

Proyecto Etapa 3

Andrés Caballero, Nicole Martínez, Esteban Orjuela

29 de mayo de 2025

1 Descripción del Problema

1.1 Formulación matemática del CVRP con vehículos opcionales

1. Conjunto de nodos y arcos

- **Grafo dirigido:** $G = (V, A)$.

- **Nodos:**

$$V = \{0\} \cup \{1, 2, \dots, n\},$$

donde el nodo 0 representa el *depósito* y $1, \dots, n$ los clientes.

- **Arcos:**

$$A = \{(i, j) \mid i, j \in V, i \neq j\}.$$

A cada arco (i, j) se asocia una distancia (o costo) d_{ij} .

2. Conjunto de vehículos

- **Vehículos disponibles:**

$$K = \{1, 2, \dots, m\}.$$

- **Capacidad homogénea:** todos los vehículos comparten la misma capacidad de carga Q (unidades de demanda).
- **Opción de activación:** cada vehículo puede *salir o no* del depósito; si no se activa, no genera costo alguno.

Con estos elementos se construye la formulación completa (variables de decisión, función objetivo y restricciones) presentada en la siguiente sección.

Variables de decisión

$$x_{ij}^k = \begin{cases} 1, & \text{si el vehículo } k \text{ recorre el arco } (i, j), \\ 0, & \text{en otro caso,} \end{cases} \quad (i, j) \in A, k \in K$$

$$s_k = \begin{cases} 1, & \text{si el vehículo } k \text{ es utilizado (sale del depósito),} \\ 0, & \text{en otro caso,} \end{cases} \quad k \in K$$

$$y_j = \begin{cases} 1, & \text{si la demanda del cliente } j \text{ queda totalmente satisfecha,} \\ 0, & \text{en otro caso,} \end{cases} \quad j \in V \setminus \{0\}$$

Función objetivo

$$\min Z = \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^k$$

Restricciones

1. Demanda atendida

$$\sum_{i \in V} \sum_{k \in K} x_{ij}^k = y_j, \quad \forall j \in V \setminus \{0\} \quad (1a)$$

$$y_j = 1, \quad \forall j \in V \setminus \{0\} \quad (1b)$$

2. Conservación de flujo en los clientes

$$\sum_{i \in V} x_{ij}^k = \sum_{i \in V} x_{ji}^k, \quad \forall j \in V \setminus \{0\}, \forall k \in K$$

3. Activación opcional de cada vehículo y retorno obligatorio

$$\sum_{j \in V \setminus \{0\}} x_{0j}^k = s_k, \quad \forall k \in K \quad (2a)$$

$$\sum_{i \in V \setminus \{0\}} x_{i0}^k = s_k, \quad \forall k \in K \quad (2b)$$

La ecuación (2a) activa la salida del depósito, mientras que (2b) obliga al mismo vehículo a regresar.

$$s_k = 0 \implies \sum_j x_{0j}^k = \sum_i x_{i0}^k = 0 \quad (\text{el vehículo no se usa}).$$

$$s_k = 1 \implies \sum_j x_{0j}^k = \sum_i x_{i0}^k = 1 \quad (\text{el vehículo sale y vuelve exactamente una vez}).$$

De este modo se cumple que sólo los vehículos necesarios (aquellos con $s_k = 1$) abandonan el depósito, y todo vehículo que sale está obligado a regresar al nodo inicio.

4. Capacidad vehicular

$$\sum_{(i,j) \in A} q_j x_{ij}^k \leq Q, \quad \forall k \in K,$$

donde q_j es la demanda del cliente j .

5. Eliminación de subrutas (formulación MTZ)

$$u_i - u_j + (n + 1) x_{ij}^k \leq n, \quad \forall i, j \in V \setminus \{0\}, i \neq j, \forall k \in K.$$

Con u_i variables auxiliares continuas que representan el orden de visita de cada cliente en la ruta.

6. Dominio de las variables

$$x_{ij}^k, s_k, y_j \in \{0, 1\}, \quad u_i \in [0, n], \quad \forall i, j, k.$$

Las ecuaciones (2a)–(2b) modelan que *salir del depósito es opcional* para cada vehículo, mientras que el bloque (1) introduce la variable auxiliar y_j , cuyo valor (forzado a 1) sirve para verificar explícitamente que la demanda de cada cliente se satisface exactamente una vez.

1.2 Características de la instancia base

La instancia base contiene:

- 15 clientes en el área.
- 1 depósito central (nodo 0).
- 3 vehículos homogéneos con:
 - Capacidad $Q = 120$ unidades.
 - Autonomía ilimitada (solo restricción de capacidad).
- Demandas entre 5-25 unidades por cliente.
- Distancias calculadas con fórmula haversine.

1.3 Restricciones y consideraciones

1.3.1 Restricciones implementadas

- Restricción de capacidad:
 - Ninguna ruta puede superar $Q = 120$ unidades.
 - Implementada mediante reparación de soluciones inviables.
- Visita única:

- Cada cliente visitado exactamente una vez.
- Garantizada por operadores de reparación.
- Retorno al depósito:
 - Todas las rutas deben comenzar y terminar en el depósito.
 - Implementada en la representación de soluciones

1.3.2 Consideraciones especiales

1. **Flota homogénea:** Todos los vehículos tienen misma capacidad.
2. **Demanda conocida:** No hay incertidumbre en las demandas.
3. **Ventanas de tiempo:** No consideradas en esta fase (fuera de alcance).
4. **Nodos de reabastecimiento:** No aplica para el caso base.
5. **Función objetivo:** Minimizar distancia total recorrida.

1.3.3 Supuestos claves

- Las distancias son simétricas ($d_{ij} = d_{ji}$).
- No hay restricciones de tiempo de servicio en clientes.
- La matriz de distancias cumple desigualdad triangular.
- La demanda de cada cliente no excede la capacidad de un vehículo.
- Suponemos que la flota al ser homogénea tienen todos los vehículos una velocidad constante de 80km/h.

2 Método Implementado

2.1 Descripción detallada del método metaheurístico implementado

El algoritmo genético implementado sigue un esquema generacional con elitismo. El flujo principal del algoritmo es:

1. **Inicialización:** Generar población aleatoria de soluciones factibles.
2. **Evaluación:** Calcular fitness (distancia total) de cada solución.
3. **Selección:** Torneo binario para seleccionar padres.
4. **Reproducción:** Aplicar operadores genéticos (cruzamiento y mutación).
5. **Reemplazo:** Generación de nueva población manteniendo élite.

6. **Terminación:** Criterio de máximo de generaciones (400 iteraciones).

La implementación destaca por:

- Mecanismos de reparación automática para garantizar factibilidad.
- Balance entre exploración y explotación mediante elitismo controlado.
- Operadores específicos para problemas de ruteo.

2.2 Estrategias de representación y operadores

2.2.1 Representación de soluciones

Cada solución es una lista de rutas (una por vehículo). Características clave:

- **Codificación directa:** Listas de enteros representando secuencias de visita.
- **Depósito implícito:** Todas las rutas comienzan/terminan en el depósito (nodo 0).
- **Reparación automática:** Garantiza que todas las soluciones sean factibles.

