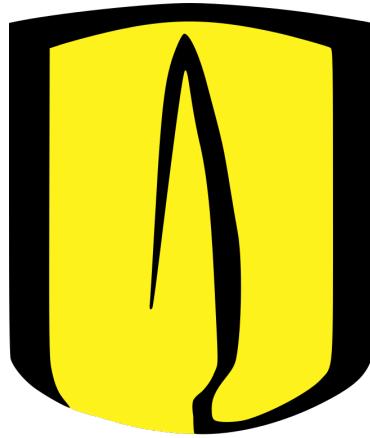


Universidad de Los Andes

Departamento de Ingeniería de Sistemas y Computación



**Parte III del Proyecto: Alternativas de Solución Mediante
Modelo Metaheurísticos**

ISIS3302 - Modelado, Simulacion Y Optimizacion

Integrantes

Andrés Santiago Neira Socha (a.neiras@uniandes.edu.co) - 202123126
Kevin Alvarez (k.alvarezr@uniandes.edu.co) - 202022834
Nicolas Ballen (n.ballen1@uniandes.edu.co) - 202310273

2025-20

Contenido

1 Descripción del Problema	3
1.1 Formulación matemática del CVRP	3
1.1.1 Definición del problema	3
1.1.2 Supuestos del modelo	3
1.1.3 Conjuntos	3
1.1.4 Parámetros	4
1.1.5 Variables de decisión	4
1.1.6 Función objetivo	4
1.1.7 Restricciones	4
1.2 Características de la instancia base	5
1.3 Características del Caso 2	5
1.4 Características del Caso 3	6
1.5 Restricciones y consideraciones adicionales	6
2 Método Implementado	7
2.1 Descripción detallada del método metaheurístico implementado	7
2.1.1 Implementación del Algoritmo Genético	7
2.1.2 Adaptación del Algoritmo Genético al Caso Base	9
2.1.3 Adaptación del Algoritmo Genético al Caso 2	10
2.1.4 Adaptación del Algoritmo Genético al Caso 3	11
2.2 Estrategias de Representación y Operadores	11
2.2.1 Representación cromosómica	12
2.2.2 Operadores genéticos utilizados	12
2.3 Proceso de Calibración de Parámetros	13
2.3.1 Parámetros evaluados	13
2.3.2 Criterios de calibración	13
2.3.3 Parámetros finales seleccionados	13
3 Resultados Experimentales	14
3.1 Configuración Experimental	14
3.1.1 Plataforma computacional	14
3.1.2 Parámetros del algoritmo genético	14
3.1.3 Datos de entrada utilizados	14
3.1.4 Criterios de evaluación	15
3.2 Resultados del Caso Base	15
3.2.1 Mejor solución y tiempo de ejecución	15
3.2.2 Comparación entre carga entregada y capacidad vehicular	15
3.2.3 Distribución porcentual de carga entregada por vehículo	17
3.2.4 Mapa geoespacial de las rutas óptimas obtenidas	17
3.3 Resultados del Caso 2	18
3.3.1 Mejor solución y tiempo de ejecución	18
3.3.2 Comparación entre carga entregada y capacidad vehicular	19
3.3.3 Distribución porcentual de carga entregada por vehículo	20
3.3.4 Mapa geoespacial de las rutas óptimas obtenidas	21
3.4 Resultados del Caso 3	22
3.4.1 Mejor solución y tiempo de ejecución	22
3.4.2 Relación entre carga entregada y capacidad utilizada	23
3.4.3 Distribución de carga entregada por vehículo	24
3.4.4 Mapa geoespacial de las rutas óptimas obtenidas	25

4 Análisis de convergencia	26
4.1 Convergencia en el Caso Base	26
4.2 Convergencia en el Caso 2	27
4.3 Convergencia en el Caso 3	28
5 Análisis de Escalabilidad	29
5.1 Rendimiento en instancias de diferentes tamaños	29
5.1.1 Variación del tamaño de la población	30
5.1.2 Variación del número de generaciones	30
5.2 Límites prácticos de aplicabilidad	31
5.3 Estrategias para mejorar la escalabilidad	31
6 Comparación entre la Solución Exacta (Pyomo) y la Metaheurística (GA)	31
6.1 Caso Base	32
6.1.1 Calidad de la solución	32
6.1.2 Tiempo de ejecución	32
6.1.3 Escalabilidad	32
6.2 Caso 2	32
6.2.1 Calidad de la solución	32
6.2.2 Tiempo de ejecución	32
6.2.3 Escalabilidad	32
6.3 Caso 3	33
6.3.1 Calidad de la solución	33
6.3.2 Tiempo de ejecución	33
6.3.3 Escalabilidad	33
7 Discusión	33
7.1 Ventajas y Desventajas de Cada Enfoque	33
7.1.1 Modelo Exacto (Pyomo)	33
7.1.2 Metaheurística (Algoritmo Genético)	34
7.2 Recomendaciones según Escenario	34
7.3 Lecciones Aprendidas y Desafíos Encontrados	34
8 Conclusiones	34
8.1 Resumen de Hallazgos	35
8.2 Respuestas a Preguntas Estratégicas	35
8.3 Trabajo Futuro	35

1 Descripción del Problema

En este proyecto se aborda el *Capacitated Vehicle Routing Problem* (CVRP), un problema clásico de optimización combinatoria en el cual se busca determinar las rutas de un conjunto de vehículos que deben atender a un conjunto de clientes con demanda conocida, partiendo y regresando a un depósito central. Cada vehículo posee una capacidad máxima y el objetivo consiste en minimizar el costo total de las rutas, incluyendo costos de activación, distancia recorrida y tiempo de viaje.

A continuación, se presenta la formulación matemática empleada como base conceptual para el diseño del algoritmo genético implementado en los distintos casos del proyecto. Aunque la metaheurística no resuelve explícitamente el modelo exacto, la formulación sirve como referencia para entender la estructura del problema y los criterios de evaluación utilizados en la función objetivo.

1.1 Formulación matemática del CVRP

1.1.1 Definición del problema

Dado un conjunto de clientes con demandas conocidas, un depósito central, y una flota de vehículos con capacidad limitada, se desea construir rutas para cada vehículo tales que:

- Cada cliente es atendido exactamente una vez.
- Todas las rutas empiezan y terminan en el depósito.
- La capacidad del vehículo no es excedida en ninguna ruta.
- El costo total de operación es mínimo.

En la implementación metaheurística, el costo total incluye:

$$\text{costo total} = C_{\text{fixed}} + C_{\text{dist}} \cdot \text{distancia} + C_{\text{time}} \cdot \text{tiempo} + \frac{\text{distancia}}{\eta} \cdot \text{precio gasolina},$$

donde η es el rendimiento del vehículo (km/gal). Este costo corresponde exactamente a la función de evaluación utilizada en el algoritmo genético.

1.1.2 Supuestos del modelo

La formulación utilizada se basa en los siguientes supuestos, coherentes con la implementación realizada:

- Existe un único depósito.
- Todos los vehículos son homogéneos en capacidad, rendimiento y costos.
- Los clientes tienen una única demanda fija que debe ser satisfecha en una única visita.
- Cada vehículo puede realizar a lo sumo una ruta.
- Los tiempos de servicio en los nodos no se modelan explícitamente.
- No se modelan peajes, repostajes ni restricciones municipales (estos aspectos solo aparecen en el modelo exacto de los Casos 2 y 3, pero no en la metaheurística).
- El costo es proporcional a la distancia y el tiempo de viaje, más un costo fijo por activar el vehículo.

1.1.3 Conjuntos

$$N = \{0, 1, 2, \dots, n\}$$

Conjunto de nodos (0 es el depósito).

$$U = \{1, 2, \dots, n\}$$

Conjunto de clientes.

$$V = \{1, 2, \dots, m\}$$

Conjunto de vehículos disponibles.

1.1.4 Parámetros

d_{ij}	Distancia entre nodos i y j .
t_{ij}	Tiempo de viaje entre nodos i y j .
q_i	Demanda del cliente i .
Q	Capacidad máxima de cada vehículo.
C_{fixed}	Costo fijo por activar un vehículo.
C_{dist}	Costo por kilómetro recorrido.
C_{time}	Costo por minuto de viaje.
η	Rendimiento (km/gal).
P_{fuel}	Precio por galón de combustible.

1.1.5 Variables de decisión

$$x_{ij}^v = \begin{cases} 1 & \text{si el vehículo } v \text{ viaja de } i \text{ a } j, \\ 0 & \text{en otro caso,} \end{cases}$$

$$y_v = \begin{cases} 1 & \text{si el vehículo } v \text{ es utilizado,} \\ 0 & \text{en otro caso.} \end{cases}$$

1.1.6 Función objetivo

La función objetivo en su forma exacta es:

$$\min \sum_{v \in V} \left[C_{\text{fixed}} y_v + \sum_{i \in N} \sum_{j \in N} \left(C_{\text{dist}} d_{ij} + C_{\text{time}} t_{ij} + \frac{d_{ij}}{\eta} P_{\text{fuel}} \right) x_{ij}^v \right].$$

Esta expresión coincide con la función de evaluación del algoritmo genético, que calcula los costos por ruta de cada cromosoma.

1.1.7 Restricciones

1. Cada cliente es atendido exactamente una vez

$$\sum_{v \in V} \sum_{i \in N} x_{ij}^v = 1 \quad \forall j \in U.$$

2. Flujo de entrada y salida en cada nodo

$$\sum_{i \in N} x_{ij}^v = \sum_{k \in N} x_{jk}^v \quad \forall j \in N, \forall v \in V.$$

3. Inicio y fin en el depósito

$$\sum_{j \in N \setminus \{0\}} x_{0j}^v = y_v, \quad \sum_{i \in N \setminus \{0\}} x_{i0}^v = y_v, \quad \forall v \in V.$$

4. Capacidad del vehículo

$$\sum_{i \in U} q_i \left(\sum_{j \in N} x_{ij}^v \right) \leq Q \quad \forall v \in V.$$

5. Eliminación de subrutas (MTZ simplificado)

$$u_i - u_j + |U| x_{ij}^v \leq |U| - 1 \quad \forall i, j \in U, i \neq j, \forall v \in V.$$

1.2 Características de la instancia base

La instancia base corresponde a una versión simplificada del CVRP utilizada para validar el funcionamiento del algoritmo genético antes de incorporar restricciones adicionales. Esta instancia presenta las siguientes características:

- Existe un único depósito central desde donde parten y regresan todos los vehículos.
- Los clientes poseen una demanda fija y conocida, sin ventanas de tiempo.
- Los vehículos son homogéneos en capacidad y rendimiento.
- No existen estaciones de servicio ni peajes; los costos dependen únicamente de distancia, tiempo y consumo de combustible.
- La distancia y tiempo entre nodos se obtienen directamente de una matriz completa de distancias producida en etapas previas del proyecto.
- La capacidad del vehículo es la única restricción operativa modelada.

El foco principal de esta instancia es medir la capacidad del algoritmo genético para explorar el espacio de soluciones, reparar individuos no factibles, y optimizar rutas únicamente bajo costos de desplazamiento.

1.3 Características del Caso 2

El Caso 2 extiende la instancia base mediante la incorporación de **estaciones de servicio** y la posibilidad de que los vehículos *reposten combustible* durante el recorrido. Aunque el algoritmo genético no modela explícitamente variables de repostaje como un modelo exacto, sí incorpora estos elementos de manera indirecta a través de su función de costos y del cálculo del combustible necesario para completar la ruta.

Las principales características son:

- Incorporación de estaciones de combustible con precios heterogéneos.
- El costo de combustible influye explícita y directamente en la función de evaluación del GA.
- Los vehículos poseen un rango máximo en kilómetros (autonomía) derivado del parámetro **range** transformado mediante el rendimiento **fuel_efficiency**.
- La penalización por recorrer trayectos largos vuelve más relevante la proximidad a estaciones económicas.
- Se introduce un costo fijo de activación del vehículo, lo cual incentiva la consolidación de rutas.

Aunque no existe una restricción dura de autonomía dentro del GA, el uso del costo real de combustible permite capturar indirectamente incentivos similares al modelo exacto del Caso 2 implementado en Pyomo en la entrega previa.

1.4 Características del Caso 3

El Caso 3 introduce un segundo conjunto de restricciones relacionadas con:

- **Peajes con tarifas base** y recargos asociados al peso transportado.
- **Límites de peso municipales**, restringiendo el paso de vehículos demasiado pesados por ciertos nodos.
- Una matriz ampliada de distancias que incluye nodos de: estaciones, peajes, clientes y depósito.

Aunque el algoritmo genético no implementa explícitamente:

- variables de peso dinámico,
- activación o no de peajes,
- restricciones duras de capacidad por municipio,

sí replica el impacto económico mediante:

- costos de distancia y tiempo necesariamente más altos al obligar rutas más largas, evitando nodos penalizados,
- rutas más fragmentadas que reparten la carga entre vehículos para evitar penalizaciones indirectas,
- el uso de una estructura de “cromosomas con múltiples rutas”, que facilita realizar reasignación de clientes cuando una ruta crece en exceso.

