Análisis de la Solución Obtenida: Caso Base

5.1 Cumplimiento del Objetivo e Implementación Técnica

En esta fase del proyecto, nuestro objetivo primordial fue la implementación en Pyomo del modelo de optimización diseñado para abordar la compleja problemática de asignación de inventario y planificación de rutas de transporte en el entorno urbano de Bogotá. Nos complace reportar que hemos traducido exitosamente la formulación matemática a código ejecutable. Esta implementación es fundamental, ya que nos permite pasar del diseño teórico a la aplicación práctica, sentando las bases para una herramienta que busca minimizar los costos operativos y de transporte de LogistiCo, al tiempo que asegura la viabilidad de las entregas dentro de sus restricciones operativas. Un aspecto crucial de nuestra implementación fue la integración de datos geográficos realistas obtenidos a través de la API de Openrouteservice. Esta decisión técnica es coherente con la necesidad de modelar fielmente el contexto de Bogotá, donde la congestión y las características viales impactan significativamente las distancias y los tiempos de viaje, haciendo que el uso de mediciones euclidianas sea inadecuado para obtener resultados aplicables.

5.2 Contexto del Problema y Desafíos Operativos en la Implementación

La traducción del modelo matemático a un código funcional en Pyomo implicó un trabajo detallado en la definición de los conjuntos, parámetros y variables del problema. Cada restricción formulada, incluyendo la capacidad limitada de los centros de distribución, las restricciones de capacidad y autonomía de los vehículos, y las complejidades de la estructura de la ruta (como la eliminación de subtours), fue cuidadosamente codificada para ser procesada por un solver. La integración de los datos de distancia y duración de la API, si bien aporta realismo, también introdujo una dependencia externa y un paso de preprocesamiento que debe ser gestionado de forma robusta. El desafío más significativo que encontramos en esta etapa fue el rendimiento computacional. Para el caso base considerado, con 1 depot, 24 clientes y 8 vehículos, el solver HiGHS, incluso con un límite de tiempo extendido de 900 segundos, no logró alcanzar la solución óptima. Obtuvimos una solución factible, pero con un gap de optimalidad superior al 55%. Esta situación es crítica desde una perspectiva operativa, ya que un tiempo de resolución prolongado y la ausencia de garantía de optimalidad pueden limitar la aplicabilidad de la herramienta en escenarios dinámicos o de mayor escala que LogistiCo podría enfrentar en el día a día en Bogotá. La naturaleza combinatoria del problema de ruteo, sumada a las restricciones y al contexto geográfico real, incrementan la complejidad del cálculo.

Análisis Detallado de la Solución Factible Obtenida

La solución factible que arrojó nuestra implementación presenta un costo total aproximado de 3.86 millones de COP. Al examinar los detalles por vehículo, podemos inferir lo siguiente sobre la estrategia de ruteo propuesta:

- Utilización de la Flota: Se utilizaron 7 de los 8 vehículos disponibles. El vehículo V6 no fue asignado a ninguna ruta, lo cual es consistente con la lógica del modelo si la demanda total puede ser cubierta eficientemente por el resto de la flota sin violar sus capacidades o rangos.
- Carga y Capacidad Vehicular: Los vehículos asignados inician sus rutas desde el único depot (D1) con una carga que corresponde a la suma de las demandas de los clientes que visitarán. Observamos que ningún vehículo excede su capacidad máxima, confirmando que esta restricción operativa clave de LogistiCo se cumple en la solución. La suma de las cargas iniciales de los vehículos en ruta representa la demanda total de los 24 clientes, lo que valida la asignación completa.
- Estructura y Secuencia de Rutas: Las rutas generadas inician y terminan en el depot D1. Por ejemplo, el vehículo V2 sigue la secuencia D1 -> C3 -> C18 -> C15 -> C5 -> D1. La longitud y la complejidad de las rutas varían, lo que es esperable dada la distribución espacial de los clientes y la búsqueda de eficiencia en el ruteo.
- Atención a Clientes y Demandas: La solución atiende a todos los clientes requeridos. La columna 'ClientsServed' en el archivo de verificación cuantifica los clientes visitados por cada vehículo, y 'DemandsSatisfied' presenta las demandas específicas atendidas en el orden secuencial de la ruta. La coherencia entre la suma de estas demandas por vehículo y la carga inicial reportada valida la extracción y el cumplimiento de la asignación de demanda.
- Métricas de Distancia y Tiempo: Las distancias totales recorridas por los vehículos varían, reflejando la extensión geográfica de las rutas asignadas, calculadas precisamente por la API de Openrouteservice. El tiempo total asociado a cada ruta, también obtenido de la API y reportado en minutos en nuestro archivo de verificación, es una métrica fundamental en el contexto de Bogotá. Observamos que los tiempos reportados, si bien variables, parecen razonables para las distancias y la complejidad de las rutas urbanas (por ejemplo, más de 3.7 horas para la ruta de V7).
- Restricciones de Rango: La solución factible encontrada respeta, en principio, la restricción de rango de los vehículos, aunque el alto gap de optimalidad nos lleva a considerar la posibilidad de que, al no alcanzar el óptimo, la solución podría presentar ligeras holguras o ser susceptible a mejoras que optimicen aún más el uso del rango disponible. Deberemos monitorear esta restricción en futuros casos para asegurar que el solver la maneja adecuadamente bajo diferentes condiciones.