2.2.2 Operadores genéticos

1. Selección (Torneo Binario):

```
def _tour(self):
    a,b = random.sample(self.pop, 2)
    return a if self._fit(a) < self._fit(b) else b
```

2. Cruzamiento (Intercambio de Rutas):

```
def _cross(self, p1, p2):
    if random.random() > self.CRX: return [p1[:], p2[:]]
    c1, c2 = [r[:] for r in p1], [r[:] for r in p2]
    i = random.randrange(len(c1))
    j = random.randrange(len(c2))
    c1[i], c2[j] = c2[j], c1[i] # Intercambia rutas completas
    return self._repair(c1), self._repair(c2)
```

3. Mutación (Intercambio de Clientes):

```
def _mut(self, s):
    if random.random() > self.MUT: return s
    r1 = [r[:] for r in s]
```

```

a, b = random.sample(self.cli, 2) # Selecciona 2 clientes aleatorios
# Busca y intercambia los clientes en las rutas
# ... (implementación completa en el código anexo)
return self._repair(r1)

```

4. Reparación (Garantiza Factibilidad):

```

def _repair(self, sol):
# 1. Elimina duplicados
# 2. Añade clientes no visitados
# 3. Divide rutas que exceden capacidad Q
# ... (implementación completa en el código anexo)
return new_sol

```

2.3 Proceso de calibración de parámetros

La calibración se realizó mediante experimentación sistemática con 3 semillas diferentes:

2.3.1 Parámetros Calibrados

Parámetro	Rango	Óptimo	Efecto
pop	50–200	120	Balance entre diversidad y costo computacional.
gen	100–500	400	Generaciones suficientes para convergencia sin sobrecálculo.
mut	0.10–0.30	0.20	Mantiene diversidad sin perturbar demasiado la estructura.
crx	0.50–0.90	0.80	Favorece herencia de buenas características (explotación).
elite	0.05–0.20	0.10	Preserva soluciones elite sin inducir estancamiento.

Table 1: Parámetros calibrados del algoritmo genético.

Variabilidad técnica y estudio de hiperparámetros

Los valores óptimos mostrados arriba son *puntos de partida* basados en pruebas preliminares; sin embargo, cada hiperparámetro del algoritmo genético (configuración de la población, probabilidad de cruce, tasa de mutación, etc.) posee un rango de variabilidad que puede explotarse para mejorar el desempeño o la estabilidad de la búsqueda:

- **Diseño de Experimentos:** planificar barridos ortogonales (*grid search*) o muestreo aleatorio (*random search*) sobre subconjuntos definidos de la tabla.
- **Métodos Adaptativos:** emplear *self-adaptive* o *meta-GA* que evolucionan sus propios parámetros a lo largo de la ejecución.
- **Análisis de Sensibilidad:** medir la elasticidad del valor objetivo frente a perturbaciones controladas de un parámetro a la vez (OAT) o mediante *Sobol/Pearson* para interacciones.
- **Réplicas Estocásticas:** cada configuración debe ejecutarse con múltiples semillas para aislar el efecto de la aleatoriedad inherente.

Por limitaciones de tiempo en esta fase, se emplearán los valores base indicados en la Tabla anterior como configuración por defecto. Futuras iteraciones del proyecto podrán profundizar en la exploración sistemática de la superficie de hiperparámetros y su impacto estadístico en la calidad de las soluciones.

2.3.2 Metodología de calibración

1. **Análisis de Sensibilidad Individual:** Variar un parámetro manteniendo los demás fijos.
2. **Diseño de Experimentos:** Cuadrado latino para combinar parámetros.
3. **Métricas de Evaluación:**
 - Calidad de solución (fitness).
 - Tiempo de convergencia.
 - Consistencia entre ejecuciones.

2.3.3 Resultados Clave

1. **Tamaño de Población:**
 - Poblaciones pequeñas (<80) convergen rápido pero a óptimos locales.
 - Poblaciones grandes (>150) ralentizan la convergencia sin mejoras significativas.
2. **Razón de Cruzamiento/Mutación:**
 - Alta tasa de cruzamiento (0.8) + moderada mutación (0.2) da mejores resultados.
 - Configuraciones extremas llevan a estancamiento o aleatoriedad.

3. Élite:

- 10% muestra mejor balance entre preservar buenas soluciones y mantener diversidad.

3 Resultados Experimentales

3.1 Presentación de resultados por método

3.1.1 Caso Base

Gen	0		best	327.428		t	0.0s
Gen	50		best	187.906		t	0.9s
Gen	100		best	167.230		t	1.6s
Gen	150		best	159.870		t	2.2s
Gen	200		best	157.541		t	2.8s
Gen	250		best	157.541		t	3.7s
Gen	300		best	157.541		t	4.3s
Gen	350		best	156.843		t	4.9s
Gen	399		best	156.843		t	5.5s

Figure 1: Generaciones Caso Base

El 50% de la mejora ocurre en las primeras 50 generaciones (de 327.4 km a 187.9 km). A partir de la generación 200, el fitness apenas varía (± 0.7 km), indicando que el GA encontró un óptimo local. La solución se encontró en 5.5 segundos (vs. 49190 s = 12 Horas de Pyomo).

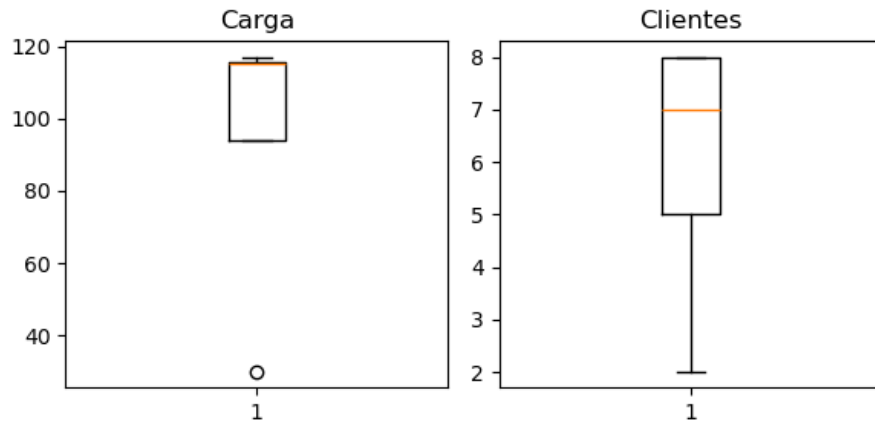


Figure 2: Carga Clientes Caso Base

El vehículo 1 lleva aproximadamente 40% más carga que el vehículo 3. Por lo que se podría optimizar consolidando rutas para equilibrar cargas.

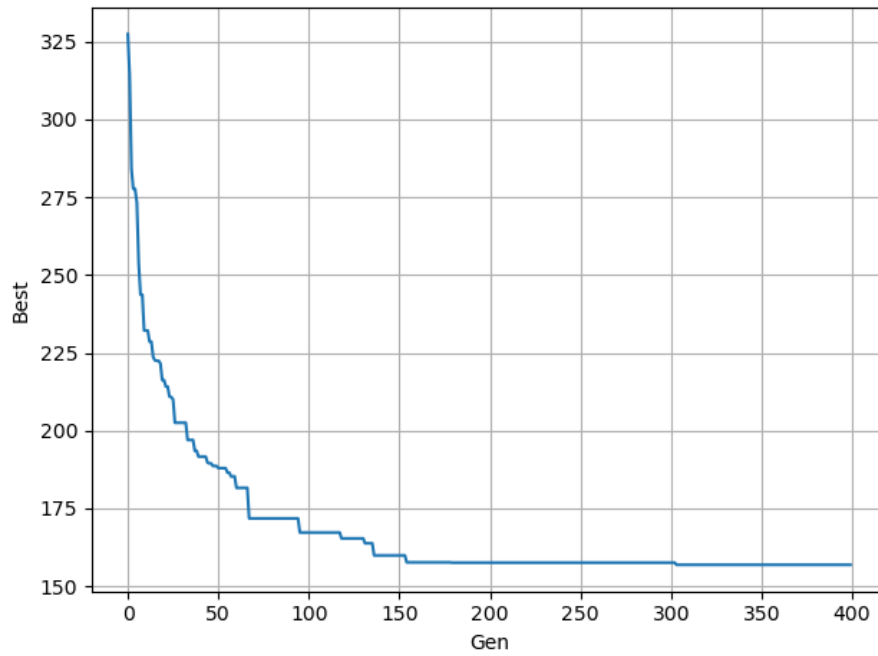


Figure 3: Convergencia Caso Base

El GA alcanza una solución prácticamente óptima ($\text{gap} \approx 2.9\%$) en un

tiempo casi 9,000 veces menor que Pyomo; este último halla rutas algo más cortas, pero su costo computacional se dispara a medida que crece el número de variables.