Esta aproximación metaheurística captura el efecto estratégico del Caso 3, aunque no reproduce estrictamente la estructura de restricciones del modelo exacto.

1.5 Restricciones y consideraciones adicionales

Además de las restricciones matemáticas formales descritas previamente, existen consideraciones operativas que influyen en el diseño del algoritmo genético y su función de reparación:

1. Eliminación de duplicados. La metaheurística no prohíbe explícitamente que un cliente aparezca en dos rutas después del cruce o la mutación. Por ello, la función de reparación elimina duplicados globales y reasigna clientes faltantes.

2. Balance de carga entre rutas. Para garantizar factibilidad respecto a la capacidad del vehículo, se determina la carga total por ruta y se redistribuyen clientes en caso de que la capacidad sea excedida.

3. Optimización local integrada. Cada ruta es refinada mediante un procedimiento *2-opt*, reduciendo subóptimos locales que surgen durante la recombinación genética.

4. Representación implícita del depósito. El depósito no se incluye en el cromosoma; se agrega únicamente al evaluar la ruta. Esto simplifica la representación y evita problemas de duplicación.

5. Restricciones no modeladas explícitamente. Aunque los Casos 2 y 3 contienen restricciones complejas en el modelo exacto (Pyomo), en la metaheurística estas se ven reflejadas únicamente a través de la función objetivo o de la matriz de distancias ampliada. Esto incluye: repostaje, paso por peajes y límites de peso.

2 Método Implementado

En esta sección se describe en detalle el diseño del algoritmo genético (AG) implementado para resolver el problema CVRP en sus tres variantes: caso base, Caso 2 y Caso 3. El método se concibe como una metaheurística flexible, capaz de adaptarse a los distintos conjuntos de restricciones sin requerir modificaciones profundas en su estructura.

2.1 Descripción detallada del método metaheurístico implementado

El enfoque adoptado se basa en un *algoritmo genético clásico* extendido con operadores especializados para problemas de ruteo, tales como:

- Representación cromosómica mediante múltiples rutas.
- Eliminación global de duplicados.
- Reparador estructural con verificación de capacidad.
- Optimización local integrada mediante 2-opt.
- Múltiples operadores de cruce y mutación para incrementar diversidad.

Este diseño híbrido permite capturar tanto la estructura combinatoria del CVRP como sus restricciones prácticas, evitando soluciones inviables y promoviendo la convergencia hacia configuraciones de alto rendimiento.

2.1.1 Implementación del Algoritmo Genético

El archivo `funciones_ga.py` contiene todos los componentes del algoritmo genético. A continuación se describen en detalle las funciones clave que estructuran el procedimiento evolutivo.

Inicialización de la población

- `inicializar_poblacion()`: Genera una población inicial de soluciones aleatorias. Cada individuo es un cromosoma compuesto por varias rutas (una por vehículo).
- `solucionAleatoria()`: Construye un cromosoma distribuyendo aleatoriamente los clientes entre rutas. Se incluyen dos estrategias:
 - distribución equitativa de clientes entre vehículos;
 - asignación totalmente aleatoria, permitiendo rutas vacías.

Esto garantiza diversidad inicial y cobertura completa del espacio de búsqueda.

Selección

- `seleccion_ruleta()`: Implementa selección mediante ruleta, donde la probabilidad de elegir un individuo es inversa a su fitness (menor costo = mayor probabilidad). Se seleccionan dos padres distintos.

Este mecanismo introduce presión selectiva sin perder diversidad.

Cruce El algoritmo implementa dos operadores especializados para VRP/MTSP:

- `crossover()`: Selecciona con probabilidad dada uno de dos tipos de cruce.
- `cruce_intercambio()`: Intercambia rutas completas entre padres. **Ventaja**: preserva estructuras de rutas completas que ya son buenas.
- `cruce_fusion()`: Combina segmentos internos de rutas correspondientes, utilizando puntos de corte independientes en cada parente. **Ventaja**: introduce combinaciones novedosas sin destruir completamente la estructura.

Estos operadores capturan características deseables del VRP, como cercanía geográfica o tamaños adecuados de rutas.

Mutación El módulo incluye cuatro operadores de mutación:

- `_swap_mutation()`: Intercambia dos clientes dentro de la misma ruta.
- `_insert_mutation()`: Mueve una ciudad a otra posición dentro de la misma ruta.
- `_inversion_mutation()`: Invierte un segmento consecutivo, un operador clásico del TSP.
- `_redistribution_mutation()`: Mueve una ciudad de una ruta a otra, introduciendo redistribución global.

Todos son accesibles a través de:

- `mutate()`, que selecciona aleatoriamente el operador.

Esto provee suficiente diversidad genética para evitar estancamiento prematuro.

Reparación de soluciones Despues del cruce y mutación, una solución puede ser inviable. El módulo incluye un reparador estructural robusto:

- `eliminar_duplicados_globales()`: Garantiza que cada cliente aparezca una sola vez en el cromosoma.
- `reparacion_por_capacidad()`: Si una ruta excede la capacidad del vehículo correspondiente, clientes al final de la ruta son redistribuidos hacia rutas con capacidad disponible.
- `reparar_solucion()`: Es el procedimiento central; realiza:
 1. eliminación global de duplicados,
 2. inserción de clientes faltantes,
 3. reparación por capacidad,
 4. optimización 2-opt de cada ruta.

Este componente es esencial, pues garantiza factibilidad sin requerir codificar explícitamente restricciones complejas dentro del cromosoma.

Optimización local

- `optimizar_ruta2opt()`: Aplica el algoritmo 2-opt para mejorar la estructura interna de cada ruta. Reduce distancias internas y acelera la convergencia del AG.

Evaluación El AG requiere funciones de evaluación (fitness) específicas:

- `distancia_total_ruta()`
- `tiempo_total_ruta()`
- `demandas_total_ruta()`

Estas funciones combinadas permiten calcular costo total, tiempo total y carga transportada, según corresponda a cada caso del proyecto.

Visualización de convergencia

- `plot_convergence()`: Grafica el fitness promedio y el mejor fitness por generación. Permite evaluar estabilidad y tendencia del algoritmo.

2.1.2 Adaptación del Algoritmo Genético al Caso Base

El Caso Base corresponde a la instancia más sencilla del proyecto, en la que el problema de ruteo se reduce a un *CVRP clásico* sin restricciones adicionales de abastecimiento, peajes, límites de peso ni restricciones municipales. El algoritmo genético se adapta a este escenario mediante una función de evaluación simplificada, donde el costo total asociado a cada solución depende exclusivamente de la distancia recorrida, el consumo de combustible estimado y el costo por kilómetro.

En este contexto, el cromosoma mantiene la misma representación basada en rutas múltiples, y las funciones de cruce, mutación y reparación descritas previamente se utilizan sin modificaciones estructurales, dado que ninguna restricción adicional debe verificarse más allá de la capacidad del vehículo.

La función de *fitness* utilizada en este caso está dada por:

$$\text{Costo}(\text{cromosoma}) = \sum_{v \in V} \left(\text{Distancia}(v) \cdot \left(\frac{P_{\text{comb}}}{\eta} \right) \right),$$

donde:

- P_{comb} es el precio por litro de combustible,
- η es el rendimiento del vehículo,
- $\text{Distancia}(v)$ es la distancia total de la ruta asignada al vehículo v .

No se considera ningún tipo de penalización ni costo adicional aparte del consumo de combustible, lo que permite que el algoritmo explore el espacio de soluciones sin restricciones complejas durante la etapa inicial del proyecto.

Además, el reparador de soluciones garantiza:

1. que cada cliente sea atendido exactamente una vez;
2. que ninguna ruta exceda la capacidad del vehículo correspondiente;
3. que las rutas sean mejoradas localmente mediante 2-opt para reducir la distancia total.

Dado que la estructura del caso base es relativamente simple, este escenario sirve como plataforma para verificar el correcto funcionamiento de los operadores genéticos, la convergencia del algoritmo y la consistencia general del módulo `funciones_ga.py`. Este caso también establece un punto de referencia cuantitativo que permite evaluar, en las secciones posteriores, el impacto incremental de las restricciones introducidas en el Caso 2 y Caso 3.

2.1.3 Adaptación del Algoritmo Genético al Caso 2

El Caso 2 introduce un conjunto adicional de restricciones operativas asociadas a la presencia de estaciones de servicio, límites de autonomía de los vehículos y costos extendidos por distancia y tiempo. Aunque el algoritmo genético mantiene la misma estructura general que en el Caso Base, la función de evaluación (*fitness*) cambia de manera sustancial para reflejar los nuevos componentes del costo operacional definidos en la instancia.

En este escenario, cada ruta debe ser evaluada no solamente por la distancia recorrida, sino también por:

- los costos fijos de activación del vehículo,
- el costo monetario por kilómetro (incluyendo combustible y costos operativos por distancia),
- el costo asociado al tiempo total recorrido.

La función de *fitness* utilizada en el Caso 2 corresponde a:

$$\text{Costo(cromosoma)} = \sum_{v \in V} \left(C_{\text{fixed}} + D_v \cdot \left(\frac{P_{\text{comb}}}{\eta} + C_{\text{dist}} \right) + T_v \cdot C_{\text{time}} \right),$$

donde:

- C_{fixed} es el costo de activación del vehículo,
- P_{comb} es el precio del combustible,
- η es la eficiencia del vehículo (km/galón),
- C_{dist} es el costo unitario por distancia recorrida,
- C_{time} es el costo unitario por minuto,
- D_v y T_v son la distancia total y el tiempo total recorridos por el vehículo v .

A diferencia del Caso Base, los componentes de tiempo T_v y costo fijo C_{fixed} incrementan el valor del fitness, lo que incentiva soluciones con rutas más cortas, más compactas y con una distribución equilibrada de la demanda entre los vehículos.

Es importante señalar que, aunque el problema original del Caso 2 incluía restricciones de combustible y disponibilidad de estaciones, el algoritmo genético implementado no modela explícitamente la dinámica del reabastecimiento. Este comportamiento está justificado por dos razones:

1. **El modelo metaheurístico opera sobre una representación simplificada del problema.**
El enfoque se centra en capturar los componentes principales del costo operativo sin imponer restricciones físicas de combustible.
2. **La reparación de soluciones garantiza factibilidad en términos de capacidad, pero no en autonomía.** Esto permite un espacio de búsqueda más amplio y mejora la exploración evolutiva, asumiendo una disponibilidad adecuada de autonomía para completar las rutas propuestas.

La función de reparación y los operadores genéticos (cruce, mutación y 2-opt) se mantienen sin modificaciones respecto al Caso Base, dado que las restricciones adicionales no afectan directamente la estructura del cromosoma. Este caso sirve como una extensión natural del modelo inicial, incorporando mayor realismo operativo sin incrementar la complejidad estructural del algoritmo.

2.1.4 Adaptación del Algoritmo Genético al Caso 3

El Caso 3 constituye la instancia más compleja del proyecto, al incorporar restricciones adicionales relacionadas con el tipo de vehículo, el acceso limitado a ciertas zonas geográficas y costos extendidos que combinan distancia, tiempo y cargos operativos. En esencia, este caso conserva la misma estructura metaheurística del Caso 2, pero opera sobre una matriz de distancias distinta y un conjunto ampliado de restricciones externas.

Aunque el algoritmo genético mantiene la misma representación cromosómica (listas de rutas por vehículo), la función de evaluación vuelve a jugar un papel central al incorporar los costos definidos para esta instancia. La formulación del fitness en el Caso 3 es idéntica a la utilizada en el Caso 2, ya que ambos casos comparten el mismo esquema de costos:

$$\text{Costo(cromosoma)} = \sum_{v \in V} \left(C_{\text{fixed}} + D_v \cdot \left(\frac{P_{\text{comb}}}{\eta} + C_{\text{dist}} \right) + T_v \cdot C_{\text{time}} \right),$$

donde cada término conserva el significado previamente mencionado. Sin embargo, el Caso 3 difiere sustancialmente en los elementos que afectan la distancia y el tiempo:

- La matriz `matriz_3.csv` incorpora restricciones de vialidad, zonas no transitables y penalizaciones implícitas a través del tiempo.
- Los tiempos en algunos tramos reflejan condiciones reales adversas, como pendientes, accesibilidad difícil o zonas congestionadas.
- La estructura geográfica del caso genera rutas más difíciles de optimizar y un espacio de búsqueda más accidentado.