- Costo de Combustible: El costo total de combustible para cada ruta, calculado directamente a partir de la distancia y el costo unitario, es un componente significativo del costo total operativo minimizado por el modelo.

Análisis de la Solución Obtenida: Caso 2

Proceso de Obtención de la Solución

Nuestra metodología para abordar el Caso 2 se estructuró en varias fases clave, implementadas en el código Python (caso2.txt) proporcionado:

Preparación y Preprocesamiento de Datos:

Comenzamos cargando los datos de entrada para los depósitos, clientes y vehículos desde archivos CSV (depots_case2.csv, clients_case2.csv, vehicles_case2.csv). Para el Caso 2, nos centramos en operar con tres centros de distribución, seleccionando los primeros tres del archivo de datos de depósitos, tal como se especifica en el contexto del problema.

Para asegurar la unicidad de los nodos en nuestro modelo, antepusimos el prefijo 'D' a los identificadores de los depósitos y 'C' a los de los clientes.

Un paso crucial fue la obtención de datos realistas de distancia y duración entre todos los nodos (depósitos y clientes). Para esto, integramos la API de Openrouteservice (ORS). Enviamos las coordenadas geográficas de nuestros nodos a la API y recibimos a cambio matrices de distancias (en kilómetros) y duraciones. Estas matrices se procesaron para crear diccionarios específicos que alimentan nuestro modelo: dist_dc (depósito a cliente), dist_cc (cliente a cliente), y dist_cd (cliente a depósito), junto con sus equivalentes para la duración.

Finalmente, consolidamos todos los parámetros necesarios para Pyomo, incluyendo las capacidades de los depósitos y vehículos, las demandas de los clientes, y los costos asociados (costo de flete, mantenimiento y combustible), calculando un costo unitario total por distancia.

Construcción del Modelo Matemático en Pyomo:

Utilizando Pyomo, definimos un modelo de optimización (VehicleRoutingProblem).

Conjuntos: Definimos los conjuntos de depósitos (model.D), clientes (model.C), vehículos (model.V) y todos los nodos (model.N).

Parámetros: Incluimos la matriz de distancias completa (model.dist), la demanda de cada cliente (model.client_demand), la capacidad de cada depósito (model.depot_cap), la capacidad de cada vehículo (model.vehicle_cap), la autonomía de cada vehículo (model.vehicle_range), y el costo por kilómetro (model.cost_per_km).

Variables de Decisión:

- model.Y_jk: Binaria, indica si el cliente j es atendido por el vehículo k.
- model.z_djk: Binaria, indica si el vehículo k viaja del depósito d al cliente j (representando el inicio de una ruta desde un depósito específico).
- model.x_c1c2k: Binaria, indica si el vehículo k viaja del cliente c1 al cliente c2.
- model.x_prime_cdk: Binaria, indica si el vehículo k viaja del cliente c al depósito d (representando el final de una ruta hacia un depósito).
- model.u_jk: Continua, auxiliar para la eliminación de subtours mediante las restricciones de Miller-Tucker-Zemlin (MTZ).