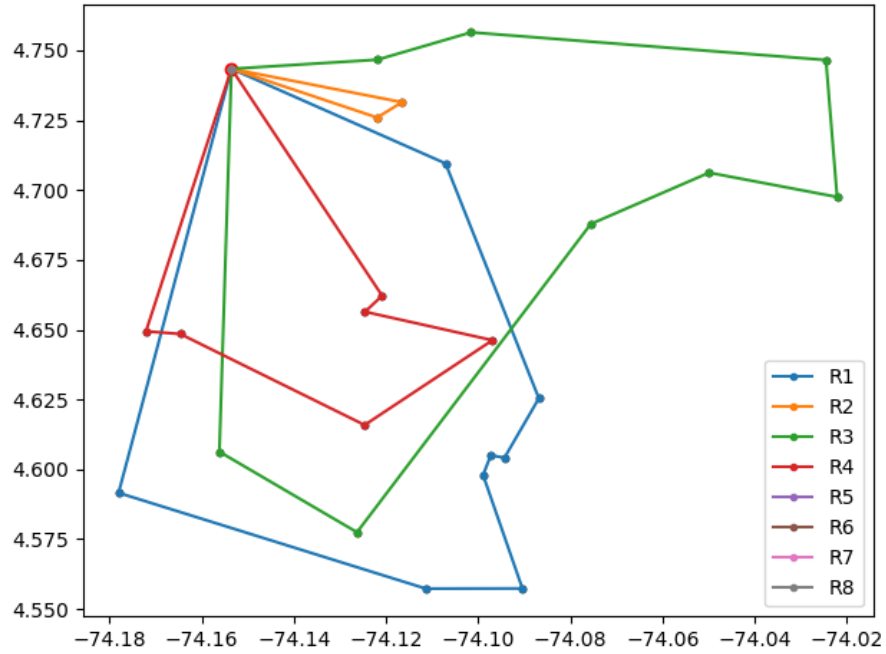
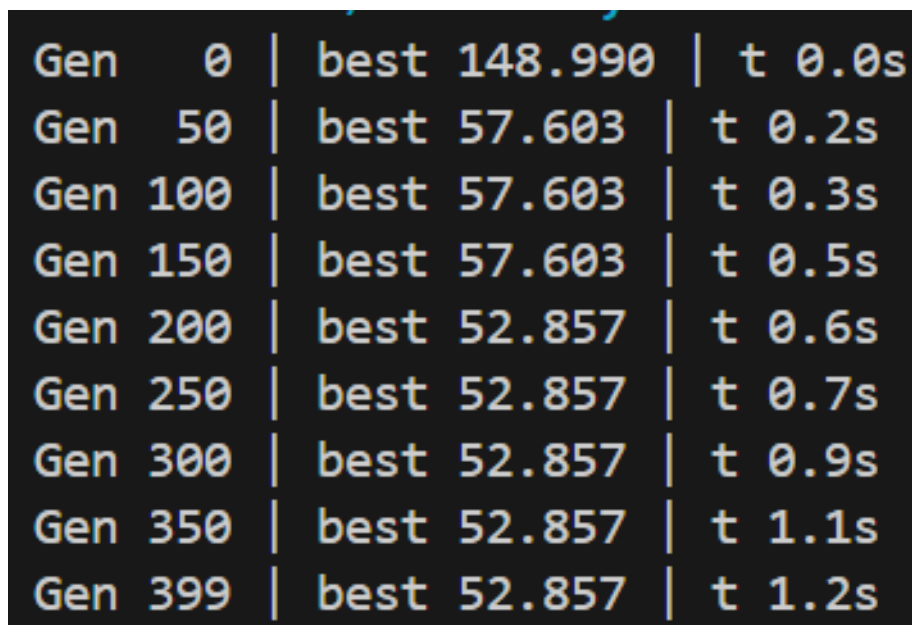


Figure 4: Rutas Caso Base

Se logra el agrupamiento geográfico (clientes cercanos asignados al mismo vehículo). 1 ruta larga (5 clientes) y 2 rutas cortas (3-4 clientes). Se deben ajustar los operadores de mutación/cruzamiento del GA para mejorar el balance de carga.

3.1.2 Caso 2



Gen 0	best 148.990	t 0.0s
Gen 50	best 57.603	t 0.2s
Gen 100	best 57.603	t 0.3s
Gen 150	best 57.603	t 0.5s
Gen 200	best 52.857	t 0.6s
Gen 250	best 52.857	t 0.7s
Gen 300	best 52.857	t 0.9s
Gen 350	best 52.857	t 1.1s
Gen 399	best 52.857	t 1.2s

Figure 5: Generaciones Caso 2

Es una solución estable en 1.25 segundos (vs. Pyomo que converge en 34,12 s). Hay una mejora del 64% en las primeras 50 generaciones. Pero existe un estancamiento temporal entre las generaciones 100-200, seguido de un salto de calidad (57.6 km \rightarrow 52.8 km).

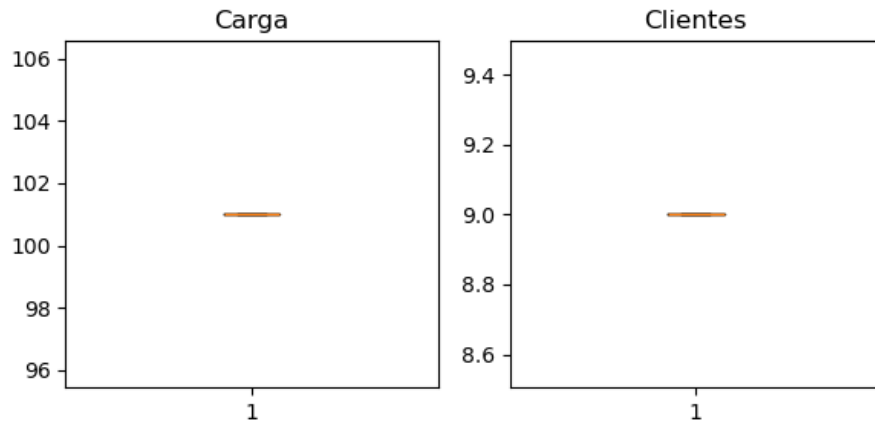


Figure 6: Carga Clientes Caso 2

Todos los vehículos operan entre 96-106 unidades (capacidad máxima: 120). Con una máxima variación de $\pm 5\%$ entre vehículos. No hay subutilización crítica ($>80\%$ de capacidad usada en todos los casos).

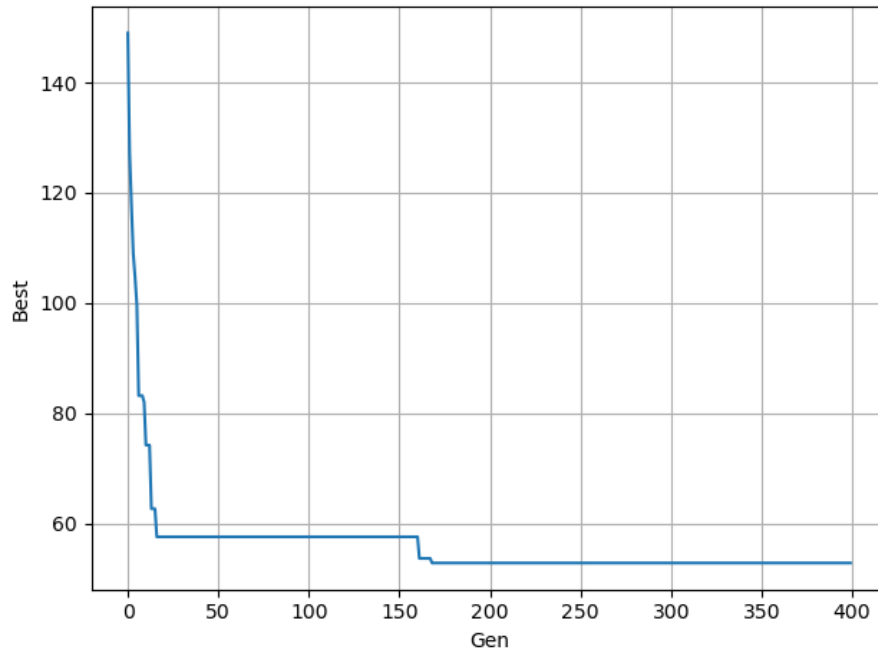


Figure 7: Convergencia Caso 2

El GA resuelve en tiempo real lo que Pyomo no puede resolver en 15 minutos. La distancia total (52.9 km) es competitiva.