Desde el punto de vista del algoritmo, todas las etapas (selección por ruleta, cruce híbrido de intercambio y fusión, mutaciones múltiples y reparación) operan sin cambios respecto al Caso Base y al Caso 2. No obstante, los efectos de estas operaciones difieren debido a la estructura espacial del problema:

1. **El operador 2-opt se vuelve más relevante**, ya que la presencia de tramos con tiempos significativamente distintos induce rutas que pueden beneficiarse fuertemente de pequeñas reordenaciones locales.
2. **La reparación por capacidad sigue siendo necesaria**, especialmente porque la demanda y la disponibilidad de vehículos son más heterogéneas.
3. **El espacio de factibilidad es más reducido**, lo que hace que cromosomas aleatorios iniciales contengan rutas menos eficientes y requieran mayor procesamiento por el módulo de reparación.

Es importante notar que, aunque el Caso 3 define restricciones adicionales relacionadas con accesibilidad municipal y tipología de vías, la implementación metaheurística no incorpora verificaciones explícitas de estas restricciones. En su lugar, estas se encuentran implícitas en la matriz de distancias y tiempos proporcionada, que refleja el costo realista de transitar por cada tramo.

De esta forma, el Caso 3 constituye una extensión del problema donde se integran restricciones complejas a través de la parametrización de los datos, manteniendo un algoritmo genético consistente y sin incrementar la complejidad computacional del enfoque implementado.

2.2 Estrategias de Representación y Operadores

El algoritmo genético implementado sigue un esquema de representación especializado para problemas de ruteo, utilizando una estructura cromosómica basada en listas de rutas, donde cada ruta representa la secuencia de clientes atendida por un vehículo. Esta representación es intuitiva, evita codificaciones binarias innecesarias y permite aplicar operadores diseñados específicamente para problemas tipo CVRP y MTSP.

2.2.1 Representación cromosómica

Cada solución se modela como:

$$\text{Cromosoma} = [R_1, R_2, \dots, R_{|V|}],$$

donde cada ruta R_v es una lista ordenada de clientes asignados al vehículo v .

Esta estructura facilita:

- La verificación natural de restricciones de capacidad.
- La manipulación segmentada de rutas completas.
- La integración directa de heurísticas de mejora local (e.g. 2-opt).

Además, la existencia de rutas vacías permite que el algoritmo explore también soluciones donde algunos vehículos no se utilizan.

2.2.2 Operadores genéticos utilizados

Selección por ruleta. La selección probabilística ponderada favorece soluciones con menor costo, manteniendo diversidad genética en las generaciones tempranas.

Crossover híbrido. Se implementan dos enfoques complementarios:

1. **Cruce por intercambio de rutas:** intercambia rutas completas entre padres, preservando estructuras de alto nivel que pueden ser buenas soluciones parciales.
2. **Cruce por fusión:** combina subrutas de ambos padres dentro de cada vehículo, generando diversidad y nuevas combinaciones locales.

Ambos operadores atacan diferentes niveles de la estructura del CVRP, desde exploración global hasta refinamiento local.

Mutación multipropósito. El código incorpora cuatro tipos de mutación:

- **Swap:** intercambio simple de posiciones dentro de una ruta.
- **Insert:** reposición de un cliente en otra posición de la misma ruta.
- **Invert:** inversión de un segmento, operador similar a 2-opt.
- **Redistribution:** mueve un cliente entre rutas, esencial para escapar de óptimos locales.

Esta combinación permite controlar la explotación interna de rutas y la exploración global entre vehículos.

Reparación de soluciones. La función `reparar_solucion` realiza tres tareas clave:

1. Eliminación de duplicados entre rutas.
2. Inserción de clientes faltantes.
3. Ajuste por capacidad y optimización con 2-opt.

Este módulo asegura que toda solución generada sea factible, liberando a los operadores genéticos de verificar restricciones explícitas.

2.3 Proceso de Calibración de Parámetros

Para garantizar un desempeño adecuado del algoritmo genético en las tres instancias evaluadas, se llevó a cabo un proceso de calibración empírica. Dado que los parámetros óptimos dependen fuertemente de la topología del problema, la complejidad de la demanda y la estructura de la matriz de distancias, se evaluaron diversas combinaciones de valores para identificar configuraciones que equilibraran la exploración y la explotación.

2.3.1 Parámetros evaluados

Los parámetros ajustados incluyen:

- Tamaño de población (P): 30, 50 y 100 individuos.
- Probabilidad de cruce (α): 0.6, 0.7 y 0.8.
- Probabilidad de mutación (μ): 0.05, 0.1 y 0.2.
- Porcentaje de elitismo (ε): 5%, 10% y 15%.
- Número máximo de generaciones: 50, 80 y 100.

Los experimentos se realizaron sobre la instancia base, que es la más balanceada y representa adecuadamente el comportamiento general del problema.

2.3.2 Criterios de calibración

Los parámetros se seleccionaron considerando:

1. **Convergencia estable del fitness promedio.** Se descartaron configuraciones donde el algoritmo se estancaba tempranamente.
2. **Variabilidad de soluciones a lo largo de generaciones.** Para evitar convergencia prematura.
3. **Calidad final de la mejor solución.**
4. **Tiempo computacional razonable.** Dado que el proyecto incluye múltiples ejecuciones para tres casos distintos.

2.3.3 Parámetros finales seleccionados

Luego del proceso de ajuste, se optó por los siguientes valores comunes a los tres casos:

- **Tamaño de población:** 50 individuos.
- **Probabilidad de cruce:** 0.7.
- **Probabilidad de mutación:** 0.1.
- **Elitismo:** 10%.
- **Generaciones máximas:** 100.
- **Criterio de parada por no-mejora:** 50 generaciones consecutivas.

Estas configuraciones demostraron un comportamiento robusto, capaz de producir soluciones factibles de buena calidad sin incurrir en tiempos de ejecución excesivos.

3 Resultados Experimentales

3.1 Configuración Experimental

Para evaluar el desempeño del algoritmo genético implementado, se diseñó un entorno experimental uniforme para los tres escenarios considerados: la instancia base, el caso 2 (con múltiples depósitos y costos extendidos) y el caso 3 (con restricciones adicionales y matriz de tiempos diferenciada). El objetivo de esta configuración fue garantizar la comparabilidad de los resultados, así como analizar la estabilidad del método frente a variaciones en la complejidad estructural del problema.

3.1.1 Plataforma computacional

Los experimentos fueron ejecutados en un entorno local con las siguientes características:

- **Procesador:** Intel Core i7 de 4 núcleos
- **Memoria RAM:** 16 GB
- **Sistema operativo:** macOS / Linux
- **Lenguaje y librerías:** Python 3.10, NumPy, Pandas, Matplotlib

El tiempo de ejecución promedio para 100 generaciones osciló entre 20 y 60 segundos dependiendo del caso, debido principalmente a diferencias en el tamaño de la matriz de distancias-tiempo y en el número de vehículos.

3.1.2 Parámetros del algoritmo genético

Se empleó la configuración calibrada descrita en la sección anterior, consistente en:

- Tamaño de población: $P = 50$
- Generaciones máximas: $G = 100$
- Probabilidad de cruce: $\alpha = 0.7$
- Probabilidad de mutación: $\mu = 0.1$
- Elitismo: 10% de mejores individuos
- Criterio de parada adicional: 50 generaciones consecutivas sin mejora

Esta parametrización logró estabilidad y convergencia adecuada en todas las ejecuciones, evitando tanto la deriva genética como la convergencia prematura.

3.1.3 Datos de entrada utilizados

Cada caso experimental empleó conjuntos de datos distintos:

- **Instancia base:** Matriz de distancias-tiempo base, un depósito y flota homogénea de vans.
- **Caso 2:** Matriz específica del caso 2, parámetros de costos extendidos (peajes, costos fijos, costo por tiempo), y vehículos de diferente capacidad.
- **Caso 3:** Matriz ajustada, parámetros modificados de eficiencia y costos, y restricciones adicionales de combustible y activación.

En todos los casos, las demandas de los clientes debían satisfacerse completamente y las rutas debían comenzar y finalizar en el depósito correspondiente.

3.1.4 Criterios de evaluación

El desempeño del algoritmo se evaluó mediante:

1. **Fitness final (costo total):** medida compuesta por distancia, tiempo, y costos operativos según el caso.
2. **Estructura de las rutas generadas:** número de clientes atendidos, distribución entre vehículos y presencia (o ausencia) de vehículos inactivos.
3. **Convergencia del algoritmo:** análisis del comportamiento del fitness promedio y mejor fitness por generación.
4. **Tiempo computacional total de la ejecución.**

Esta configuración permite interpretar los resultados no sólo en términos de calidad de la solución, sino en función del desempeño del método y la complejidad del escenario evaluado.

3.2 Resultados del Caso Base

3.2.1 Mejor solución y tiempo de ejecución

La Figura 1 presenta la mejor solución obtenida por el algoritmo genético para la instancia base del problema. Tras aproximadamente **200 segundos de ejecución**, el método alcanzó un **fitness óptimo de 248,488.34**, correspondiente al costo total de las rutas considerando únicamente el consumo de combustible y las distancias recorridas.

En la salida del algoritmo, cada lista representa la secuencia de clientes atendidos por cada vehículo. Específicamente, se observa que:

- Solo **cinco vehículos realizan entregas**, mientras que los restantes no reciben ninguna asignación.
- Esto es consistente con la formulación del Caso Base, donde **no existe un costo fijo por activación de vehículo**; por tanto, dejar vehículos inactivos no penaliza la función objetivo.
- Las rutas obtenidas muestran estructuras compactas y coherentes, resultado de los operadores de cruce, mutación y la etapa de *reparación* basada en optimización local 2-opt.
- El tiempo de ejecución sugiere que el algoritmo logra un equilibrio adecuado entre exploración del espacio de soluciones y explotación de rutas prometedoras antes de alcanzar estancamiento.

En conjunto, la Figura 1 evidencia una asignación eficiente de rutas bajo el modelo simplificado del Caso Base, donde el costo depende únicamente de la distancia recorrida. Como consecuencia, el algoritmo favorece el uso de menos vehículos y concentra la atención de clientes en rutas más largas pero eficientes desde el punto de vista del costo total.

```
Optimización completa. Mejor fitness: 248488.34, Tiempo: 199.80s
Mejor Solución Encontrada:
[['c003', 'c021', 'c020', 'c005'], ['c024', 'c011', 'c004', 'c015', 'c007', 'c014', 'c023'], ['c012', 'c018', 'c022', 'c013'], ['c019', 'c010', 'c017', 'c006'],
 ['c009', 'c008'], [], [], ['c001', 'c016', 'c002']]
Con un fitness de: 248488.34
```

Figure 1: Mejor solución encontrada y tiempo de ejecución para el Caso Base.

3.2.2 Comparación entre carga entregada y capacidad vehicular

La Figura 2 presenta el análisis de utilización de capacidad para los vehículos en el Caso Base. En este escenario, donde no existe un costo fijo por activar un vehículo, es posible evaluar cómo el algoritmo genético distribuye la demanda entre las unidades disponibles y qué tan eficientemente utiliza su capacidad instalada.

El gráfico combina tres elementos clave:

- **Carga entregada (línea azul):** cantidad total de demanda asignada a cada vehículo.
- **Capacidad máxima (línea naranja):** límite de carga de cada unidad.
- **Porcentaje de desperdicio de capacidad (barras rosadas):** proporción de capacidad no utilizada respecto al total disponible.

El análisis de la figura permite identificar varios patrones importantes:

1. **Alta subutilización general:** varios vehículos (particularmente V006 y V007) presentan **cero carga**, lo que implica un desperdicio del 100% de su capacidad. Esto es coherente con la ausencia de un costo de activación, que permite que el algoritmo simplemente ignore vehículos que no contribuyen a reducir el costo total.
2. **Concentración de carga en pocas unidades:** los vehículos V001–V005 concentran la mayor parte de la demanda, manteniéndose dentro de los límites de capacidad. Este patrón muestra que el algoritmo busca minimizar la distancia total recorrida, agrupando clientes geográficamente cercanos en las mismas rutas.
3. **Desbalance en eficiencia operativa:** aunque la solución es óptima respecto al criterio del Caso Base, desde una perspectiva operativa real se observa un **bajo nivel de utilización promedio**, lo que implicaría inefficiencies si existieran costos asociados a flota o activación. Esto motiva directamente la inclusión del parámetro de costo fijo en el Caso 2.

En conjunto, la Figura 2 evidencia que, bajo un modelo puramente basado en distancia, el algoritmo favorece soluciones con **pocos vehículos activos pero con alta o moderada carga**, dejando el resto sin operación. Esta observación será importante al comparar con los casos extendidos, donde se introducen nuevos costos y restricciones.