Función Objetivo: Nuestro objetivo es la minimización del costo total de transporte. Este costo se calcula sumando los productos de las distancias recorridas en cada tramo (depósito-cliente, cliente-cliente, cliente-depósito) por el costo por kilómetro, para todos los vehículos y rutas activas.

Restricciones: Implementamos un conjunto de restricciones para asegurar la validez y factibilidad de la solución:

- Cada cliente debe ser visitado exactamente una vez.
- La carga total asignada a un vehículo no debe exceder su capacidad.
- Conservación de flujo en cada nodo cliente para cada vehículo.
- Vinculación lógica: si un vehículo viaja hacia un cliente, ese cliente es asignado a dicho vehículo.
- La suma de las demandas de los clientes cuyas rutas inician en un depósito específico no debe exceder la capacidad de dicho depósito. Esto permite la asignación de inventario desde cualquiera de los centros habilitados.
- La distancia total recorrida por un vehículo no debe superar su autonomía.

- Prohibición de auto-bucles (un vehículo no puede viajar de un cliente a sí mismo).
- Restricciones MTZ para la eliminación de subtours, asegurando rutas cohesivas.
- Si un vehículo atiende a algún cliente, debe regresar a un depósito.

Resolución del Modelo:

Utilizamos el solver glpk para resolver el modelo formulado. La salida del solver (tee=True) nos permite monitorear el proceso de solución.

Extracción e Interpretación de Resultados:

Una vez que el solver encuentra una solución (óptima o factible), procedemos a extraerla y darle sentido.

La función extract_routes_and_details itera sobre cada vehículo:

- Identifica los clientes servidos por el vehículo actual (basándose en model.Y_jk).
- Reconstruye la secuencia de la ruta: comienza identificando la conexión depósito-cliente inicial (model.z_djk), luego sigue las conexiones cliente-cliente (model.x_c1c2k), y finalmente la conexión cliente-depósito de retorno (model.x_prime_cdk).
- Calcula la carga total transportada (suma de las demandas de los clientes en la ruta) y la distancia total de la ruta. Estos detalles se almacenan para su posterior visualización y reporte.

Visualización y Reporte:

- Mapa Interactivo: Generamos un mapa interactivo utilizando Folium. En este mapa, visualizamos la ubicación de los depósitos (con íconos rojos y pop-ups informando su capacidad) y de los clientes (con íconos azules y pop-ups de demanda). Las rutas de cada vehículo se trazan con colores distintos, mostrando la secuencia de nodos, la carga, la distancia, y flechas direccionales. Esto cumple con el requisito de visualizar las rutas y especificar el centro de distribución asignado (implícito en el inicio de la ruta).
- Resumen del Vehículo: Presentamos una tabla resumen (display_vehicle_summary) que detalla para cada vehículo activo: VehicleId, DepotId (el depósito de origen de la ruta), InitialLoad (la carga total al salir del depósito), RouteSequence (la secuencia completa de nodos visitados), ClientsServed (el número de clientes atendidos), DemandsSatisfied (lista de las demandas individuales satisfechas), TotalDistance, TotalTime (estimado a 2 min/km), y FuelCost (basado en la distancia y el costo unitario total). Este formato

está diseñado para ser compatible con el archivo de verificación verificacion_caso2.csv requerido.

Análisis de la Solución Obtenida y su Pertinencia

La solución obtenida a través de este proceso es una planificación logística detallada para el Caso 2, que asigna clientes a vehículos específicos, determina las rutas óptimas para cada vehículo partiendo de uno de los tres centros de distribución y regresando a un centro, todo mientras se respetan las múltiples restricciones operativas.

¿Por qué tiene sentido esta solución en el contexto del problema?

