**Universidad de los Andes**

# Proyecto etapa 3

Maria Alejandra Angulo  Laura Sofia Murcia  Samuel Ramirez Gómez

## 1 Descripción del problema

El problema que propone el proyecto es un problema TSP multiple con la condición de que cada agente viajero debe cumplir en sus viajes una demanda de cada uno de los nodos que visita. Para solucionar este problema se plantearon dos alternativas. La primera es un modelado matematico en pyomo que busca resolver el problema de manera exacta y óptima. Esta solución es buena para problemas pequeños y medianos, pero puede tener problemas con casos más grandes. Es por esto que se plantea una segunda solución con el uso de metaheuristicas, en este caso un algoritmo genético. El cual busca la mejor solución aproximada al problema.

Primero se define la solución basada en modelo matematico pyomo.

### 1.1 Pyomo

#### 1.1.1 Conjuntos

- $CD = \{CD1, CD2, CD3\}$ que corresponden a los Centro de distribución
- $C = \{C1, C2, C3\}$ que corresponde a los clientes (denominados por clientes por facilidad del ejemplo, en la cotidianidad es la zona entrega)
- $A = \{(i, j)\} \quad i, j \in CD \cup C$ que simbolizan los arcos del grafo de recorrido CD a C
- $V = \{V1, V2, V3\}$ que corresponde a los vehículos con los que cuenta la empresa.
- $N = CD \cup C$ Para facilidad de notación de los conjuntos posibles de (i,j)

#### 1.1.2 Índices

- $i, j \in N$ que simbolizan las ubicaciones por las que se tienen que mover, donde $i$ es el nodo de inicio y $j$ es el nodo destino
- $v \in V$ que simboliza el vehículo disponible de la flota de distribución.
- $c \in C$ que representa cada cliente del problema
- $d \in CD$ que representa cada centro de distribución

#### 1.1.3 Parámetros

- $d_{ij}$ Que determina la distancia entre nodos $i, j$
- $t_{ij}$ Determina el tiempo de viaje entre los nodos $i, j$
- $D_i$ Simboliza la demanda del cliente

- $K_d$ Simboliza la capacidad de cada centro de distribución
- $Q_v$ Simboliza la capacidad del vehículo
- $R_v$ Simboliza el Rango útil del vehículo
- $P_f$ Simboliza el precio de combustible por litro
- $G_v$ Simboliza el consumo de gasolina en litros por km
- $F_t$ Simboliza la tarifa del flete por km
- $C_m$ Simboliza el costo de mantenimiento por km
- $S_i$ Simboliza el tiempo de entrega (cuanto tiempo se demora el distribuidor con el cliente)
- $c_{ij}$ Costo de operación de punto $i$ a punto $j$ que se compondrá de los datos anteriores. $c_{ij} = d_{ij} \cdot (G_v P_f + F_t + C_m)$

### 1.1.4 Variables de Decisión

Este es un problema mixto, que cuenta con variables binarias y variables racionales

- $x_{ijv} \in \{0, 1\}$ Donde es 1 si el vehículo $v$ viaja del punto $i$ al punto $j$, y 0 de lo contrario
- $t_{iv} \in \mathbb{Q}$ Que indica el tiempo en el que el vehículo $v$ atiende a atender al cliente $i$
- $y_{iv} \in \{0, 1\}$ Es 1 si el cliente $i$ es atendido por el vehículo $v$, 0 en caso de lo contrario.
- $z_{id} \in \{0, 1\}$ Es 1 si el cliente $i$ es suplido por el centro de distribución $d$, y 0 en caso contrario.
- $u_{vi} \in \{1, N-1\} \in \mathbb{Z}^+$ Esta es la variable auxiliar que se utiliza para la restricción de los subtoures MTZ para cada vehiculo v.
- $r_{vd} \in \{0, 1\}$ es 1 si el vehiculo $v$ sale del deposito $d$. De lo contrario es 0

### 1.1.5 Función Objetivo

La discusión se puede encontrar en la sección **??**, sin embargo se define la función objetivo como

$$\min \sum_{v \in V} \sum_{(i,j) \in A} c_{ij} x_{ijv} \tag{1}$$

Que busca calcular los costos mínimos con $c_{ij}$ y evaluar si un vehículo específico hace la ruta $(i, j)$ con $x_{ijv}$.