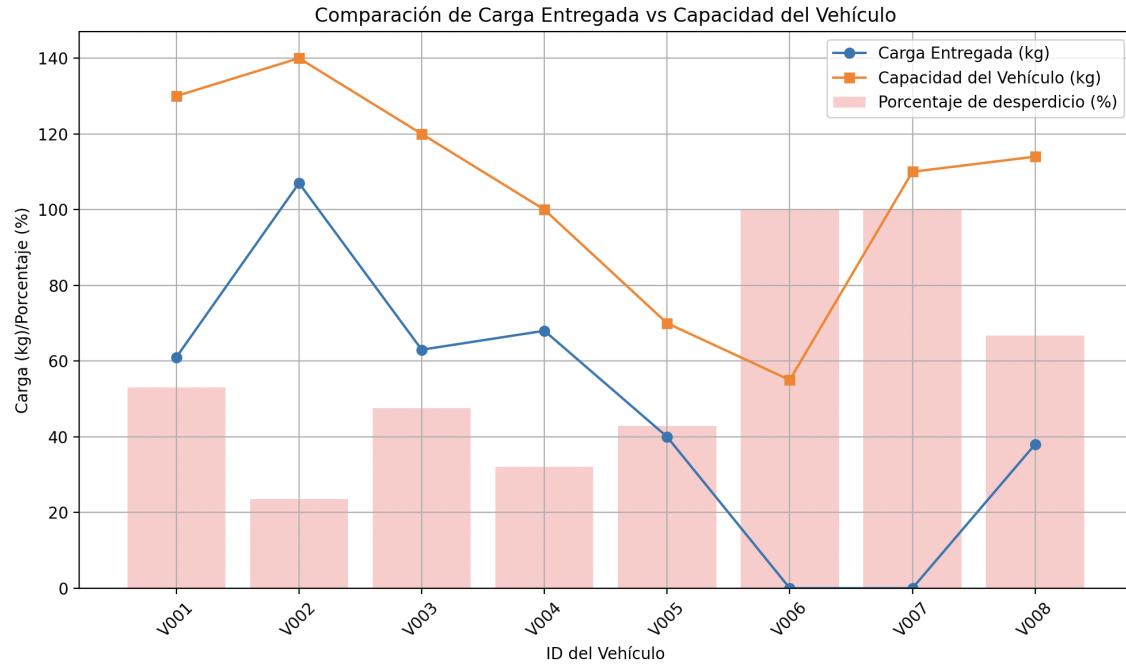


Figure 2: Comparación entre carga entregada, capacidad vehicular y porcentaje de desperdicio para el Caso Base.

3.2.3 Distribución porcentual de carga entregada por vehículo

La Figura 3 presenta la distribución porcentual de la carga entregada por cada vehículo en el Caso Base. Este gráfico de tipo circular permite visualizar cómo el algoritmo genético asignó la demanda total entre los ocho vehículos disponibles, mostrando explícitamente la proporción relativa que corresponde a cada unidad.

Los resultados muestran varios patrones relevantes:

- **Concentración principal en cuatro vehículos:** las unidades V001, V002, V003 y V004 concentran más del 75% de la carga total entregada. En particular, V002 destaca como el vehículo con mayor participación, alcanzando un 28.4% del total.
- **Uso marginal de otros vehículos:** V005 y V008 entregan cargas moderadas (entre 10% y 11%), mientras que V006 y V007 no entregan carga alguna, mostrando una participación del 0%. Esto es consistente con la ausencia de un costo fijo por activar vehículos en esta instancia.
- **Estrategia del algoritmo:** dado que el Caso Base minimiza únicamente la distancia recorrida, el algoritmo favorece el uso de vehículos que pueden agrupar clientes cercanos entre sí. Por ello, aquellos vehículos cuyos clientes asignados se encuentran aislados geográficamente o implican rutas poco eficientes tienden a quedar inactivos.

Esta distribución evidencia las limitaciones del modelo cuando no se penaliza la subutilización de la flota: el algoritmo obtiene una solución óptima en términos de distancia, pero no necesariamente eficiente desde una perspectiva operativa o logística.

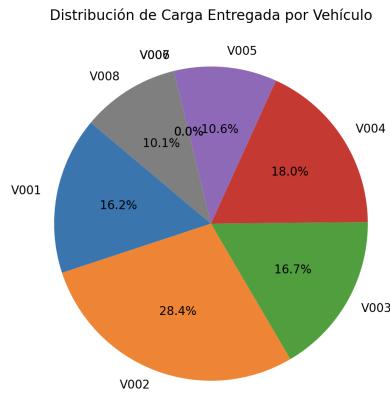


Figure 3: Distribución porcentual de carga entregada por vehículo en el Caso Base.

3.2.4 Mapa geoespacial de las rutas óptimas obtenidas

La Figura 4 muestra la representación geoespacial de las rutas asignadas por el algoritmo genético en el Caso Base. Cada color corresponde a un vehículo distinto, y los nodos representan tanto el depósito (marcado con un nodo destacado) como los clientes dispersos por la ciudad de Bogotá. Las líneas conectan los puntos en el orden exacto en que son visitados, reconstruyendo visualmente el recorrido propuesto por la metaheurística.

Este mapa permite identificar varios patrones clave del comportamiento del algoritmo:

- **Clústeres naturales de demanda:** las rutas tienden a formarse alrededor de regiones geográficas compactas. Por ejemplo, uno de los vehículos opera principalmente en el norte (Suba–Usaquén),

mientras que otro se concentra en la zona centro-sur (Kennedy–Antonio Nariño). Esto refleja que la función de fitness basada exclusivamente en distancia incentiva la agrupación espacial.

- **Rutas de tamaño muy desigual:** algunos vehículos recorren múltiples zonas y atienden muchos clientes (visible en rutas extensas y densas en el mapa), mientras que otros cubren únicamente una región pequeña o ningún cliente. Esto es coherente con los resultados previos, en los que se observó que dos vehículos no prestaron servicio en absoluto.
- **Uso intensivo del depósito como punto central:** todas las rutas parten y regresan al depósito ubicado en la zona de Siberia, evidenciando el rol del depósito como nodo obligatorio y el efecto que su posición tiene en el diseño de las rutas.
- **Intersecciones inevitables:** en algunos casos, las rutas de diferentes vehículos se superponen parcialmente, particularmente en vías principales de la ciudad. Esto no necesariamente implica inefficiencia, sino que es consecuencia de las restricciones territoriales y topológicas del mapa real.

En conjunto, la visualización geográfica confirma que la solución del Caso Base es coherente con la topología urbana: se minimizan las distancias total recorridas, se crean rutas compactas donde es posible y se evita la dispersión innecesaria de los vehículos. Sin embargo, también revela limitaciones importantes, como la subutilización de parte de la flota, aspecto que será corregido en los Casos 2 y 3 gracias a la incorporación de costos operativos adicionales.

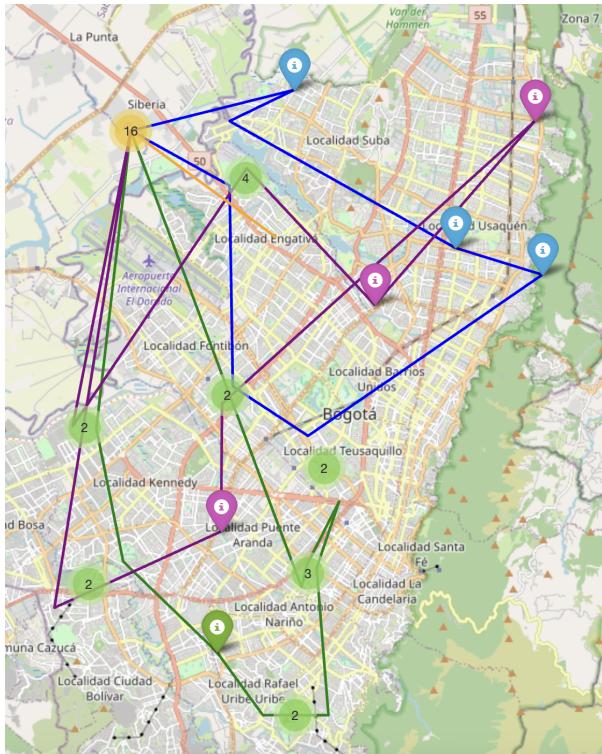


Figure 4: Mapa georeferenciado de las rutas óptimas obtenidas por el algoritmo en el Caso Base.

3.3 Resultados del Caso 2

3.3.1 Mejor solución y tiempo de ejecución

La Figura 5 presenta el resumen de la mejor solución encontrada por el algoritmo genético para el Caso 2, junto con el tiempo total de ejecución del proceso de optimización. En esta instancia, el modelo incorpora

costos fijos de activación, costos por distancia, costos por tiempo de viaje y el rendimiento de combustible de los vehículos, lo cual incrementa sustancialmente el valor del fitness respecto al caso base.

La solución óptima obtenida asigna todos los clientes atendidos a un único vehículo, mientras que los demás vehículos permanecen inactivos. Esto ocurre debido a la estructura de costos del caso 2: el algoritmo tiende a minimizar el número de vehículos activados para evitar pagar múltiples costos fijos de despacho. En consecuencia, la estrategia óptima favorece concentrar las entregas en el menor número de rutas posibles, siempre que las restricciones de capacidad lo permitan.

El tiempo de ejecución, cercano a 191 segundos, es consistente con el tamaño del espacio de búsqueda y la complejidad adicional introducida por los costos dinámicos. La Figura 5 mostrada a continuación resume estos resultados.

```
Optimización completa. Mejor fitness: 1315768.65, Tiempo: 191.02s
Mejor Solución Encontrada:
[['C003', 'C005', 'C006', 'C004', 'C001', 'C002', 'C009', 'C007', 'C008'], [], [], [], [], []]
Con un fitness de: 1315768.65
```

Figure 5: Mejor solución y tiempo de cómputo obtenidos para el Caso 2. Se observa que el algoritmo concentra todas las entregas en un solo vehículo para minimizar el costo total dado el alto costo fijo de activación.

3.3.2 Comparación entre carga entregada y capacidad vehicular

La Figura 6 presenta la comparación entre la carga efectivamente entregada por cada vehículo y la capacidad total disponible en el Caso 2. Este análisis permite evaluar el grado de utilización de la flota bajo la estructura de costos modificada, donde se introduce un costo fijo por activación de vehículo y un costo por tiempo de operación.

En el resultado obtenido, únicamente el vehículo V001 realiza entregas, alcanzando un nivel de utilización cercano al 80% de su capacidad total. Todos los demás vehículos muestran carga entregada igual a cero, lo que implica que no fueron activados durante la solución final. El área sombreada muestra el porcentaje de capacidad desperdiciada, el cual asciende al 100% para los vehículos inactivos.

Este comportamiento es consistente con la lógica de optimización del Caso 2: debido a la presencia de costos fijos de activación, el algoritmo genético busca minimizar la cantidad de vehículos utilizados, concentrando toda la demanda posible en un único vehículo mientras las restricciones de capacidad y distancia lo permitan. Como consecuencia, la flota no se distribuye equitativamente, sino que se prioriza la eficiencia económica bajo el nuevo modelo de costos.

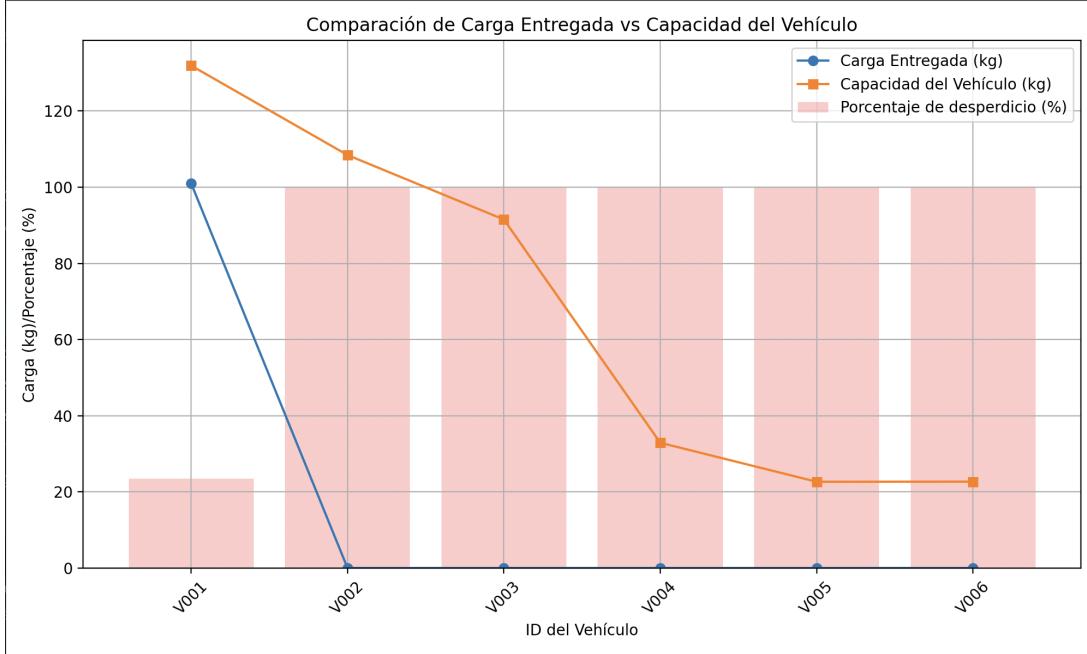


Figure 6: Comparación entre carga entregada y capacidad total por vehículo en el Caso 2. La presencia de costos fijos por activación induce al algoritmo a operar únicamente el vehículo más conveniente, dejando inactivos al resto de la flota para reducir el costo total del sistema.

