 y visita C006 y C002, V003 parte de CD12 y atiende C004 y C001, y V005 desde CD11 cubre C009. El mapa confirma estos resultados del archivo de verificación, donde cada punto es un cliente que se visitó, además se ve que el camión se regresa siempre cumpliendo los requisitos. En términos de cargas, el archivo de verificación muestra que toda la demanda del sistema (101 unidades sumando C001–C009) es atendida exactamente una vez. El vehículo V001 carga 42 unidades, V002 carga 26, V003 18 y V005 15; en cada caso la columna InitialLoad coincide con la suma de las demandas listadas en DemandsSatisfied, lo que indica que la lógica de balance entre depósitos, vehículos y clientes está funcionando correctamente. Si se compara esa carga con las capacidades de la flota, ninguno de los vehículos se satura. En cuanto al desempeño operativo, la solución recorre en total unos 108 km y emplea alrededor de 289 minutos de tiempo de ruta entre todos los vehículos. V001 es el que más aporta en distancia, luego V003, V002 y luego el que menos recorre es V005 ya que solo atiende a un cliente; esto sugiere que el vehículo V001 y V003 son los más responsables de la alza del costo de operación ya que son los que más recorren. Lo bueno es que por ejemplo el V001 es el que más recorre pero es un camión pequeño, por lo que no consume mucha gasolina.

Finalmente, al combinar los costos fijos por vehículo utilizado, los costos proporcionales a la distancia, los costos por tiempo de operación y el costo de combustible, la función objetivo alcanza un valor mínimo de aproximadamente 558 702 pesos. Esto demuestra que con las condiciones reales de la ciudad y más variables del negocio, el modelo encontró una configuración en la que solo se usan 4 vehículos y 4 depósitos para atender toda la demanda, esto ayuda a tomar decisiones estratégicas sobre el uso de los recursos de la empresa para atender todos los viajes necesarios para satisfacer toda la demanda

### Caso 3:



La solución final obtenida para el Caso 3 evidencia un comportamiento diferenciado entre la zona sur y la zona norte de Bogotá, resultado directo tanto de la estructura espacial de los clientes como de las restricciones computacionales y metodológicas del proyecto. La representación geográfica muestra claramente dos patrones: en el sur, donde se aplicó una heurística tipo “estrella”, las rutas aparecen como conexiones directas entre cada depósito y sus clientes asignados, generando un conjunto de trayectorias radiales (líneas azules) que parten y regresan al mismo punto. Este patrón es coherente con la naturaleza del algoritmo aplicado para la asignación sur, que priorizó el cumplimiento de capacidad por depósito y la proximidad simple entre nodo-cliente y nodo-depósito. En contraste, la región norte exhibe trayectorias más complejas (líneas rojas), con múltiples clientes conectados de manera secuencial formando rutas completas optimizadas por HiGHS. En esta zona se observa el carácter completo del VRP: vehículos recorriendo circuitos más largos, visitando varios clientes en un solo tour y cerrando las rutas en un depósito. La visualización sugiere que el solver logró consolidar clientes geográficamente cercanos en tours eficientes, minimizando redundancias y evitando patrones radiales innecesarios. Como resultado, la solución total es una composición híbrida: eficiente y estructurada en el norte, operativa y simplificada en el sur. Esta dualidad permite atender todo el conjunto de clientes preservando la factibilidad global del sistema y respetando la heterogeneidad espacial.

### **Análisis de sensibilidad:**

La sensibilidad del modelo está dominada por tres grupos de parámetros:

1. **Costo fijo por vehículo  $C_{fixed}$**

Es el parámetro que más afecta la estructura de la solución.

Aumentarlo reduce el número de vehículos usados y alarga las rutas; disminuirlo hace que aparezcan rutas más cortas y más vehículos activos. Pequeñas variaciones pueden producir saltos discretos en  $Z$  porque cambian cuántos vehículos entran en servicio.

2. **Costo efectivo por kilómetro ( $C_{dist} + fuel\_cost\_per\_km$ )**

Es el parámetro que más afecta la longitud de los recorridos.

Si aumenta, se favorecen rutas cortas y asignaciones a depósitos cercanos; si disminuye, el modelo tolera rutas más largas para reducir el número de vehículos.

3. **Costo por unidad de tiempo  $C_{time}$**

Afecta principalmente la selección de rutas con menor tiempo, pero tiene menor influencia estructural mientras su magnitud sea pequeña comparada con los costos por distancia y por vehículo. Solo cuando  $C_{time}$  se vuelve comparable en magnitud al costo por km, se observan cambios significativos.

Esto se puede ver en el notebook con las gráficas y el análisis de la última celda del caso 3.