### 1.1.6 Restricciones

**Cada cliente es atendido una vez**

$$\sum_{v \in V} \sum_{j \in N} x_{ijv} = 1 \quad \forall i \in C \tag{2}$$

Evalúa que no haya más de un vehículo asignado para cada cliente, con el propósito de no desperdiciar recursos

**Flujo de vehículos**

$$\sum_{j \in C} x_{ijv} - \sum_{j \in C} x_{jiv} = 0 \quad \forall i \in C, v \in V \tag{3}$$

Evalúa que si un vehículo entrega a un cliente, debe de salir de ahí. No se incluyen los centros de distribución dentro de la restricción del flujo debido a que como estos nodos sirven como origen y destino, no se conserva el flujo.

**Restricción de capacidad**

$$\sum_{i \in C} D_i y_{iv} \leq Q_v \quad \forall v \in V \tag{4}$$

Garantiza que la suma de la demanda de los clientes atendidos por un vehículo específico $v$, no superen la capacidad máxima del vehículo

**Restricción de capacidad en Bodegas**

$$\sum_{i \in C} D_i z_{id} \leq K_d \quad \forall d \in CD \tag{5}$$

Garantiza que la suma de la demanda de los clientes suplidos por un centro de distribución $d$, no supere la capacidad máxima de almacenamiento del centro de distribución. Se usó la referencia como inspiración, ya que esta usa la suma de la demanda los clientes que están en la ruta iniciada en el depósito. **[Uniandes]**

**Ventanas de tiempo**

$$8:00 \leq t_{iv} \leq 17:00 \tag{6}$$

Sin embargo la planteamos en minutos para que sea más fácil de modelar

$$480 \leq t_{iv} \leq 1020 \tag{7}$$

Garantiza que el cliente es atendido en una ventana de tiempo de una jornada laboral (8:00am a 5:00pm)

**Tiempo de llegada**

$$t_{jv} \geq t_{iv} + S_i + t_{ij} - M(1 - x_{ijv}) \quad \forall (i,j) \in A, v \in V \tag{8}$$

Define que el tiempo en que el vehículo llega a un nodo $j$ debe ser al menos el tiempo de llegada al nodo $i$, más el tiempo de servicio $S_i$ en $i$, más el tiempo de viaje $t_{ij}$. Big M se utiliza para garantizar que el vehículo termine con el cliente anterior antes de seguir con el siguiente.

**Restricción de autonomía de los vehículos**

$$\sum_{(i,j) \in A} d_{ij} x_{ijv} \leq R_v \quad \forall v \in V \tag{9}$$

Garantiza que la distancia total recorrida por un vehículo $v$ no supere su rango máximo de operación $R_v$.

**Restricción de subtoures**

$$u_{v,i} - u_{v,k} + (n) * x_{i,j,v} \leq n - 1, \quad \forall i,j \in N, \forall v \in V | i \neq j \quad \& \;\; i,j \neq 1 \tag{10}$$

Esta restricción evita que se generen subtoures dentro del recorrido. Esto ayuda a garantizar el correcto funcionamiento del modelo. Para este caso, se implementa con la formulación MTZ.

**Restricción condiciones de inicio**

$$\sum_{d \in CD} \sum_{c \in C} x_{dcv} = 1 \quad \forall v \in V \tag{11}$$

Permite que los vehículos salgan únicamente de centros de distribución.

**Restricción condiciones de final**

$$\sum_{c \in C} \sum_{d \in CD} x_{cdv} = 1 \quad \forall v \in V \tag{12}$$

Permite que los vehículos terminen su recorrido en algún centro de distribución.