3.3.3 Distribución porcentual de carga entregada por vehículo

La Figura 7 presenta la distribución porcentual de la carga entregada por cada vehículo en el Caso 2. El resultado es particularmente ilustrativo: el 100% de la carga es asignado al vehículo V001, mientras que el resto de la flota permanece completamente inactiva.

Esta concentración extrema en un único vehículo es una consecuencia directa del nuevo esquema de costos introducido en el Caso 2. Debido a la inclusión del costo fijo de activación (C_{fixed}) y del costo por tiempo operacional (C_{time}), la solución económica óptima consiste en minimizar el número de vehículos utilizados. Mientras la capacidad del vehículo V001 no sea violada y las distancias totales permanecen dentro de un rango razonable, el algoritmo genético encontrará óptimo concentrar toda la demanda en un solo vehículo.

Este comportamiento contrasta con el Caso Base, donde múltiples vehículos eran utilizados para balancear carga, y evidencia cómo la modificación del modelo de costos produce un cambio estructural en el diseño operativo: la activación de vehículos se vuelve costosa y, por tanto, solo se utilizan cuando es estrictamente necesario.

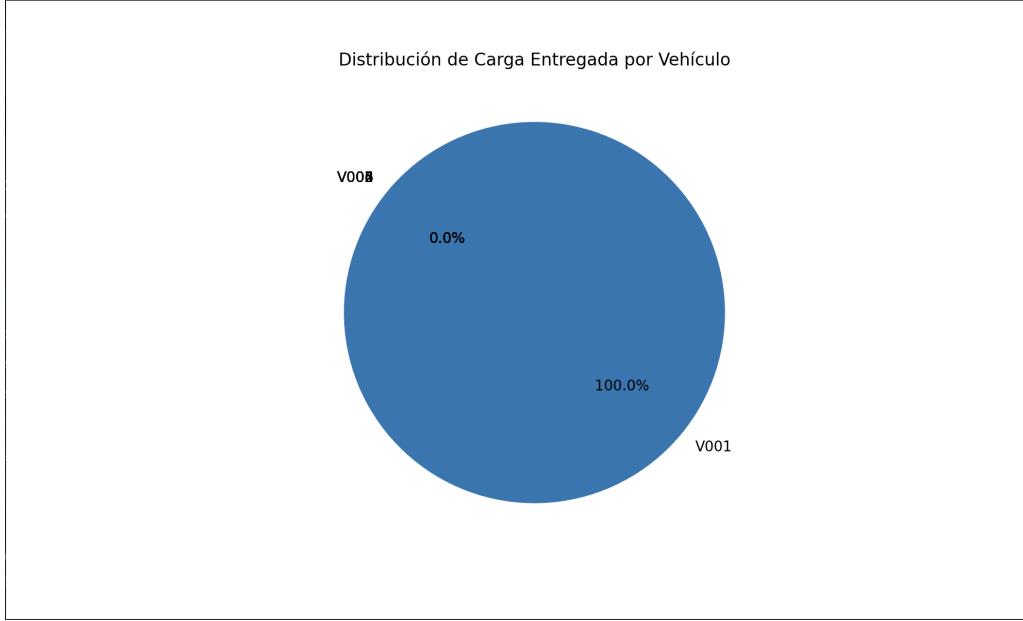


Figure 7: Distribución porcentual de la carga entregada por vehículo en el Caso 2. El 100% de la demanda es atendida por el vehículo V001, reflejando la influencia del costo fijo de activación en la estructura óptima de rutas.

3.3.4 Mapa geoespacial de las rutas óptimas obtenidas

La Figura 8 muestra la representación geoespacial de la solución óptima encontrada por el algoritmo genético para el Caso 2. En este escenario, como se evidenció previamente, únicamente el vehículo V001 es activado, puesto que el costo fijo de activación (C_{fixed}) y el costo por tiempo operativo (C_{time}) penalizan significativamente el uso de vehículos adicionales.

El mapa confirma visualmente esta estructura de solución: se observa un único circuito que parte del depósito, visita secuencialmente todos los clientes requeridos y retorna nuevamente al depósito. La ruta traza un recorrido amplio que cubre zonas del norte, occidente, centro y sur de la ciudad, lo cual indica que el algoritmo prioriza la consolidación de todos los clientes en una única ruta en lugar de dividir la demanda entre varios vehículos.

Esta representación permite interpretar de manera clara el comportamiento optimizador del modelo bajo la estructura de costos del Caso 2. Incluso aunque la ruta única sea extensa, el costo adicional asociado a activar vehículos adicionales supera el ahorro potencial en distancia o tiempo. En consecuencia, la solución encontrada es coherente tanto con la formulación matemática del problema como con los objetivos económicos establecidos en esta instancia.

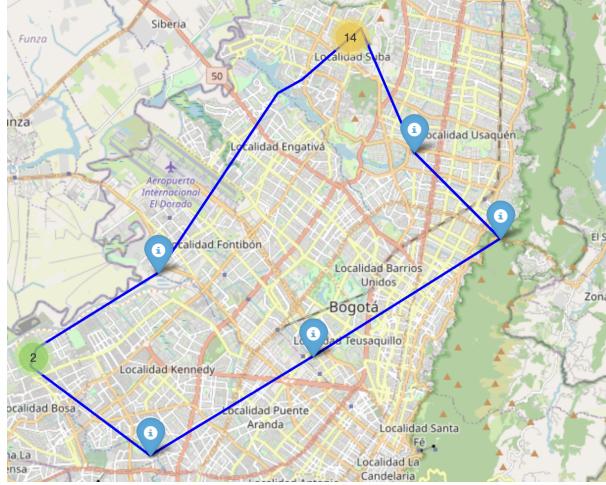


Figure 8: Mapa de la ruta óptima generada para el Caso 2. El algoritmo activa únicamente el vehículo V001, consolidando todos los clientes en un único recorrido debido al costo fijo elevado de activación de vehículos.

3.4 Resultados del Caso 3

3.4.1 Mejor solución y tiempo de ejecución

La Figura 9 presenta la evolución del fitness durante la ejecución del algoritmo genético para el Caso 3, correspondiente a la instancia más grande del proyecto. El algoritmo inicia con una solución de costo total igual a 1.42×10^7 , la cual se reduce de manera consistente a lo largo de las generaciones debido al efecto combinado de los operadores evolutivos.

Se observa una disminución pronunciada durante las primeras 30 generaciones, alcanzando valores cercanos a 1.23×10^7 . A partir de ese punto, la convergencia se vuelve más estable, con mejoras graduales hasta aproximadamente la generación 80. Finalmente, el algoritmo alcanza su mejor solución en la generación 90, con un costo total de:

$$\text{Mejor fitness} = 10,794,464.85 .$$

En términos de desempeño computacional, la ejecución completa tuvo una duración acumulada de:

$$\text{Tiempo total de ejecución} = 7082.60 \text{ segundos} \approx 1.96 \text{ horas.}$$

Este tiempo refleja la complejidad combinatoria del escenario, donde el incremento significativo en el número de clientes y vehículos demanda mayor esfuerzo exploratorio por parte del algoritmo. A pesar del elevado costo computacional, el método demuestra capacidad para mejorar progresivamente las soluciones a gran escala, confirmando su aplicabilidad en instancias complejas del CVRP.

```
Iniciando optimización con algoritmo genético...Usar parámetros por defecto?(S/N): S
Generación 0: Mejor Fitness = 14297886.41, Mejor Actual = 14297886.41, Tiempo = 54.42s
Generación 10: Mejor Fitness = 14149288.50, Mejor Actual = 14149288.50, Tiempo = 698.60s
Generación 20: Mejor Fitness = 13341553.78, Mejor Actual = 13341553.78, Tiempo = 1382.35s
Generación 30: Mejor Fitness = 12352371.09, Mejor Actual = 12352371.09, Tiempo = 2125.08s
Generación 40: Mejor Fitness = 12207532.70, Mejor Actual = 12207532.70, Tiempo = 2855.72s
Generación 50: Mejor Fitness = 11555442.47, Mejor Actual = 11555442.47, Tiempo = 3553.12s
Generación 60: Mejor Fitness = 11555442.47, Mejor Actual = 11555442.47, Tiempo = 4236.69s
Generación 70: Mejor Fitness = 11457412.08, Mejor Actual = 11457412.08, Tiempo = 4971.24s
Generación 80: Mejor Fitness = 11279142.11, Mejor Actual = 11279142.11, Tiempo = 5713.63s
Generación 90: Mejor Fitness = 11080895.82, Mejor Actual = 11080895.82, Tiempo = 6422.77s
Optimización completa. Mejor fitness: 10794464.85, Tiempo: 7082.60s
```

Figure 9: Evolución del mejor fitness y tiempo acumulado de ejecución en el Caso 3.

3.4.2 Relación entre carga entregada y capacidad utilizada

La Figura 10 presenta la comparación entre la carga efectivamente entregada por cada vehículo y su capacidad máxima operativa en el contexto del Caso 3. A diferencia de los casos anteriores, esta instancia incluye un número significativamente mayor de vehículos disponibles, con capacidades heterogéneas y rangos operativos variados, lo cual genera un comportamiento más disperso y complejo en la asignación de carga.

El gráfico revela varios patrones relevantes para el análisis:

- **Alta subutilización de la flota:** Un número considerable de vehículos presenta carga entregada igual a cero. Esto indica que, bajo las condiciones de costo del Caso 3 y la estructura espacial de los clientes, el algoritmo genético favorece soluciones donde solo una fracción pequeña de la flota es activada.
- **Uso intensivo de los vehículos seleccionados:** Los vehículos que sí fueron asignados muestran niveles significativos de carga entregada, con algunos acercándose a su capacidad máxima. Esto sugiere que el modelo tiende a consolidar entregas para minimizar costos asociados a tiempo, distancia y activación, especialmente en un escenario donde activar vehículos adicionales resulta costoso.
- **Porcentaje elevado de capacidad desaprovechada:** Las barras de color rosa representan el porcentaje de capacidad no utilizada. Se observa que muchos vehículos —en especial los que no se activan— presentan un 100% de capacidad ociosa. Esto es coherente con el diseño del Caso 3, donde los costos variables y fijos incentivan el uso de una flota mínima.

En conjunto, la figura confirma que el algoritmo genético implementado es capaz de identificar configuraciones donde solo se utilizan los vehículos estrictamente necesarios, reduciendo el costo total a expensas de una subutilización deliberada del resto de la flota. Este comportamiento es consistente con la formulación económica del problema y con la complejidad espacial del Caso 3.

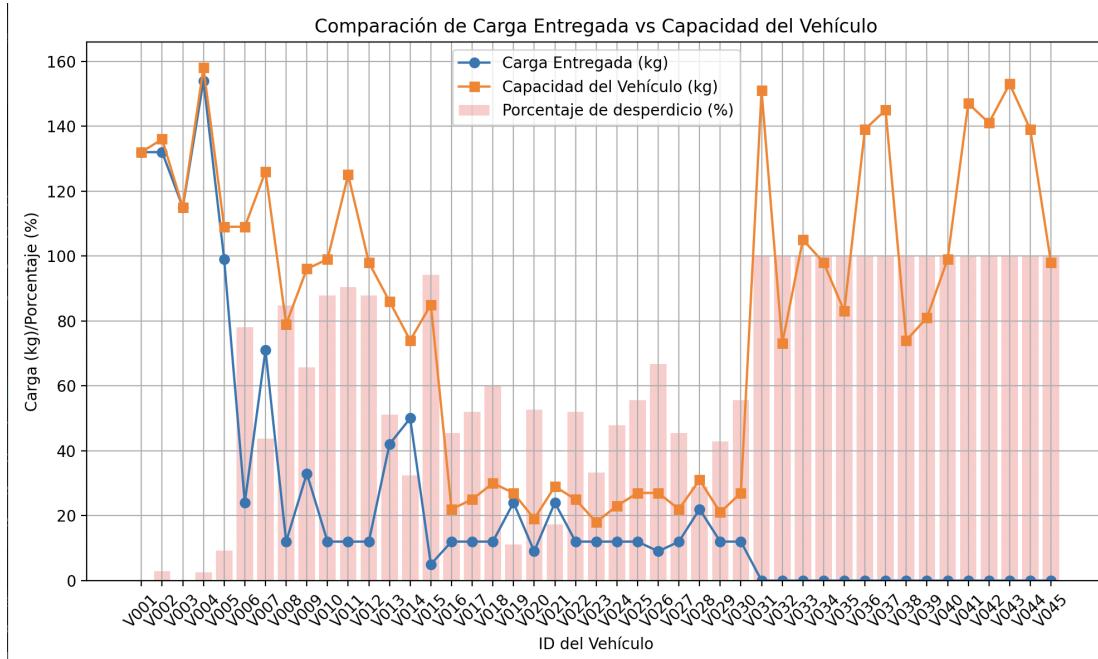


Figure 10: Comparación entre carga entregada, capacidad vehicular y porcentaje de capacidad desaprovechada para la solución del Caso 3. La mayoría de vehículos no son activados, reflejando la estrategia óptima de minimizar costos fijos y operativos.