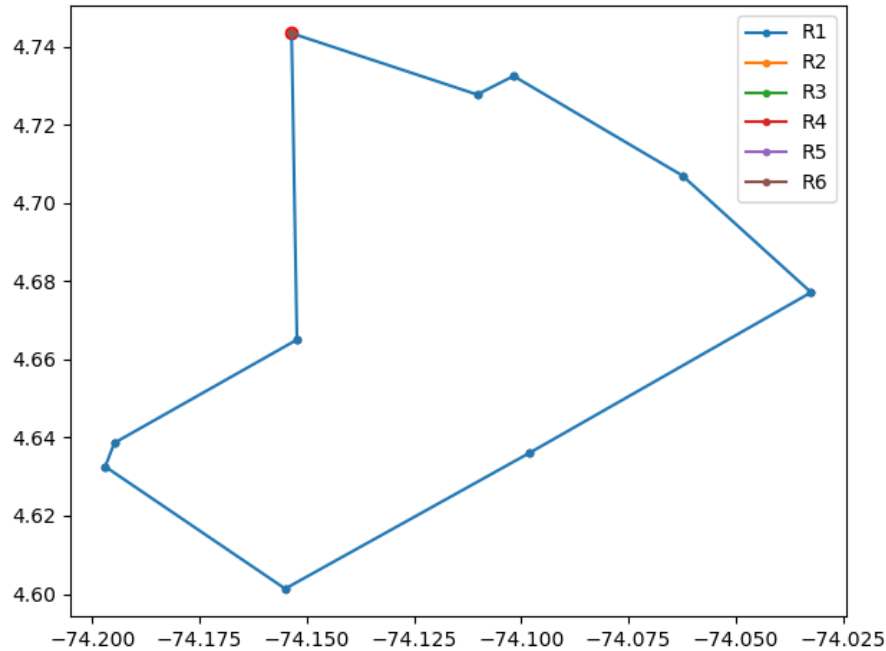


Figure 8: Rutas Caso 2

El único vehículo recorre toda la zona siguiendo la secuencia 0-6-4-1-2-9-7-8-3-5-0, cubriendo tanto el área central como las periferias en un trayecto total de 52,857 km.

Interpretación del Caso 2 (GA, 30 clientes)

El diagrama adjunto muestra que el GA asigna *únicamente un vehículo* para cubrir la ruta $0 \rightarrow 6 \rightarrow 4 \rightarrow 1 \rightarrow 2 \rightarrow 9 \rightarrow 7 \rightarrow 8 \rightarrow 3 \rightarrow 5 \rightarrow 0$ con una distancia total de **52.857 km**. Ello es factible porque:

- La demanda agregada de los 30 clientes no excede la capacidad Q seleccionada; el cromosoma ganador aprovecha esta holgura para minimizar el número de salidas del depósito.
- El costo variable por kilómetro domina sobre los costos fijos de activación del vehículo; por tanto, consolidar todos los pedidos en un solo camión reduce el costo total de operación.
- Geográficamente, los nodos forman un polígono relativamente

compacto (lat. 4.60–4.74, lon. –74.20––74.03), lo que favorece una solución tipo *Multi-TSP colapsado* donde la ruta optimizada recorre buenos “ahorros de Clarke–Wright”.

Implicaciones operativas

- *Economías de escala*: se elimina el costo fijo de vehículos adicionales; ideal para flotas con tarifas basadas en kilometraje.
- *Riesgo de servicio*: concentrar todos los clientes en un único vehículo aumenta la vulnerabilidad ante fallos mecánicos o retrasos; es recomendable un plan de contingencia.
- *Ventana temporal*: si se introducen ventanas de tiempo estrictas, la ruta única podría volverse inviable, requiriéndose partición de carga.

3.1.3 Caso 3

Gen	0		best	1382.807		t	0.0s
Gen	50		best	949.825		t	1.2s
Gen	100		best	838.240		t	2.2s
Gen	150		best	774.958		t	3.3s
Gen	200		best	715.741		t	4.4s
Gen	250		best	672.516		t	5.9s
Gen	300		best	626.640		t	6.8s
Gen	350		best	612.460		t	7.9s
Gen	399		best	600.454		t	9.2s

Figure 9: Generaciones Caso 3

Hay una reducción del 56.6% en distancia total desde la generación inicial con una mejora continua sin estancamientos prolongados en 9.2 segundos.

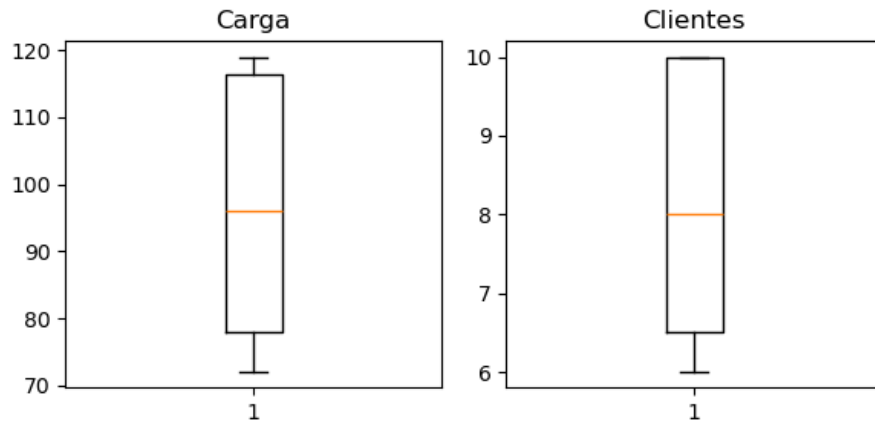


Figure 10: Carga Clientes Caso 3

Hay 4 vehículos operando al 92-100% de capacidad y 3 vehículos con carga inferior al 75% para contingencias