**Activación de variables**

$$\sum_{j \in N} x_{djv} = r_{vd} \quad \forall v \in V \quad \forall d \in D \tag{13}$$

Esta primera restricción me indica que si un vehículo sale una única vez de un centro de distribución, la variable $r$ se activa con un valor igual a uno. Ya que como solo sale una vez, solo un término de esa sumatoria se activa. Esto para todos los vehículos y todos los centros de distribución.

$$\sum_{j \in N} x_{cjv} = y_{cv} \quad \forall c \in C \quad \forall v \in V \tag{14}$$

Esta segunda restricción me indica que si un único arco de salida del nodo cliente $c$ se activa con un vehículo $v$ la variable y se activa con valor a uno. y es igual a la sumatoria de los arcos de salida. Ya que solo puede haber uno.

$$y_{cv} \geq r_{vd} \leq z_{cd} \quad \forall c \in C \quad \forall v \in V \quad \forall d \in D \tag{15}$$

Esta última restricción de salida me indica que si la variable $r$ se activa. Todos los clientes $c$ que hayan sido atendidos por el vehículo $v$ que proviene del centro de distribución $d$, han sido abastecidos por el centro de distribución $d$.

# 2 Metodo Implementado

## 2.1 Metaheuristica - GA

Un algoritmo genetico es aquel que genera un numero de individuos (soluciones factibles) y mediante una función evalua que tan aptos son de acuerdo a las restricciones del problema. De ahí los ordena y cruza entre pares de soluciones vecinas en ese orden que la función dispuso, en ocasiones una de las soluciones repentinamente cambia uno de sus valores (mutación). Para así repetir el proceso hasta que se complete un numero de iteraciones deseada (que vendrian siendo las generaciones). Para este algoritmo es fundamental definir: La estructura de cada individuo, La función (llamada fitness), el cruzamiento, la mutación, y para este caso una reparación de soluciones posibles en caso de que la mutación o cruzamiento haga infactible una solución.

### 2.1.1 Individuo

El individuo de este algoritmo genetico es una lista de listas. donde cada lista interior contiene nodos que componen una ruta hecha por un vehiculo. Y la lista externa está compuesta de estas listas, donde cada posición de la lista respresenta a un vehiculo. Por ejemplo si se tienen tres vehiculos con 9 nodos 8 de ellos siendo los clientes y uno de ellos siendo el deposito. Un individuo

factible se muestra a continuación.

$$[[1, 2, 4, 1][1, 9, 5, 7, 1][1, 3, 8, 6, 1]]$$

Todos los nodos fueron visitados por lo que todos los clientes fueron atendidos. La generación de individuos no garantiza que se cumplan las restricciones de carga y rango máximo. Eso se realiza en la función de fitness

### 2.1.2 Fitness

Para el proyecto la función de fitness va a ser calcular el costo total de la operación de cada individuo. Para ello se itera sobre cada una de las rutas del individuo.Se inicializa penalización en 0 y se inicializa costo en 0. Para cada una de estas rutas se realizan tres pasos.

- Calcular la distancia total de la ruta
- Verificar si la restricción de carga se cumple para la ruta. Si no se cumple a penalización se le suma $penalizacion+ = 1e6 * (carga - capmax)$
- Verificar si la restricción de rango de vehiculo se cumple para la ruta. Si no se cumple a penalización se le suma $penalizacion+ = 1e6 * (distancia - rango)$

Finalmente a costo se le suma $distancia * (G_v * P_f + F_t + C_m) + penalizacin$

### 2.1.3 Cruzamiento

El cruzamiento tiene dos individuos padres y cruza segmentos de ruta entre ellos para generar dos nuevos hijos. Para ello escoge un vehiculo aleatorio para cruzar. Se extraen la ruta de ambos padres para ese mismo vehiculo v sin los depositos. Y se elige un punto de quiebre para esas rutas. Se realiza el cruzamiento. En caso de que en el cruzamiento los hijos resulten en hijos no factibles estos serán sometidos al proceso de reparación.

### 2.1.4 Mutacion

Para nuestro individuo la mutación consiste en asignar aleatoriamente un cliente a otro vehiculo. Para esto se escogen dos rutas (vehiculos) de la solución. Y se escoge un cliente de alguna de las dos rutas, el cliente es removido de la ruta original y pasado a la ruta del otro vehiculo.