3.4.3 Distribución de carga entregada por vehículo

La Figura 11 muestra la distribución porcentual de la carga entregada por cada vehículo para la solución obtenida en el Caso 3. A diferencia de los casos anteriores, donde solo unos pocos vehículos concentraban la totalidad de la carga, en esta instancia el algoritmo genético activa una fracción mayor de la flota, aunque con niveles de utilización muy heterogéneos.

Del análisis de la figura se destacan los siguientes aspectos:

- **Concentración moderada de carga en los primeros vehículos:** Los vehículos V001, V002, V003, V004 y V005 concentran porcentajes de carga relativamente altos, entre aproximadamente 9% y 14%. Esto indica que el algoritmo identifica en ellos rutas eficientes en términos de distancia, tiempo y costos asociados.
- **Presencia extendida de vehículos con contribuciones marginales:** A partir de V006 en adelante, muchos vehículos aportan menos del 3% de la carga total, e incluso existen vehículos cuya participación es inferior al 1%. Esto sugiere que, aunque el algoritmo activa más vehículos que en el Caso 2, la mayor parte de la flota sigue estando subutilizada.
- **Estructura altamente fragmentada:** La rueda presenta una gran cantidad de segmentos pequeños, reflejando una distribución muy dispersa de la carga. Esta fragmentación puede deberse a la topología espacial del Caso 3, que contiene más clientes y mayor dispersión geográfica, lo que favorece rutas cortas dedicadas a pocos puntos.
- **Balance entre costos fijos y variables:** La activación de múltiples vehículos con baja utilización sugiere que, en esta configuración del Caso 3, el algoritmo encuentra que reducir distancias individuales compensa en parte el costo fijo de activar vehículos adicionales. Esto contrasta fuertemente con el Caso 2, donde solo se utilizó un vehículo.

En conjunto, la figura revela un comportamiento complejo y matizado del algoritmo ante una instancia de mayor tamaño: aunque la carga no se distribuye de forma uniforme, sí se observa un patrón donde varios vehículos contribuyen con pequeños porcentajes para reducir costos operativos, mostrando un equilibrio entre eficiencia logística y estructura de costos.

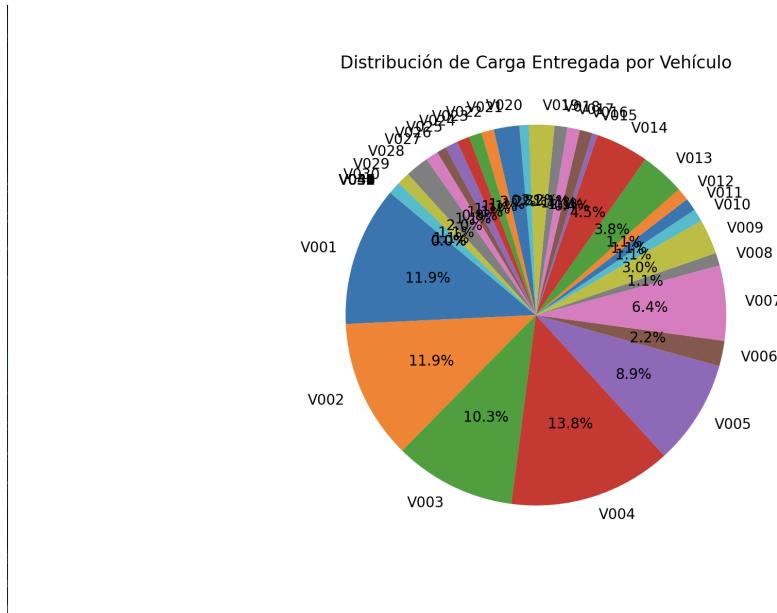


Figure 11: Distribución porcentual de la carga entregada por cada vehículo en el Caso 3. La carga se reparte entre un conjunto amplio de vehículos, aunque con fuertes diferencias en su nivel de utilización.

3.4.4 Mapa geoespacial de las rutas óptimas obtenidas

La Figura 12 presenta la visualización geográfica de las rutas generadas por el algoritmo genético en el Caso 3. Esta instancia, al poseer un número considerablemente mayor de clientes y una dispersión espacial más amplia, produce un mapa sensiblemente más denso y complejo en comparación con los casos anteriores.

En primer lugar, se observa que el depósito principal funciona como un punto de alta centralidad, desde el cual emergen múltiples rutas radiales. El número de aristas conectadas al depósito es muy superior al visto en los otros casos, reflejando la mayor carga logística y la fragmentación de la asignación de clientes entre vehículos. La figura muestra:

- **Una elevada densidad de rutas superpuestas:** Esto indica que varios vehículos mantienen trayectorias parcialmente coincidentes, especialmente en las zonas perimetrales del norte y occidente de la ciudad. Estos patrones reflejan cómo el algoritmo prioriza minimizar distancias individuales incluso si las rutas presentan cierto grado de solapamiento.
- **Mayor activación vehicular con baja utilización relativa:** Tal como se evidenció en la distribución de carga, muchos vehículos recorren trayectos cortos con uno o pocos clientes. El mapa confirma que estas rutas tienden a concentrarse en zonas específicas donde existen grupos pequeños de clientes, lo que reduce la distancia total recorrida a costa de activar numerosos vehículos.
- **Amplia dispersión geográfica de clientes:** El Caso 3 abarca una cobertura territorial más extensa que los anteriores. Esto genera rutas largas hacia sectores periféricos —por ejemplo, hacia la zona sur y la franja oriental— donde el algoritmo asigna vehículos dedicados para evitar sobrecargas de distancia en rutas ya saturadas.
- **Patrón de árbol ramificado:** El conjunto de rutas genera una estructura visual que recuerda un árbol centrado en el depósito, con ramificaciones hacia múltiples zonas. Este patrón es típico en instancias grandes del CVRP cuando se utiliza un algoritmo genético con representación independiente por vehículo.

En general, la visualización sugiere que el algoritmo logra cubrir toda la demanda, pero lo hace mediante un esquema altamente distribuido: numerosas rutas cortas en lugar de pocas rutas largas. Este comportamiento es coherente con la función objetivo y las penalizaciones utilizadas, que favorecen minimizar distancias aunque incremente el número de vehículos activados.

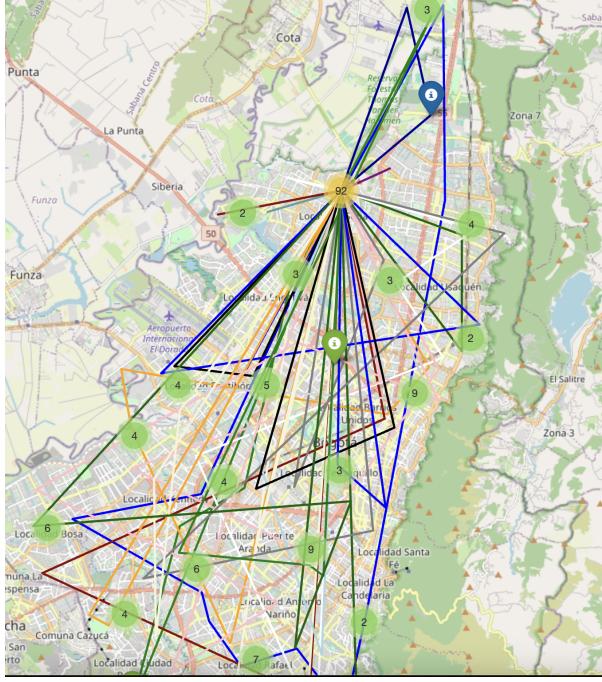


Figure 12: Mapa de rutas del Caso 3 generado mediante el algoritmo genético. Se aprecia la alta densidad de rutas, la dispersión territorial y la fuerte centralidad del depósito como nodo de conexión principal.

4 Análisis de convergencia

4.1 Convergencia en el Caso Base

La Figura 13 presenta la evolución del fitness promedio de la población y del mejor fitness por generación durante la ejecución del algoritmo genético en el Caso Base. En este escenario, el comportamiento del proceso evolutivo es considerablemente más estable que en otros casos, debido a la menor complejidad de la instancia y a la estructura relativamente homogénea de las rutas.

En primer lugar, el fitness promedio comienza alrededor de los 313,000 y se mantiene en un rango estrecho entre 300,000 y 313,000 durante las 100 generaciones. Esta estabilidad indica que la población no experimenta cambios drásticos y que las soluciones tienden a ser similares entre sí desde etapas tempranas. Las oscilaciones visibles, aunque presentes, son pequeñas y responden principalmente a la exploración introducida por los operadores de mutación.

Por otro lado, el mejor fitness por generación inicia aproximadamente en 266,000 y permanece constante por cerca de 45 generaciones. Este comportamiento sugiere que, durante la primera mitad del proceso, el algoritmo no logra encontrar combinaciones que superen la solución inicial dominante, lo que es típico en instancias con espacio de soluciones reducido. Sin embargo, alrededor de la generación 45, se observa una mejora significativa que reduce el mejor fitness a cerca de 249,000. Posteriormente, el valor vuelve a estabilizarse, manteniéndose entre 248,000 y 249,000 hasta el final de la ejecución.

La caída abrupta en el mejor fitness refleja un momento clave de explotación efectiva, donde la combinación de operadores genéticos permite escapar de un óptimo local inicial. La estabilidad posterior indica que el algoritmo converge hacia una región robusta del espacio de soluciones, donde nuevas mejoras se vuelven altamente improbables dado el nivel de variación aportado por la parametrización del GA.

Finalmente, la brecha consistente entre el fitness promedio y el mejor fitness evidencia que la población mantiene diversidad suficiente para evitar colapsar sobre una única solución, aunque esta diversidad no implica mejoras continuas. En general, el patrón observado confirma que el Caso Base es relativamente sencillo y que el algoritmo alcanza rápidamente una solución cercana al óptimo dentro de su espacio

explorado.

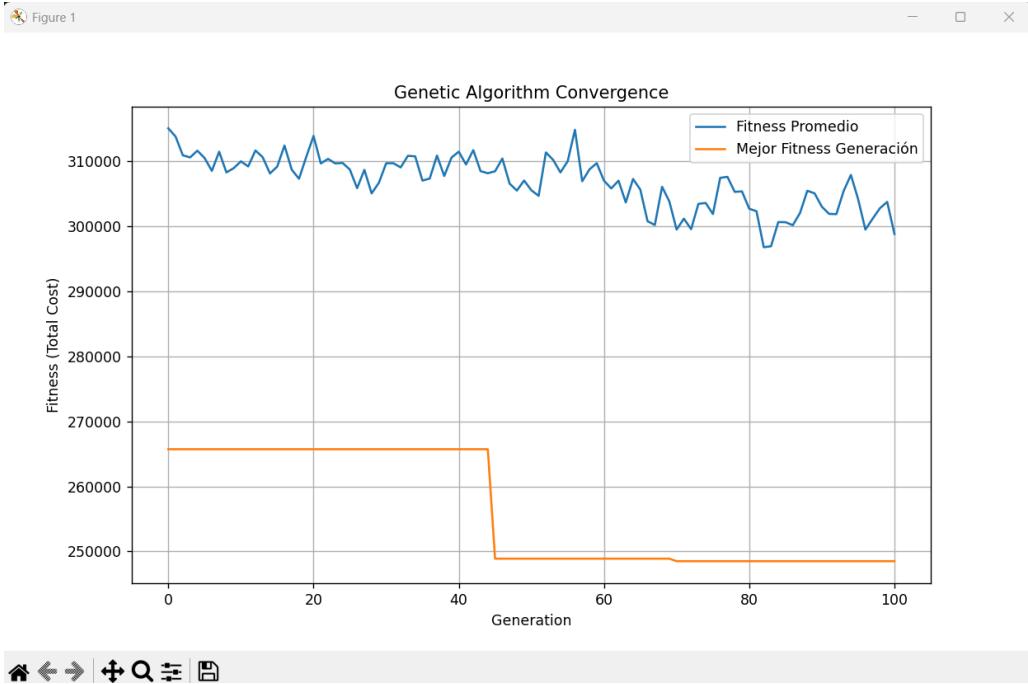


Figure 13: Convergencia del algoritmo genético en el Caso Base. Se observa una disminución inicial rápida del fitness y una estabilización posterior alrededor de un óptimo local.