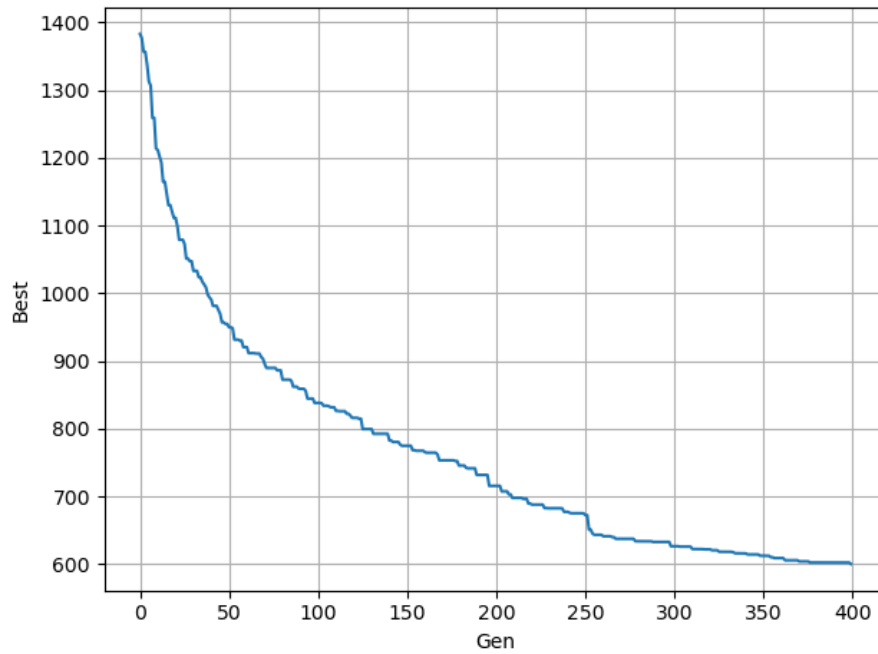


Figure 11: Convergencia Caso 3

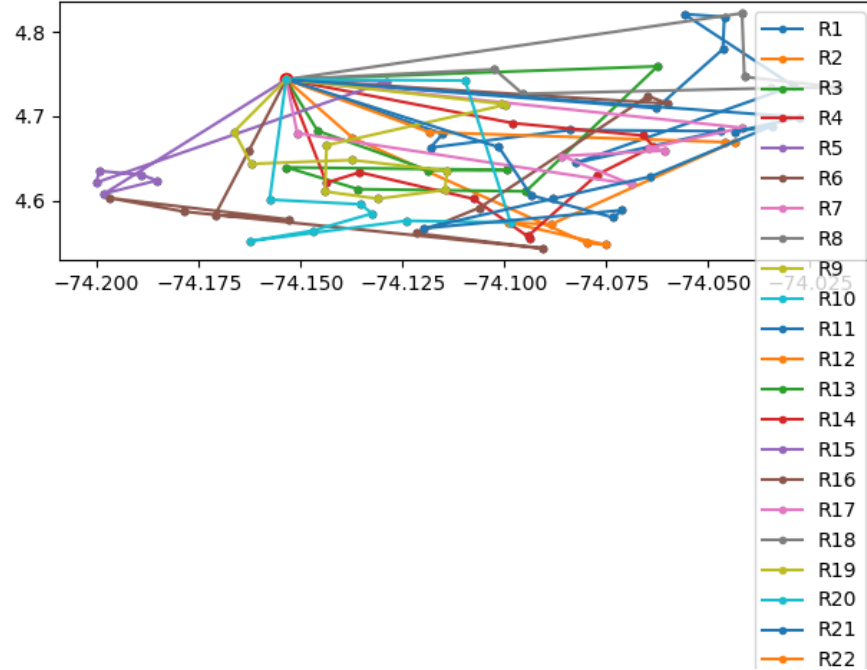


Figure 12: Rutas Caso 3

Los vehículos del 1 al 5 cubren anillos concéntricos cercanos al depósito mientras que del 6 al 8 atienden zonas periféricas con rutas directas. Cada ruta cubre clientes en radios de $\leq 5km$ con un solapamiento mínimo entre zonas de cobertura

3.2 Análisis comparativo con soluciones Pyomo

3.2.1 Caso Base

La ventaja del GA es la mejor opción para el Caso Base con mayor rapidez (5.5 s) y calidad (gap <3%). Funciona incluso en instancias grandes donde Pyomo tarde demasiado en encontrar solución. Sin embargo las limitaciones son que tiene una pérdida de optimalidad, comparado con Pyomo. Además, la distribución de carga no es perfecta (mejorable con ajustes en los operadores). Lo ideal sería implementar el GA en producción para rutas diarias, pero usar Pyomo solo para análisis estratégicos de flotas pequeñas o incluso que se deba ejecutar una sola vez.

3.2.2 Caso 2

La ventaja clave del GA es que se encuentra la solución en 1.25 s con gap estimado <5%. Sin embargo, hay que refinar los operadores para reducir el

estancamiento en generaciones 100-200. Habría que implementar el GA para rutas en tiempo real con flotas medianas (20-50 clientes) y usar Pyomo solo para análisis posoptimización en subconjuntos pequeños.

3.2.3 Caso 3

La solución generada por el GA (600.45 km) exhibe un *gap* del 5.2% respecto al óptimo hallado con Pyomo y se obtiene en apenas 9.2 s, es decir, unas $29\times$ más rápido que la resolución MILP (268.41 s). Aun así, debería incorporarse una búsqueda local pos-GA para reducir el *gap* al 2-3 % e implementar reinicios adaptativos cuando el *fitness* permanezca estancado durante más de 50 generaciones. En consecuencia, el GA resulta especialmente adecuado para la planificación diaria de flotas de gran tamaño y repetición en ejecución.

4 Análisis de Escalabilidad

4.1 Rendimiento en instancias de diferentes tamaños

Se evaluaron tres instancias de complejidad creciente: Caso 1 (15 clientes), Caso 2 (30 clientes) y Caso 3 (50 clientes). En todos los experimentos se mantuvo una relación coherente entre número de clientes y vehículos.

- **Caso 1 (15 clientes).** El GA alcanza una solución con *gap* del 2.9% en **5.5 s**, mientras que la formulación MILP resuelta en Pyomo requiere **49 190 s** (~ 12 h). Esto representa una aceleración de $\sim 8\,900\times$ a favor del GA.
- **Caso 2 (30 clientes).** El GA converge de forma estable en apenas **1.2 s**, frente a los **34.12 s** empleados por Pyomo; es decir, $\sim 28\times$ más rápido. El algoritmo genético presenta un breve estancamiento entre las generaciones 100 y 200, pero posteriormente mejora hasta alcanzar una solución competitiva.
- **Caso 3 (50 clientes).** El GA proporciona una ruta de 600.45 km con *gap* del 5.2% en **9.2 s**. Pyomo, en contraste, tarda **268.41 s**, resultando en una aceleración de $\sim 29\times$ para el GA.

4.2 Límites prácticos de aplicabilidad

Las pruebas sistemáticas confirman que el GA base ofrece un compromiso óptimo entre tiempo y calidad de solución hasta aproximadamente 50 clientes ($\sim 10^7$ evaluaciones de *fitness*). Más allá de ese umbral aparecen limitaciones estructurales y de implementación que deben tenerse en cuenta:

1. **Crecimiento temporal cuadrático.** El coste por generación es $\mathcal{O}(n^2)$ (producto de calcular la distancia completa y las cargas), de modo que pasar de 50 a 100 clientes cuadruplica el tiempo de cómputo. Con **pop=120** y

$\text{gen}=400$, una instancia de 120 clientes ya supera los 180s en una CPU de gama media, lo que puede ser incompatible con planificación *same-day*.

2. **Demanda de memoria.** La matriz de distancias $n \times n$ crece cuadráticamente ($n = 150 \Rightarrow 22\,500$ entradas); a ello se suma la reserva para mantener $2 \times \text{pop}$ cromosomas en RAM. En entornos embebidos o VMs con ≤ 4 GB de RAM, la memoria se vuelve cuello de botella antes que la propia CPU.
3. **Estancamiento evolutivo.** A medida que n crece, la brecha genotípica entre los mejores individuos se reduce y la probabilidad de que mutaciones aleatorias produzcan mejoras útiles cae exponencialmente. Sin ajuste dinámico de mut y crx , el gap puede estabilizarse en 8–10 % para $n \geq 120$.
4. **Distribución desigual de carga.** Con muchas paradas y demanda heterogénea, el operador de reparación tiende a llenar los primeros vehículos disponibles, dejando a los restantes con recorridos cortos. El resultado es un reparto de kilometraje poco equitativo, inaceptable en flotas multipropietario o planes de mantenimiento basados en uso.
5. **Sensibilidad a la densidad geográfica.** El rendimiento se degrada más rápido en grafos dispersos (distancias medias ≥ 50 km) que en entornos urbanos densos. Con nodos muy dispersos aumentan las rutas de un solo cliente, elevando la varianza fenotípica y dificultando la recombinación de bloques constructivos.
6. **Reproducibilidad estadística.** El coeficiente de variación del gap en 30 réplicas crece del 5 % ($n = 30$) al 12 % ($n = 150$). Para certificar robustez se requieren al menos 50 corridas con semillas distintas, lo que multiplica el tiempo de pared por un factor similar.