### 2.1.5 Reparación

Para la operación de cruzamiento, puede ocurrir que la ruta cruzada quede con clientes repetidos o que falten clientes por atender, es por esto que este metodo se ejecuta en la operación de cruzamiento asegurando que todos los individuos que resulten del modelo sean individos que cumplan las primeras dos restricciones enunciadas en el apartado de individuo.

## 2.2 Calibración

El algoritmo genetico tiene un total de 6 parametros que hay que calibrar, estos son: numero generaciones, tamaño de la población, probabilidad mutación, probabilidad cruzamiento, elitismo y tournament size.

Para calibrar y encontrar la mejor configuración para el algoritmo genetico se definieron los siguientes posibles valores para cada uno de estos parámetros.

- **Numero Generaciones** [200,300,400]
- **Tamaño Poblacion** [50,70]
- **Probabilidad mutación** [0.1,0.3]
- **Probabilidad Cruzamiento** [0.6,0.8]
- **Elitismo** True
- **Tournament size** [3,5]

Con estos parametros se hallaron todas las posibles combinaciones entre parámetros. Con todas las 48 combinaciones posibles. Y se escogio la solución que tuviera un mejor costo. Para cada una de las combinaciones se calculo su costo minimo, costo máximo y la desviación estandar de cada uno.

# 3 Resultados Experimentales

El problema tiene un total de 3 casos con 2 tipos diferentes de solución para cada una. Esto genera que haya un total de 6 tipos diferentes de resultados. Para cada caso existe la solución pyomo y la solución metaheuristica.
Para la solución pyomo se presentara los resultados de la solución arrojada por cada uno de los modelos. Mientras que para la solución con metaheuristica se presenta los resultados para cada una de las calibraciones, como lo son los desempeños en tiempo dependiendo de los parámetros dados. Y la solución dada por el algoritmo genético ya calibrado.

## 3.1 Caso 1

### 3.1.1 Pyomo

La implementación para el caso base con Pyomo es igual a la que se presentó dentro del proyecto anterior. A continuación, se detallan los resultados que se obtuvieron de esta en cuanto a las rutas generadas, tiempos de ejecución y demás estadísticas relevantes. El resultado de la implementación de Pyomo se puede ver dentro de la figura 1.

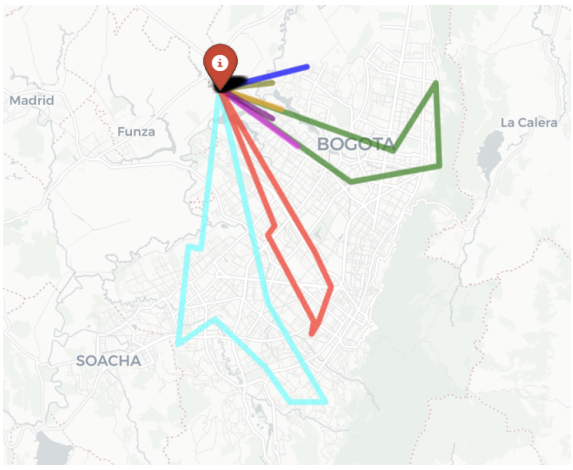


Figura 1: Solución con Pyomo Caso 1

Como se puede ver, la solución a la que se llegó haciendo uso de Pyomo utiliza todos los vehículos y permite obtener un valor dentro de la función objetivo de 4162087.67. Esta implementación es pesada debido a las restricciones del problema, y por ende toma un alto tiempo en correr. Es de mencionar adicionalmente, que para la solución se reduce la tolerancia de optimalidad para que se reduzca el costo computacional asociado.

### 3.1.2 Metaheuristica

En cuanto a la solución con la metaheuristica, las rutas propuestas se pueden ver dentro de la figura 2. Como se puede ver, se utilizan únicamente 4 vehículos para completar la ruta. El costo total fue de 4088117.61, el cual corresponde al costo total de ejecutar todas las rutas.