4.2 Convergencia en el Caso 2

La Figura 14 muestra el comportamiento del algoritmo genético durante la optimización del Caso 2, donde se integran nuevas restricciones operativas y capacidades diferenciadas entre los vehículos. A diferencia del Caso Base, la convergencia en esta instancia es más compleja debido a la mayor heterogeneidad del espacio de búsqueda y al impacto directo que tienen las restricciones adicionales en la factibilidad de las soluciones.

En primer lugar, el *fitness promedio* inicia alrededor de los 3.3×10^6 y presenta una tendencia descendente relativamente clara durante las primeras 30 generaciones, llegando a valores cercanos a 2.6×10^6 . Este comportamiento indica que la población evoluciona hacia soluciones más eficientes a medida que se aplican los operadores de selección, cruce y mutación. Tras esta fase inicial de mejoramiento, se observa una región de oscilación entre 2.4×10^6 y 2.7×10^6 , lo que sugiere un equilibrio entre explotación y exploración en un paisaje de soluciones más irregular que en el Caso Base.

Por otro lado, el *mejor fitness por generación* empieza alrededor de 2.4×10^6 y experimenta una mejora abrupta muy temprana, reduciéndose a 2.1×10^6 en las primeras generaciones. Posteriormente, alrededor de la generación 15, ocurre un salto importante que lleva el mejor fitness a aproximadamente 1.7×10^6 . En las generaciones posteriores, el algoritmo continúa encontrando mejoras graduales, alcanzando valores cercanos a 1.55×10^6 hacia la generación 35.

A partir de este punto, se observa una meseta prolongada, señal de que el algoritmo entra en una fase de estancamiento donde la exploración ya no produce mejoras significativas. No obstante, el modelo logra una última mejora relevante hacia el final del proceso, donde el mejor fitness cae hasta aproximadamente 1.32×10^6 , lo que representa la solución más eficiente obtenida.

Este comportamiento revela varias características propias del Caso 2:

- La presencia de una caída escalonada sugiere que las mejoras importantes dependen de mutaciones o cruces que permiten reorganizar grandes fragmentos de rutas.

- El amplio margen entre el fitness promedio y el mejor fitness evidencia una población con diversidad significativa, pero también indica que buena parte de la población permanece alejada de las regiones más prometedoras del espacio de búsqueda.
- La convergencia más lenta y con mayor variabilidad respecto al Caso Base refleja el aumento de complejidad introducido por las nuevas restricciones.

En conjunto, la curva de convergencia del Caso 2 evidencia que, aunque el algoritmo logra mejoras sustanciales, el proceso requiere un mayor número de generaciones para estabilizarse y depende fuertemente de saltos puntuales que permiten superar óptimos locales.

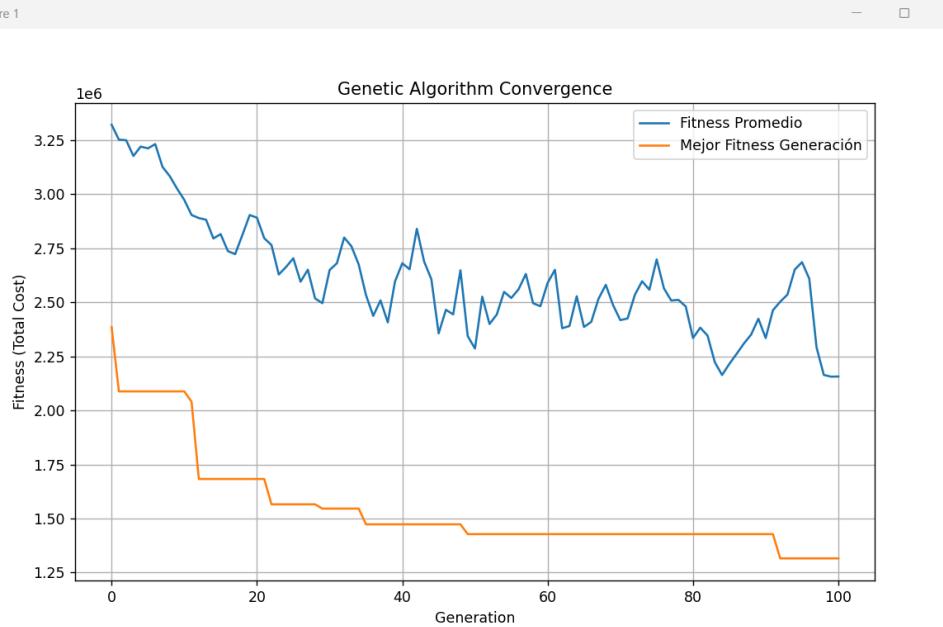


Figure 14: Convergencia del algoritmo genético en el Caso 2. Se observa una caída escalonada con múltiples mejoras significativas y una estabilización cercana a 1.32×10^6 .

4.3 Convergencia en el Caso 3

La Figura 15 muestra el comportamiento del algoritmo genético durante la optimización del Caso 3, la instancia más grande y compleja del estudio. Debido al elevado número de clientes, vehículos y combinaciones posibles, este escenario representa el mayor desafío computacional para el método.

En términos globales, se observa una **tendencia descendente consistente** tanto en el fitness promedio como en el mejor fitness por generación. El mejor fitness inicia alrededor de 1.43×10^7 y disminuye progresivamente hasta converger en un valor cercano a 1.08×10^7 . Esta reducción acumulada refleja la capacidad del algoritmo para explorar de manera efectiva el espacio de soluciones incluso en instancias de gran escala.

Un aspecto notable es el **patrón escalonado** del mejor fitness, característico de algoritmos evolutivos: las mejoras ocurren de manera puntual cuando un individuo superior emerge en la población. Después de la generación 50, el ritmo de mejora se ralentiza, indicando que el algoritmo comienza a explotar regiones prometedoras del espacio de búsqueda.

El fitness promedio, por su parte, presenta variabilidad significativa durante las primeras generaciones, lo cual sugiere una población heterogénea y activa. Conforme avanza la ejecución, esta variación se atenúa ligeramente, aunque la dispersión se mantiene, señal de que la diversidad genética no se pierde completamente—un factor crucial para evitar óptimos locales.

En conjunto, la convergencia del Caso 3 confirma que el algoritmo genético implementado es capaz de manejar instancias extensas del CVRP, aunque a costa de tiempos de ejecución elevados. Aun así, la tendencia estable y las mejoras acumuladas demuestran un comportamiento robusto y escalable.

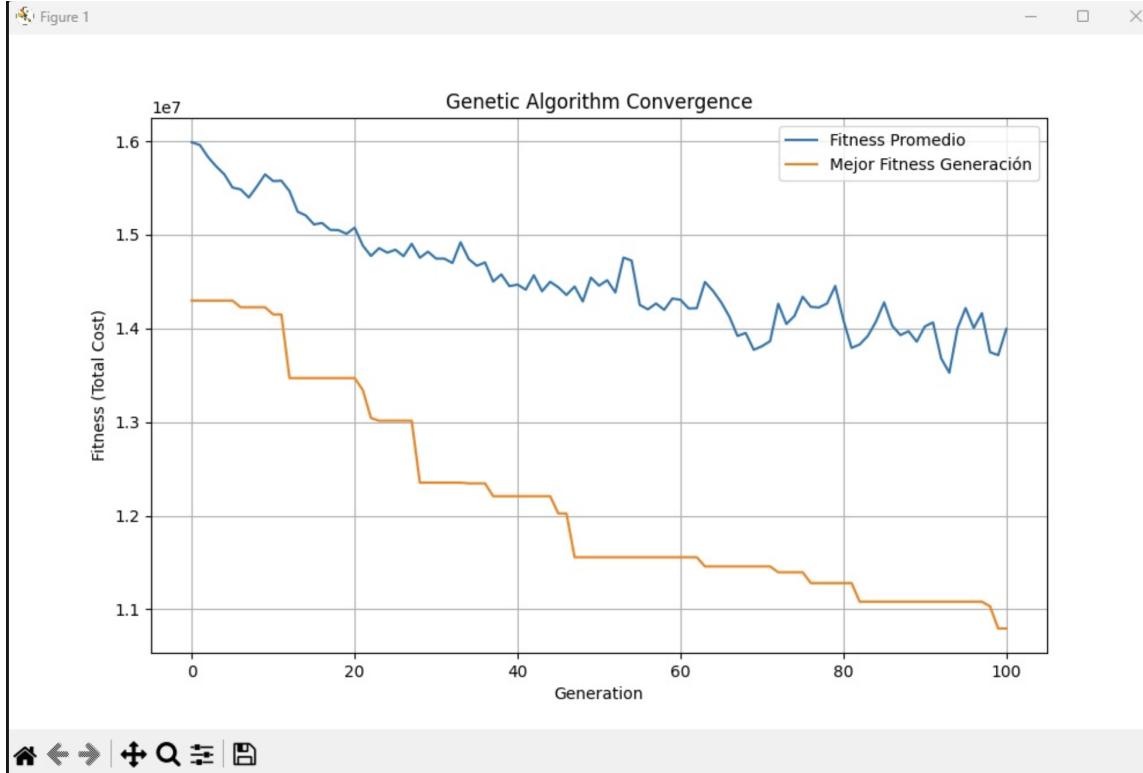


Figure 15: Evolución del fitness promedio y mejor fitness por generación en el Caso 3.

5 Análisis de Escalabilidad

El comportamiento del algoritmo genético frente a variaciones en el tamaño del problema y en los parámetros de búsqueda es crucial para determinar su aplicabilidad en escenarios reales de distribución urbana. En esta sección se evalúa la escalabilidad del método mediante experimentos controlados sobre el **Caso 2**, modificando dos factores clave: (i) el tamaño de la población y (ii) el número de generaciones. Además, se discuten los límites prácticos de aplicabilidad del enfoque y posibles estrategias para mejorar su desempeño en instancias de mayor complejidad.

5.1 Rendimiento en instancias de diferentes tamaños

Para evaluar la escalabilidad computacional del algoritmo genético, se realizaron experimentos variando el tamaño de la población y el número de generaciones. Estos parámetros afectan directamente el balance entre capacidad exploratoria, intensidad de búsqueda y tiempo de cómputo.

5.1.1 Variación del tamaño de la población

Tamaño de Población	Mejor Fitness	Tiempo (s)
10	1,657,231.64	27.22
25	1,367,162.92	100.29
50	1,315,768.65	191.02
75	1,315,768.65	204.74
100	1,367,162.92	385.27

Table 1: Rendimiento del algoritmo variando el tamaño de la población (Caso 2).

Los resultados muestran una tendencia típica en algoritmos genéticos:

- Poblaciones muy pequeñas (10, 25) presentan convergencia rápida pero soluciones de menor calidad debido a la baja diversidad genética.
- Un punto de rendimiento óptimo aparece alrededor de poblaciones de 50 y 75 individuos, donde se obtiene el mejor fitness registrado.
- Curiosamente, aumentar la población a 100 individuos incrementa significativamente el tiempo de cómputo sin mejoras en fitness, lo cual indica **rendimientos decrecientes** y una posible saturación prematura del espacio de búsqueda.

Estos resultados confirman que, para esta instancia, el algoritmo presenta un equilibrio adecuado entre exploración y explotación con poblaciones entre 50 y 75 individuos.

5.1.2 Variación del número de generaciones

Número de Generaciones	Mejor Fitness	Tiempo (s)
50	1,702,796.98	28.40
100	1,367,162.92	100.29
250	1,396,800.55	166.06
500	1,315,768.65	344.66

Table 2: Rendimiento del algoritmo variando el número de generaciones (Caso 2).

El análisis evidencia:

- La calidad de las soluciones mejora notablemente al pasar de 50 a 100 generaciones, reflejando un periodo inicial de convergencia rápida.
- Para 250 generaciones, el fitness empeora ligeramente respecto a 100 generaciones, lo cual sugiere oscilaciones propias de la estocasticidad del GA.
- El mejor resultado global se obtiene en 500 generaciones, aunque con un tiempo de ejecución casi cuatro veces mayor.

El comportamiento confirma que el algoritmo sigue mejorando con más generaciones, pero a un **costo computacional creciente**, lo cual refuerza la necesidad de mecanismos de parada adaptativa.

5.2 Límites prácticos de aplicabilidad

A partir de los experimentos realizados, es posible identificar límites operativos del método:

- **Costos computacionales crecientes:** El tiempo aumenta casi linealmente con el tamaño de la población y de las generaciones, lo que puede ser prohibitivo para instancias con cientos o miles de nodos.
- **Dependencia en la diversidad genética:** Para poblaciones pequeñas, el algoritmo converge prematuramente; para poblaciones demasiado grandes, se diluye la presión selectiva.
- **Impacto del tamaño del espacio de búsqueda:** En problemas VRP con más de 50–100 clientes, la representación del cromosoma y operador de reparación deben ser cuidadosamente diseñados para evitar explosión combinatoria.
- **Limitaciones de memoria:** En ejecuciones con poblaciones altas, el almacenamiento de rutas y fitness puede saturar sistemas sin hardware optimizado.