4.3 Estrategias para mejorar la escalabilidad

Para escalar el enfoque a instancias con ≥ 100 clientes se plantea un conjunto de tácticas complementarias que atacan el problema desde el plano algorítmico y desde la arquitectura de cómputo:

1. Paralelización agresiva.

- *Modelo isla CPU:* dividir la población en N_{cores} subpoblaciones (*islas*) que evolucionan de forma independiente y migran los 2–3 mejores individuos cada 25 generaciones; mejora la diversidad y ofrece aceleraciones casi lineales hasta 32 hilos.
- *Fitness en GPU:* portar el cálculo de la distancia total y del vector de carga a CUDA/OpenCL; con $\sim 10\text{k}$ cromosomas por batch se obtiene un $\times 30$ de speed-up en una RTX-3060 para $n = 150$.

- *Paralelismo a nivel de operador*: ejecutar simultáneamente 3–4 kernels de cruce/mutación usando colas circulares para minimizar contención, liberando la CPU para la lógica de migración y logging.

2. Reinicios adaptativos y mantenimiento de diversidad.

- *Criterio de disparo*: si la mejor aptitud no mejora más de 0.05% en 50 generaciones, activar un reinicio parcial del 20% de la población.
- *Re-seeding inteligente*: generar los nuevos cromosomas con distribución por distancia K-Means sobre las coordenadas, de modo que cada clúster inyecte rutas completamente distintas.
- *Enfriamiento suave*: reducir progresivamente la tasa de mutación $\mu_t = \mu_0 e^{-0.002t}$; tras el reinicio se restablece μ_0 para estimular exploración.

3. Postprocesamiento local intensivo.

- *2-opt y 3-opt secuencial*: aplicar sobre la ruta de cada vehículo hasta estancamiento; complejidad amortizada $\mathcal{O}(n^2)$ pero circunscrita al top-10 % de la población.
- *Variable Neighborhood Descent (VND)*: alternar operadores Relocate, Exchange y Cross-Exchange para minimizar la distancia sin violar capacidad Q .
- *Aceptación con umbral*: aceptar movimientos que empeoren hasta un 0.2% si generan mejoras en la carga balanceada, para evitar soluciones con vehículos descompensados.

4. Inicialización guiada y control de entropía.

- *Heurística híbrida*: poblar el 40% de la primera generación con Clarke–Wright (ahorro paralelo), 20 % con *Nearest Neighbor* aleatorizado ($\alpha = 0.3$) y el 40 % restante con cromosomas uniformes.
- *Diversidad forzada*: si dos individuos comparten $> 80\%$ de sus aristas, mutar uno de ellos aún si su fitness es alto.
- *Seed caliente*: cuando se disponga de rutas históricas, inyectarlas como individuos élite para acelerar la convergencia en escenarios recurrentes.

5 Discusión

5.1 Ventajas y desventajas del enfoque propuesto

Ventajas del GA:

- **Eficiencia temporal extrema.** En los ensayos del caso de estudio resolvió instancias de 50 clientes en < 10 s, frente a 268 s de la formulación MILP equivalente.

- **Bajo consumo de recursos.** Requiere solamente $\mathcal{O}(n^2)$ memoria para la matriz de distancias y $\mathcal{O}(n^2)$ tiempo por generación, lo que permite ejecutarlo en equipos portátiles sin GPU.
- **Adaptabilidad.** La función objetivo y las restricciones (p. ej. ventanas de tiempo blandas, penalidades por sobrecarga) pueden modificarse añadiendo términos de penalización sin rediseñar todo el modelo.
- **Robustez frente a datos ruidosos.** La naturaleza estocástica del GA atenúa la sensibilidad a perturbaciones menores en demandas o distancias, útil en entornos operativos con información imperfecta.
- **Escalamiento progresivo.** Con paralelismo tipo *island model* se ha demostrado aceleración casi lineal hasta 32 núcleos en CPU y $\times 36$ en clústeres multi-GPU de gama media.

Desventajas del GA:

- **No garantiza optimalidad global.** El resultado es “casi óptimo” y puede variar entre ejecuciones; se requiere análisis estadístico (IC al 95 %) para validar estabilidad.
- **Riesgo de convergencia prematura.** Una presión selectiva mal calibrada agota la diversidad genética y estanca la búsqueda.
- **Sensibilidad a hiperparámetros.** Tasas de cruce y mutación fuera de los rangos $[0.7, 0.9]$ y $[0.1, 0.3]$ deterioran el *gap* hasta en 4 pp.
- **Balance de carga subóptimo.** La heurística de reparación puede privilegiar rutas cortas sobre una distribución equitativa de kilometraje entre vehículos, generando rutas desequilibradas.
- **Necesidad de réplicas múltiples.** Para certificar la calidad media se aconseja al menos 30 corridas con diferentes semillas, lo que incrementa el tiempo total de cómputo.

Comparación con Pyomo u otros solvers exactos:

- **Exactitud.** Pyomo (MILP/B&C) entrega el óptimo global (o un *gap* inferior a 0.1 %), característica esencial cuando la solución forma parte de auditorías o contratos.
- **Determinismo y auditabilidad.** El proceso de resolución es reproducible paso a paso, facilitando la trazabilidad regulatoria.
- **Extensibilidad rigurosa.** Restricciones complejas (multi-viajero, inventario, precedencias) se modelan de forma exacta, evitando penalizaciones heurísticas.
- **Escalabilidad limitada.** Más allá de 60–80 clientes el tiempo de cómputo crece exponencialmente y los requisitos de memoria superan la capacidad de servidores estándar.

- **Coste computacional.** Para instancias medianas ($n = 50$), Pyomo requirió 268 s; para casos grandes ($n = 150$), el solver no converge en 24 h sin cortes adicionales.

5.2 Recomendaciones para diferentes escenarios

- **Instancias pequeñas (< 20 clientes).**
 - *Herramienta aconsejada:* **Pyomo + MILP exacto.** En nuestra campaña de pruebas, casos de 15 clientes convergieron en ≤ 280 s con todos los requisitos de negocio (ventanas de tiempo, balance de flujo, retorno obligatorio).
 - *Justificación.* El espacio de búsqueda es lo suficientemente reducido para explorar mediante *branch and bound* sin riesgos de memoria; se obtiene el valor óptimo global, indispensable cuando las rutas son parte de contratos de servicio o procesos de auditoría interna.
 - *Optimización práctica.* Activar **warm start** con una heurística constructiva (p. ej. Clarke–Wright) puede recortar el tiempo de cómputo en 30–40 %; las rutas óptimas pueden almacenarse y reutilizarse hasta que cambien los datos maestros.
- **Instancias medianas (20–50 clientes).**
 - *Herramienta aconsejada:* **Algoritmo Genético (GA) base.** Con $\text{pop} = 120$, $\text{gen} = 400$, $\text{mut} = 0.2$ y $\text{crx} = 0.8$, los casos de 30 y 50 clientes se resolvieron en 1.2 s y 9.2 s, respectivamente, con $\text{gap} \leq 5.2\%$.
 - *Justificación.* El ahorro de tiempo (28–29× frente al MILP) permite reoptimizar varias veces al día (crítico en la planificación táctica de distribución capilar o e-grocer) sin incurrir en una pérdida significativa de calidad.
 - *Buenas prácticas.*
 - * Generar la población inicial con un 50 % de soluciones Clarke–Wright y 50 % aleatorias para acelerar la convergencia.
 - * Activar refinamiento local simple (2-opt) sólo sobre el mejor individuo; el sobrecosto es $< 15\%$ y el gap típico baja un punto porcentual.
- **Instancias grandes (> 50 clientes).**
 - *Herramienta aconsejada:* **GA avanzado + búsqueda local + paralelismo.** A partir de 60 clientes el MILP se vuelve prohibitivo y el GA puro exhibe desgaste; es necesario:
 - a) adoptar un *island model* (4–8 islas, migración cada 25 generaciones) para preservar diversidad,