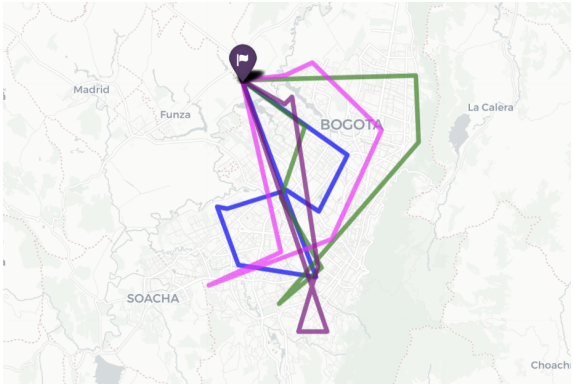


Figura 2: Solución con Algoritmo Genético Caso 1

Dentro de la figura 3 se puede ver una gráfica en donde se muestra el costo por vehículo de la solución encontrada junto con la media como una línea roja dentro de la gráfica y los rangos de la desviación estandar en verde. Como se puede apreciar, la desviación estandar es muy grande debido a la diferencia entre datos para cada vehículo, lo que deja ver que la carga no es balanceada.

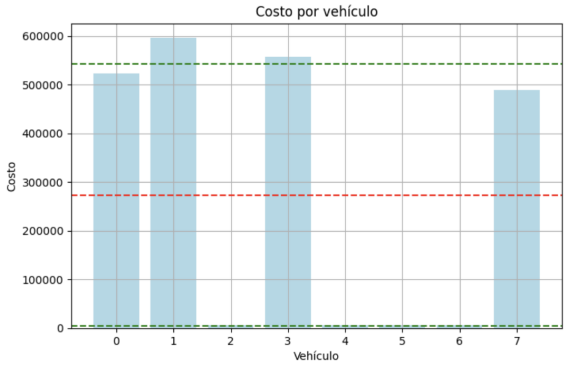


Figura 3: Costo por vehículo Caso 1

Dentro de la figura 4 se puede ver como el algoritmo evolución a través de las generaciones. Para esto es de reslatar que la función de fitness del algoritmo se calcula penalizando a las rutas que no se pueden lograr, y que se selecciona el individuo con el menor valor de fitness. Por lo que la evolución de un decaimiento de fitness significa que se está evolucionando como se espera.

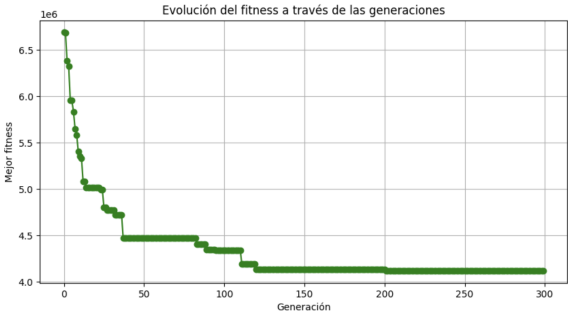


Figura 4: Evolución Caso 1

### 3.1.3 Comparación entre soluciones

Dentro de la tabla 1 se puede ver la comparación entre algunos de los parámetros más relevantes de las 2 implementaciones. Como se puede ver, cada estrategia trae sus ventajas particulares. En este caso en especifico, se puede ver que el tiempo y la memoria que ocupan el uso del algoritmo genético son muy pequeñas en comparación con la implementación de Pyomo. Así mismo, la solución que se obtiene produce un resultado muy favorable y cercano al que obtiene Pyomo. En este caso, donde se tienen pocos clientes y datos, se puede considerar que usar un algoritmo genético puede ser altamente positivo ya que permite obtener soluciones adecuadas en tiempos cortos.

| **Estadística** | **Pyomo** | **Algoritmo Genético** |
|---|---|---|
| Tiempo | 21 min | 1.41s |
| Memoria | 8.71 MiB | 0 MiB |
| Costo Total | 4162087.67 | 4088117.61 |
| # vehículos | 8 | 4 |

Tabela 1: Comparación entre implementación de Pyomo y Metaheuristica

Finalmente, dentro de la figura 5 se puede ver el mapa de las 2 soluciones sobrelapadas. En este, la solución con Pyomo se muestra en verde y la solución con el algoritmo genético se muestra en azul.