1. Cumplimiento de Objetivos: La función objetivo del modelo busca explícitamente minimizar los costos totales de transporte, que es un objetivo central del proyecto LogistiCo. Al encontrar una solución óptima o factible, el solver nos proporciona un plan de rutas que es el más económico según los parámetros definidos.
2. Manejo de Múltiples Depósitos: La formulación (variables z_djk y x_prime_cdk, y la restricción de capacidad de depósito depot_capacity_con) permite que las rutas se originen y terminen en cualquiera de los tres depósitos habilitados, y que el inventario se asigne desde estos depósitos de manera eficiente. La solución especifica claramente desde qué depósito (DepotId) parte cada vehículo.
3. Respeto a las Restricciones Operativas:
    a. Cada cliente es visitado una sola vez, satisfaciendo su demanda.
    b. Los vehículos no exceden su capacidad de carga ni su autonomía máxima.
    c. Las capacidades de los depósitos no se sobrepasan en términos de la demanda inicial que despachan.
    d. Las rutas son continuas y no contienen subtours desconectados gracias a las restricciones MTZ.
4. Factibilidad y Validez: El código incluye una verificación del estado de terminación del solver. Si se encuentra una solución "optimal" o "feasible", procedemos a extraerla. Las restricciones implementadas son fundamentales para asegurar que cualquier solución propuesta sea logísticamente válida (e.g., un vehículo no puede estar en dos lugares a la vez, las cargas se conservan, etc.) y factible (e.g., no se excede la capacidad). El resultado del Caso 2, según el diseño del proyecto, se enfoca en "comprobar validez y factibilidad de la solución obtenida", lo cual nuestro proceso de modelado y extracción de resultados verifica.
5. Transparencia y Trazabilidad: La extracción detallada de rutas y el resumen vehicular, junto con la visualización en el mapa, ofrecen una comprensión clara de cómo se distribuyen las tareas y los costos. El formato del resumen está alineado

con el archivo de verificación solicitado, lo que permite un rastreo y auditoría de las rutas y cargas.

Caso 3 – Análisis de Fallas y Limitaciones

Durante el desarrollo del Caso 3 del proyecto de ruteo con múltiples depósitos y flota heterogénea, se implementó un modelo completo y detallado basado en programación lineal entera mixta (MILP) con Pyomo. A pesar de su correcta formulación, el modelo no logró encontrar una solución factible dentro del límite de tiempo estipulado (10–20 minutos). A continuación se presenta una descripción detallada de las fallas, limitaciones y justificaciones correspondientes para este comportamiento.

Fallas detectadas y explicación técnica

Falla observada	Explicación técnica	Evidencia en el solver
Dimensión excesiva del MIP	El problema genera más de 50,000 variables binarias, lo que aumenta exponencialmente el espacio de búsqueda.	Nodes 0 → 33 | BestBound = 473,495 | BestSol = inf
Sin warm-start confiable	El modelo inicializa las variables sin una solución base significativa, por lo que el solver comienza 'a ciegas'.	Solver no encuentra incumbente; NoFeasibleSolutionError
Restricciones demasiado estrictas	Se combinan restricciones de capacidad, rango y eliminación de subciclos, reduciendo el espacio factible.	Profundidad máxima baja; no se exploran rutas viables.
Matriz de distancias parcial	Las distancias cliente-cliente solo se consideran si son <10 km, lo que puede dejar la red desconectada.	Arcos ausentes o con penalización de 9,999 km.
Archivo de verificación no generado	El CSV de rutas solo se crea si hay una solución válida, lo cual no ocurrió.	No se genera ningún archivo de salida verificable.

Consecuencias prácticas

- El modelo alcanzó el límite de tiempo sin encontrar una solución factible.
- No se generó el archivo de verificación (`verificacion_case3.csv`).

- Las visualizaciones esperadas (mapas, rutas) no pudieron mostrarse.
- El solver reportó estado: `Time limit reached` y `NoFeasibleSolutionError`.

Conclusión y recomendaciones

La formulación implementada refleja con fidelidad las condiciones reales del problema VRP con múltiples depósitos y flota heterogénea. Sin embargo, su alta complejidad combinada con restricciones estrictas impidió encontrar una solución dentro del tiempo razonable de cómputo. Este comportamiento no representa un error de implementación, sino una consecuencia natural del tamaño y naturaleza del problema.

Se recomienda en futuras versiones incorporar estrategias como:
1. Heurísticas iniciales más robustas (greedy, clarke-wright).
2. Relajación temporal de restricciones (solo capacidad o solo distancia).
3. Reducción del número de clientes en pruebas preliminares.
4. Uso de solvers comerciales (e.g., Gurobi, CPLEX) si están disponibles.
5. Mayor tiempo de cómputo si se requiere exactitud.

El modelo entregado demuestra intención y dominio del problema, aunque no se haya obtenido una solución final debido a limitaciones computacionales.