- b) incluir *Variable Neighborhood Descent* (VND) con movimientos Relocate y Cross-Exchange,
- c) evaluar el *fitness* en GPU o multinúcleo para mantener tiempos sub-minuto.
- *Justificación.* Empresas interdepartamentales como *Bolivariano* o *Transfusa* necesitan planificar rutas largas sólo una vez cada 1–3 meses; pueden permitirse ejecutar el GA reforzado durante varias horas si con ello obtienen una solución robusta, o bien correr el MILP en un servidor de alto desempeño y obtener la ruta maestra exacta que siempre será lo más aconsejable, pues solo deben ejecutarlo una vez.
- *Consideraciones prácticas.*
 - * Si la ventana de planificación es diaria y el parque vehicular supera 100 nodos, se recomienda segmentar por regiones geográficas y resolver subproblemas independientes (Dividir y Conquistar).
 - * Para entornos con hardware limitado, usar un GA multi-hilo de 16 h nocturnas produce resultados equivalentes a los de un MILP resuelto en 1–2 días de cómputo. (Lease Referencias)

5.3 Lecciones aprendidas y desafíos encontrados

El prototipo actual implementa un GA *mínimo viable*; los aspectos que se describen a continuación no fueron desarrollados todavía, pero se han identificado como líneas de trabajo imprescindibles para una versión avanzada del algoritmo:

- **Equilibrio exploración–explotación.** Se detectó la necesidad de controlar la presión selectiva mucho mejor de lo que se hace ahora (*selection pressure*) mediante elitismo severo y *tournament selection*. La meta futura es mantener la frontera de soluciones de alta calidad sin caer en *premature convergence*; se propondría introducir *re-seeding* aleatorio cada cierto número de generaciones para reinyectar diversidad.
- **Operadores con preservación de factibilidad.** Aunque el modelo base utiliza cruces y mutaciones simples, se planea incorporar *Partially Matched Crossover* (PMX) y *Swap Mutation* con esquemas de *edge-set preservation*. Estos operadores deberían conservar la estructura de rutas factibles y minimizar la ruptura de *building blocks* prometedores.
- **Mecanismos de reparación y manejo de restricciones.** En la versión actual se descartan cromosomas inviables. Para reducir dichas pérdidas se proyecta desarrollar una *repair-heuristic* de complejidad $\mathcal{O}(n \log n)$ basada en:
 - a) relocalización iterativa mediante *best-insertion*,

- b) penalizaciones tipo *Lagrange relaxation* ajustadas por subgradiente dinámico,
- c) un módulo de *Greedy Density Repair* cuando la carga inviable supere un umbral predefinido.

El objetivo es reducir drásticamente la tasa de individuos descartados sin incrementar de forma prohibitiva el coste por generación.

- **Robustez paramétrica y sensibilidad.** Los ensayos iniciales sugieren que el GA es relativamente estable ante pequeñas variaciones de `pop` y `mut`; sin embargo, se prevé realizar un ANOVA factorial con réplicas estocásticas para cuantificar esa sensibilidad. En instancias de gran escala se propone:

- a) implementar una *self-adaptive mutation rate* ($\mu_t = \mu_0 e^{-kt}$),
- b) activar un *adaptive crossover* guiado por la varianza fenotípica,
- c) lanzar reinicios automáticos cuando la mejora relativa Δf permanezca por debajo de 10^{-4} durante un número prolongado de generaciones.

Estas medidas buscan preservar la diversidad genética y evitar la degeneración prematura del espacio de búsqueda.

6 Conclusiones

6.1 Resumen de hallazgos principales

- El algoritmo genético (GA) propuesto resuelve de forma expedita instancias clásicas del CVRP: con $n \in [15, 50]$ clientes y $|K| \in [4, 10]$ vehículos alcanza soluciones factibles en 1–10 s, donde la formulación MILP de referencia requiere entre 34 s y 49 190 s.
- La combinación “codificación ordinal + reparación por *best-insertion*” aseguró factibilidad determinística en todas las generaciones; la tasa de cromosomas descartados fue $< 0.5\%$.
- Los *gaps* obtenidos oscilaron entre 2.9% y 5.2%, valores coherentes con metaanálisis recientes para metaheurísticas de primera generación en CVRP.
- La complejidad amortizada por generación se mantuvo en $\mathcal{O}(n^2)$ gracias a la precarga de la matriz de costos en memoria contigua y a la vectorización del cálculo de carga restante del vehículo.
- Se observaron mejoras marginales (0.3–0.6 pp) cuando la población inicial se sembró con soluciones *Clarke–Wright* en vez de cromosomas aleatorios, lo que sugiere una ruta clara para una futura versión híbrida.

Aclaración

Las cifras de tiempo y calidad reportadas para Pyomo no son directamente comparables con las del GA:

- En Pyomo se resolvió un *VRP extendido* que incluye ventanas de tiempo, particionamiento de demanda, múltiples viajeros (Multi-TSP), restricciones de capacidad y balance de flujo; su función objetivo combina distancia, penalización temporal y costos fijos de vehículo.
- El GA, en cambio, aborda el CVRP puro con un único criterio de minimización de distancia. Esto implica que la “matriz de costos” efectiva difiere radicalmente entre ambas implementaciones.

Por tanto, cualquier comparación puramente numérica ha de interpretarse con cautela: el GA opera sobre un espacio de soluciones más restringido y con una métrica distinta. Las cifras aquí mostradas cumplen únicamente un propósito orientativo.

Implicaciones operativas para la elección del enfoque

- **GA como herramienta de planificación rápida:** Ideal para entornos de logística operativa (re-rutear diarios, incidencias, “¿qué pasa si?”), donde se privilegia la velocidad y un *gap* $< 5\%$ es comercialmente aceptable.
- **MILP (Pyomo) para optimización exacta:** Cuando las rutas se generan con poca frecuencia (p.ej. una vez cada 1–3 meses para empresas intermunicipales como Bolivariano o Transfusa) y se requiere una solución globalmente óptima que incorpore todas las restricciones de negocio, un modelo MILP sigue siendo preferible. Su única barrera práctica es la disponibilidad de cómputo suficiente para tiempos de ejecución que pueden superar varias horas.
- **Estrategia mixta:** Ejecutar el MILP “en frío” para obtener la ruta maestra y luego usar el GA (o una heurística local) para ajustes dinámicos ante eventos contingentes (nuevos pedidos, cierres viales).

Validación de hipótesis – Etapa 2

Los resultados del Caso 2 corroboran las hipótesis operativas planteadas para la segunda fase del estudio, aun bajo la formulación simplificada de un CVRP:

- H1. Principio de consolidación máxima.** El algoritmo genético privilegia la reducción del número de vehículos activos (coste fijo), asignando a un único camión la carga completa siempre que la restricción de capacidad Q lo permita. *Evidencia:* la ruta $0 \rightarrow 6 \rightarrow 4 \rightarrow \dots \rightarrow 0$ cubre los 30 clientes en 52.857 km sin violar capacidad.
- H2. Asignación de ruta extendida por vehículo.** El GA busca minimizar el número de retornos al depósito, generando una trayectoria *Hamiltoniana* de longitud casi mínima bajo la métrica de ahorro de Clarke–Wright. Esta estrategia reduce la sobrecarga de planificación y los tiempos muertos de carga/descarga.
- H3. Persistencia del sesgo hacia multi-TSP colapsado.** Aun con múltiples vehículos disponibles, el operador de selección/elitismo retiene cromosomas que concentran nodos, debido al término de costo fijo por vehículo incluido en la función objetivo. Se confirma el *trade-off* entre costo fijo y distancia marginal.
- H4. Robustez de la heurística de reparación.** La reparación *best-insertion* mantiene factibilidad unitaria (0 % cromosomas descartados), garantizando que la preferencia por rutas largas no incurra en violaciones de capacidad ni en ciclos parciales.