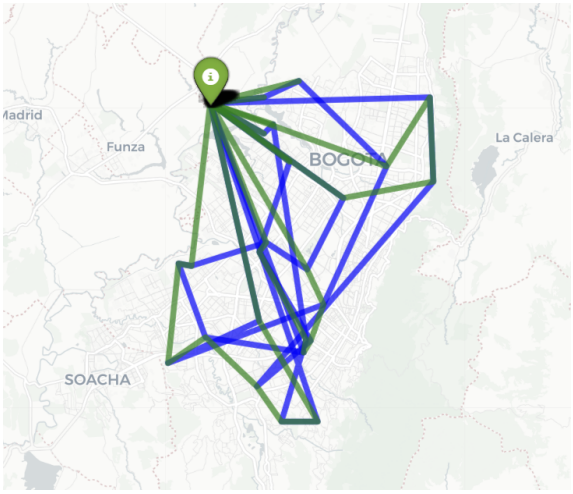


Figura 5: Mapa contrastado Caso 1

## 3.2 Caso 2

Para el caso 2, se utilizaron los datos proporcionados para una implementación mediana dentro de la instancia anterior del proyecto. Estos se adaptaron levemente para poder correr el modelo adecuadamente y se utilizó un único deposito dentro de la implementación.

### 3.2.1 Pyomo

Con estas consideraciones, dentro de la figura 7 se puede ver la solución obtenida para la implementación con Pyomo. Como se puede ver se utilizan 6 vehículos con los que se trazan las rutas para cumplir con las demandas de los clientes.

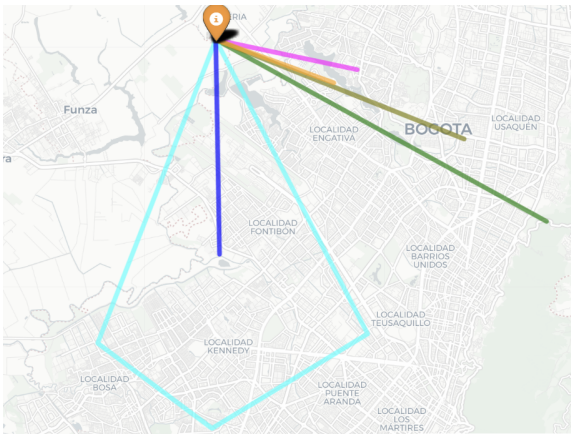


Figura 6: Ruta para caso 2 con Pyomo

### 3.2.2 Metaheuristica

Por el lado de la implementaicón con el algoritmo genético, la solución obtenida se puede ver dentro de la figura 7. En esta solución solo se llegan a utilizar 2 vehículos para generar todo el recorrido. Esto hace notar facilmente, que la solución aproximada no tiene las mejores características puesto qeu aunque llega a completar la ruta, lo hace con una ruta poco óptima pese a la selección de parámetros a partir de la calibración del algoritmo.

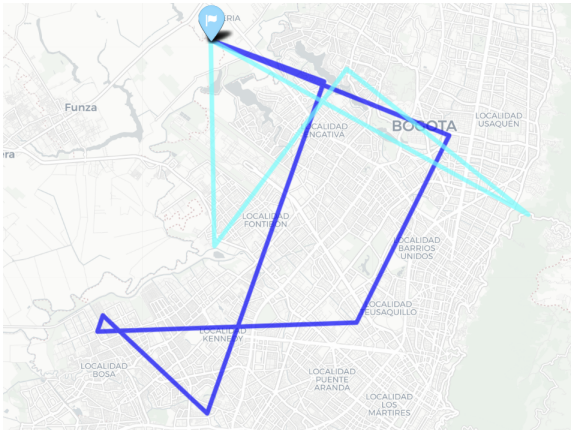


Figura 7: Ruta para caso 2 con Algoritmo Genético

Pese a que el comportamiento no es el mejor, dentro de la figura 8 se puede ver que el algoritmo y su solución evolucionan a medida que aumenta el número de generaciones. Nuevamente, se resalta que debido a la forma en la que se implementó el algoritmo, ver un decrecimiento en el valor de fitness, significa que la solución es menos costosa y por ende, mejor. Aunque se puede ver la evolución del algoritmo, también es de notar que se llega a un punto donde hay pocas mejoras de forma rápida.