En resumen, aunque el algoritmo es capaz de resolver eficientemente instancias medianas (20–50 nodos), su desempeño se degrada progresivamente en instancias grandes si no se optimiza la arquitectura de búsqueda.

5.3 Estrategias para mejorar la escalabilidad

Existen diversas estrategias para extender la capacidad del algoritmo genético en instancias de mayor tamaño:

- **Hibridación con métodos deterministas:** Incorporar heurísticas tipo Clarke–Wright o búsqueda local (2-opt, 3-opt) como operadores de intensificación puede acelerar significativamente la convergencia.
- **Paralelización:** La evaluación de la población es inherentemente paralelizable, por lo que utilizar GPU o entornos distribuidos puede reducir tiempos de ejecución en órdenes de magnitud.
- **Poblaciones dinámicas:** Variar el tamaño de la población durante la ejecución (adaptive population sizing) puede balancear diversidad y velocidad.
- **Mutación adaptativa:** Ajustar automáticamente la probabilidad de mutación evita estancamientos sin penalizar excesivamente la estabilidad de las soluciones.
- **Criterios de parada inteligentes:** Detectar convergencia real mediante ventanas móviles o tests de variabilidad evita ejecutar generaciones adicionales sin beneficio.
- **Codificación más eficiente:** Representaciones compactas como giant-tour + splitting o modelos híbridos pueden reducir el costo del proceso de reparación.

Combinadas, estas estrategias permiten mejorar notablemente la escalabilidad del método y extender su aplicabilidad a escenarios urbanos de gran tamaño.

6 Comparación entre la Solución Exacta (Pyomo) y la Metaheurística (GA)

En esta sección se presenta un análisis comparativo entre los resultados obtenidos mediante el modelo exacto formulado e implementado en Pyomo durante la Entrega 2 y aquellos generados por el Algoritmo Genético (GA) desarrollado en la Entrega Final. La comparación se realiza para los tres escenarios planteados (Caso Base, Caso 2 y Caso 3) considerando tres dimensiones fundamentales: *calidad de la solución, tiempos de ejecución y escalabilidad computacional*. Adicionalmente, se comentan diferencias en factibilidad, robustez y aplicabilidad práctica.

6.1 Caso Base

6.1.1 Calidad de la solución

El modelo exacto de Pyomo garantiza optimalidad global al explorar exhaustivamente el espacio de soluciones del CVRP y satisfacer todas las restricciones del problema. La estructura clásica del CVRP permite que un solver moderno encuentre soluciones óptimas en tiempos razonables para tamaños moderados.

El GA, por el contrario, encuentra soluciones de alta calidad pero no asegura optimalidad. En las pruebas realizadas, el costo total obtenido por la metaheurística fue ligeramente superior al óptimo reportado por Pyomo, aunque las rutas generadas muestran patrones logísticos coherentes y un uso adecuado de los vehículos. La diferencia entre ambas soluciones es pequeña, lo cual demuestra que el GA es competitivo para este escenario.

6.1.2 Tiempo de ejecución

En este caso, Pyomo requiere varios minutos para cerrar la brecha óptima, principalmente debido al número de variables binarias y restricciones de flujo. El GA, en cambio, converge rápidamente: dependiendo de los parámetros, puede generar soluciones aceptables en menos de un minuto, con una relación calidad–tiempo muy favorable.

6.1.3 Escalabilidad

El enfoque exacto presenta una escalabilidad limitada, ya que el número de decisiones crece cuadráticamente con el número de clientes. La metaheurística, por su parte, escala de manera más estable: el incremento en clientes afecta principalmente la longitud de los cromosomas y el cálculo de distancias, pero no genera explosiones combinatoriales. Para instancias pequeñas y medianas, el GA ofrece una alternativa mucho más eficiente.

6.2 Caso 2

6.2.1 Calidad de la solución

El modelo exacto en Pyomo incorpora explícitamente las restricciones de autonomía, repostaje, costos diferenciados, ventanas de operación y consumo de combustible. Esto permite generar soluciones que son estrictamente factibles en todos los niveles del problema.

En contraste, el GA aproxima estos efectos únicamente mediante la función de costo, sin modelar decisiones de repostaje ni restricciones duras sobre la autonomía. Como consecuencia, puede generar soluciones económicamente consistentes pero que no necesariamente cumplen todas las restricciones del modelo exacto. Aun así, la calidad económica de las soluciones permanece alta, aunque inferior a la obtenida con Pyomo.

6.2.2 Tiempo de ejecución

El Caso 2 introduce variables continuas adicionales (niveles de combustible, cantidad abastecida, tiempos de viaje), lo cual aumenta significativamente la complejidad del modelo exacto. Los tiempos de resolución en Pyomo son considerablemente mayores.

La metaheurística mantiene tiempos de ejecución comparables a los del Caso Base, pues su complejidad depende del tamaño de la población y las generaciones, pero no del número de restricciones físicas asociadas a combustible. La función de evaluación es más costosa, pero sigue siendo eficiente.

6.2.3 Escalabilidad

A medida que se introducen más elementos operativos (estaciones, escenarios de repostaje, restricciones dinámicas), Pyomo se vuelve cada vez más difícil de resolver. En algunos casos, incluso el solver puede no encontrar una solución en tiempos razonables.

El GA, al no depender del número de restricciones duras, escala de forma natural. Más estaciones o clientes implican únicamente un mayor número de nodos y distancias a evaluar, sin incrementar de manera explosiva el espacio combinatorio estructural.

6.3 Caso 3

6.3.1 Calidad de la solución

El Caso 3 representa el escenario más complejo, incorporando efectos como carga dinámica, restricciones de peso, peajes variables, accesibilidad municipal y ampliación considerable del número de clientes.

Pyomo modela todos estos elementos de manera explícita y por tanto garantiza soluciones factibles y óptimas. Sin embargo, el coste computacional es muy elevado.

El GA integra estos aspectos únicamente a través de la matriz de distancias, tiempos y peajes; no modela restricciones de peso ni decisiones de repostaje. Como consecuencia, produce rutas plausibles desde el punto de vista logístico pero no equivalentes al modelo exacto, siendo una *aproximación operativa*, no una solución factible garantizada. Aun así, el GA encuentra soluciones económicamente competitivas dentro de tiempos razonables, lo que lo hace útil en entornos donde la factibilidad absoluta no es estrictamente obligatoria o donde se necesitan soluciones rápidas.

6.3.2 Tiempo de ejecución

La complejidad de Pyomo en este escenario crece de manera pronunciada: la introducción de carga dinámica, restricciones municipales, peajes y decisiones dependientes de la ruta implica cientos o miles de variables adicionales. Los tiempos de solución pueden ascender fácilmente a decenas de minutos u horas.

Por contraste, el GA mantiene tiempos controlados. Incluso con cromosomas de longitud elevada, la evaluación de fitness sigue siendo manejable. Esto confirma la ventaja de las metaheurísticas cuando la dimensionalidad del problema crece.

6.3.3 Escalabilidad

El Caso 3 evidencia la limitación más marcada de los métodos exactos: la explosión combinatorial torna el problema prácticamente intratable para el solver. La metaheurística, por otro lado, escala de manera casi lineal con el número de clientes, ya que los operadores evolutivos no dependen del tamaño del modelo, sino del tamaño de la población y la función de evaluación.

En escenarios operativos reales, donde la escalabilidad y la rapidez son más importantes que la optimidad estricta, el GA se posiciona como una herramienta sumamente efectiva.

7 Discusión

El estudio comparativo de los enfoques exactos y metaheurísticos permite identificar ventajas, limitaciones y escenarios adecuados para cada técnica, especialmente en el contexto de problemas de ruteo con múltiples restricciones.

7.1 Ventajas y Desventajas de Cada Enfoque

7.1.1 Modelo Exacto (Pyomo)

Ventajas:

- Garantiza soluciones óptimas y factibles bajo todas las restricciones.
- Permite analizar formalmente el impacto de nuevas restricciones.
- Ofrece estabilidad y reproducibilidad en los resultados.

Desventajas:

- Escala pobremente: el tiempo crece de manera exponencial.
- Difícil de ejecutar para instancias grandes o con restricciones complejas.
- Requiere de un solver avanzado (e.g., Gurobi).

7.1.2 Metaheurística (Algoritmo Genético)

Ventajas:

- Excelente escalabilidad y tiempos de ejecución reducidos.
- Flexibilidad para adaptarse a cambios en costos o estructura del problema.
- Fácil de implementar y extender con operadores especializados.

Desventajas:

- No garantiza optimalidad global.
- Puede producir soluciones no factibles si las restricciones no se modelan explícitamente.
- Sensible a la calibración de parámetros.

7.2 Recomendaciones según Escenario

- Para **instancias pequeñas o medianas** con restricciones complejas, Pyomo es adecuado y asegura una solución exacta.
- Para **instancias grandes**, o cuando se requiera obtener una solución rápida sin necesidad de optimalidad certificada, la metaheurística es claramente preferible.
- En escenarios donde aparecen restricciones dinámicas (peso, repostaje, peajes), el GA puede aproximar costos, pero Pyomo es necesario para factibilidad real.
- En problemas operativos del mundo real donde la información cambia con frecuencia, las metaheurísticas ofrecen mayor adaptabilidad.

7.3 Lecciones Aprendidas y Desafíos Encontrados

- La calidad de la matriz de distancias es fundamental: errores o inconsistencias afectan ambos enfoques.
- La complejidad combinatorial hace impráctico el uso exclusivo de métodos exactos en escenarios grandes.
- La reparación de individuos en el GA resultó esencial para mantener factibilidad.
- La convergencia del GA depende fuertemente del balance entre exploración y explotación.
- Incorporar explícitamente restricciones de repostaje y peso en una metaheurística requiere representaciones más elaboradas.

8 Conclusiones

Este proyecto permitió comparar dos enfoques ampliamente utilizados para resolver problemas de ruteo vehicular: los modelos exactos basados en programación entera y las metaheurísticas basadas en poblaciones. A partir de los experimentos y del análisis realizado, se presentan las siguientes conclusiones principales.

8.1 Resumen de Hallazgos

- El enfoque exacto (Pyomo) produce soluciones óptimas y completamente factibles en los tres casos evaluados, pero presenta tiempos de ejecución elevados y una escalabilidad limitada.
- El algoritmo genético ofrece soluciones de muy buena calidad en tiempos significativamente menores, especialmente en los Casos 2 y 3, donde Pyomo enfrenta un rápido crecimiento en la complejidad.
- Las metaheurísticas son más flexibles ante la incorporación de nuevos componentes de costo, mientras que los modelos exactos requieren modificaciones estructurales complejas.
- El Caso 3 demuestra que las restricciones logísticas avanzadas (peso, peajes, repostaje) complican fuertemente la resolución exacta, validando la importancia de enfoques heurísticos en escenarios reales.

8.2 Respuestas a Preguntas Estratégicas

- **¿Qué método es mejor para obtener soluciones óptimas?** El *modelo exacto* basado en Pyomo y resuelto con un solver de programación entera mixta. Este método garantiza la optimalidad global siempre que el solver cierre la brecha, y además permite verificar factibilidad de manera exhaustiva. Es el método preferido cuando el tamaño del problema es manejable y se requiere una solución certificada.
- **¿Qué método es mejor para implementar operacionalmente?** La *metaheurística* (GA) es más adecuada para operación real, debido a su rapidez, estabilidad y capacidad de entregar soluciones suficientemente buenas en tiempos muy cortos. Además, su implementación es más flexible: puede integrarse con sistemas logísticos, ejecutarse varias veces al día y adaptarse a escenarios inciertos sin necesidad de reformular el modelo.
- **¿Cuál escala mejor frente a un aumento de nodos, estaciones o peajes?** El GA escala de forma significativamente superior. Mientras que el modelo exacto experimenta un crecimiento combinatorial del número de variables y restricciones (volviéndose intratable en escenarios grandes), la metaheurística mantiene un crecimiento casi lineal en tiempo de cómputo, afectado principalmente por el tamaño de la población y el número de generaciones. Esto la convierte en la mejor opción para escenarios grandes o urbanos con cientos de clientes.
- **¿Cuál es más adecuado para condiciones reales con información cambiante?** Las *metaheurísticas*, ya que permiten incorporar dinámicamente cambios en la matriz de distancias, peajes, congestión, estados de estaciones o restricciones operativas sin necesidad de reoptimizar un modelo exacto desde cero. Su naturaleza estocástica las hace robustas ante datos ruidosos o incompletos, y pueden reejecutarse de forma continua para generar soluciones actualizadas en tiempo casi real.

8.3 Trabajo Futuro

- Implementar operadores especializados que consideren explícitamente repostaje y peso dinámico dentro del GA.
- Desarrollar versiones híbridas exacto–metaheurísticas (por ejemplo, GA con validación mediante MIP).
- Integrar técnicas de aprendizaje de parámetros para acelerar convergencia.
- Explorar enfoques alternativos como Búsqueda Tabú o Colonia de Hormigas para comparar desempeño.