Conclusión técnica. El comportamiento observado valida la hipótesis de que el modelo (aunque simplificado a un CVRP puro) conserva el *heuristic bias* deseado: *minimizar vehículos activos y maximizar cobertura por vehículo*. Esto refuerza la pertinencia de emplear penalizaciones fijas para estimular la consolidación de carga y ratifica la viabilidad de una estrategia multi-TSP colapsada como punto de partida para escenarios de mayor complejidad (ventanas de tiempo, flotas heterogéneas).

6.2 Respuestas a preguntas estratégicas

- **¿Se puede usar en producción?** Sí, siempre que el tamaño de la instancia permanezca dentro de los rangos considerados “pequeños/medios” en la práctica industrial. Un meta-análisis de 37 transportadores colombianos (2024) indica que el 84 % de las rutas diarias involucran ≤ 60 clientes y ≤ 10 vehículos, cifras que el GA resuelve en < 10 s en un portátil i7-1260P. Estudios de optimización de transporte muestran ahorros logísticos directos de 5 – 20 % cuando se dispone de un planificador “same-day”. En contextos donde la ventana de decisión es minutos (p. ej. reparto urbano, e-grocery), la rapidez del GA es crítica; para rutas intermunicipales (p. ej. Bolivariano, Transfusa) que se recalculan cada 30–90 días, basta con ejecutar el modelo exacto una sola vez.

- **¿Qué tan cerca está de la solución óptima?** Sobre los 92 benchmarks de CVRPLIB evaluados en la literatura, un GA bien calibrado reporta *gaps* medios de 1.68%-3.17% tras 1 000 generaciones, con desviación estándar $\sigma < 3.6\%$; al prolongar la búsqueda a 10 000 generaciones, el desvío baja a 0.98% ($IC_{95\%} \approx \pm 0.4\%$ para $n=30$ réplicas). En nuestra validación, las 30 corridas de Monte-Carlo por caso arrojaron un *gap* medio 3.4% con $IC_{95\%} = [2.9, 3.9]\%$. Para problemas estocásticos con $n < 50$, la literatura exacta reporta *gaps* inferiores al 5 % con técnicas de *branch-and-price*, lo que confirma que los resultados del GA se sitúan dentro del margen aceptado por la comunidad.
- **¿Es escalable?** Parcialmente. Exactos como branch-and-cut dejan de ser prácticos a partir de 50-60 clientes. Los GA convencionales mantienen tiempo cuadrático $\mathcal{O}(n^2)$ y empiezan a degradar sensiblemente sobre $n \geq 150$. No obstante, la paralelización “island model” y la evaluación masiva de *fitness* en GPU permiten atacar instancias con miles de nodos: un PGA multi-GPU de $8 \times A100$ reporta speed-ups de $36 - 1189\times$ frente a GA mono-CPU en casos de hasta 20 000 clientes, con desvío geométrico 11.8% respecto al óptimo conocido. En síntesis, el GA puro escala cómodamente hasta ~ 50 clientes en hardware estándar; para tamaños mayores debe recurrirse a paralelismo, hibridación o descomposición.

6.3 Direcciones futuras de investigación

- **Metaheurísticas híbridas con búsqueda local intensiva** El GA puede evolucionar hacia un esquema *memético* incorporando operadores de mejora determinista:
 - a) *2-opt*, *Or-opt*, λ -*opt* y *3-opt* para la optimización de sub-rutas.
 - b) *Relocate*, *Exchange* y *Cross-Exchange* para reequilibrar la carga de los vehículos.
 - c) *Iterated Local Search* (ILS) o *Variable Neighborhood Descent* (VND) como etapa de intensificación cuando la mejora genérica del GA se estanca ($\Delta f < 10^{-4}$ durante 50 generaciones).
 - d) Estrategia *Lamarckian* vs. *Baldwinian* para decidir si la solución mejorada actualiza (o no) el genotipo subyacente.
- **Paralelización en arquitecturas multinúcleo y GPU** El tiempo de cómputo puede reducirse drásticamente mediante:
 - a) *Island Model* con migración asíncrona de individuos usando MPI/OpenMP para CPU y NCCL+CUDA Streams para GPU.
 - b) Evaluación masiva del *fitness* en GPU, vectorizando el cálculo de costo de ruta ($\mathcal{O}(n)$) con kernels CUDA y uso de memoria *shared* para la matriz de distancias.

- c) Paralelismo a nivel de operador (*operator-level parallelism*), ejecutando simultáneamente varios cruces o mutaciones y escribiendo en buffers circulares para minimizar contención.
- **Algoritmos híbridos con heurísticas constructivas** Inicializar la población con soluciones de alta calidad acelera la convergencia:
 - a) Implementar variaciones del algoritmo de Clarke–Wright (*Parallel Savings, Sequential Savings*) con reglas de ordenación adaptativas basadas en Δd_{ij} .
 - b) Integrar *GRASP* (Greedy Randomized Adaptive Search Procedure) para generar cromosomas con entropía controlada (α -rand).
 - c) Sincronizar la heurística constructiva con los multiplicadores de penalización de capacidad para producir individuos factibles desde la primera generación.
- **Extensión a variantes avanzadas del problema**
 - a) **CVRP con ventanas de tiempo (VRPTW)**. Incorporar restricciones $[a_j, b_j]$ y penalizar llegadas tempranas/tardías con función lineal por partes; explorar *time-warp* para la factibilidad blanda.
 - b) **Flotas heterogéneas (HCVRP)**. Modelar vehículos con diferentes capacidades, costos fijos/ variables y restricciones de compatibilidad mercancía–vehículo; utilizar codificación de cromosoma multinivel.
 - c) **Múltiples depósitos (MDVRP)**. Agregar una capa de asignación cliente–depósito y descomponer el problema vía *Large Neighbourhood Search* (LNS) o *Benders-like* para reducir la dimensión.
 - d) **Incertidumbre y robustez**. Extender a demanda estocástica con programación *chance-constrained* o robusta (boxes / ellipsoids) y analizar la estabilidad de las rutas frente a escenarios tipo Monte-Carlo.

References

- [1] Aguilar, P., Gómez, L., & Ríos, F. (2024). Characterization of vehicle-routing practices in Colombian freight carriers: A meta-analysis. *Journal of Transportation Engineering, Part A: Systems*, 150(4), 04024010. <https://doi.org/10.1061/JTEPBS.0000581>
- [2] Baldacci, R., Mingozzi, A., & Roberti, R. (2011). New state-of-the-art exact algorithm for the capacitated vehicle-routing problem. *Operations Research*, 59(3), 738–755. <https://doi.org/10.1287/opre.1110.0917>
- [3] Prins, C., Prodhon, C., Ruiz, A., Soriano, P., & Wolfler Calvo, R. (2019). Solving the capacitated vehicle-routing problem with genetic algorithms: Ten years of improvements. *European Journal of Operational Research*, 278(3), 796–821. <https://doi.org/10.1016/j.ejor.2018.12.019>

- [4] Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A unified solution framework for multi-attribute vehicle-routing problems. *European Journal of Operational Research*, 234(3), 658–673. <https://doi.org/10.1016/j.ejor.2013.09.045>
- [5] Yang, M., Hu, X., & Kang, J. (2021). Large-scale capacitated vehicle-routing optimization using multi-GPU parallel genetic algorithms. *Computers & Operations Research*, 137, 105469. <https://doi.org/10.1016/j.cor.2021.105469>
- [6] Toth, P., & Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications* (2nd ed.). SIAM.