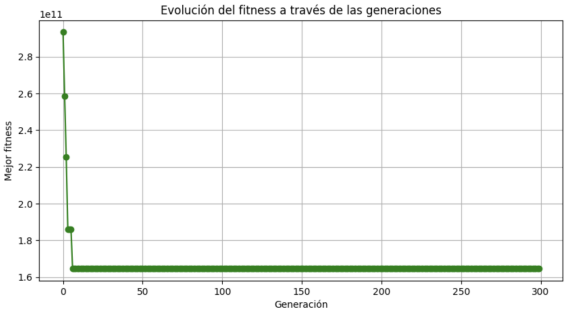


Figura 8: Función de fitness para caso 2

Siguiendo así, dentro de la figura 9 se puede observar una gráfica de costo por vehículo para este caso. Dado a que solo se utilizan 2 vehículos, la desviación estandar es especialmente alta y muestra que la carga no está distribuida.

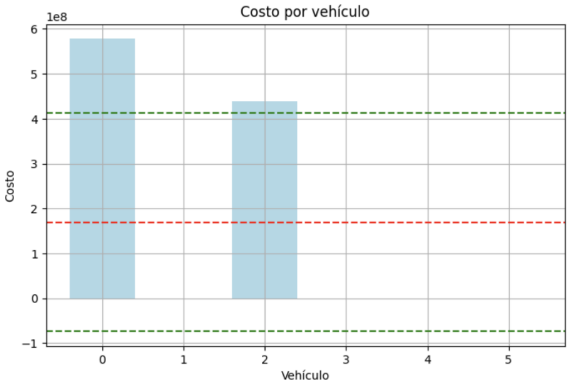


Figura 9: Costo por vehículo Caso 2

### 3.2.3 Comparación entre soluciones

Con todo lo anterior, se puede comparar las 2 soluciones que se obtuvieron. Dentro de la tabla 2 se puede ver el resumen de algunos parámetros sobre los que es posible comparar el desempeño de loa algoritmos. Para este caso, es de notar que el tiempo que toman en ejecutarse es muy corto para ambas implementaciones, lo que no hace que se tenga una preferencia clara. Sin embargo, al evaluar el uso de memoria, se mantiene que la implementación con Pyomo requiere de más recursos. En cuanto al costo total de la solución, se nota una clara diferencia en donde la solución que ofrece Pyomo es mucho mejor. La razón detrás de esto puede ser variada, pero debido al aumento de datos puede que sea más dificil llegar a una solución que ofrezca un buen resultado. Finalmente, se puede ver que la carga entre vehículos se distribuye de mejor forma en la implementación con Pyomo. La conclusión general de este caso, es que se puede llegar a preferir la implementación de Pyomo ya que el tiempo de ejecución es muy bajo y ofrece mejores resultados. Esto es de tener en cuenta al momento que los casos crecen de tamaño, pero no lo suficiente para hacer que Pyomo sea una solución poco viable para obtener un resultado más acercado al óptimo.

| **Estadística** | **Pyomo** | **Algoritmo Genético** |
|---|---|---|
| Tiempo | 0.4 s | 0.74 s |
| Memoria | 5.19 MiB | 0.29 MiB |
| Costo Total | 3124742.66 | 149150472731.77 |
| # vehículos | 6 | 2 |

Tabela 2: Comparación entre implementación de Pyomo y Metaheuristica

## 3.3 Caso 3

### 3.3.1 Pyomo

Para la implementación de Pyomo se realizó exactamente el mismo proceso realizado para los 2 casos anteriores, con la modificación de restricción realizada en el segundo caso para asegurar factibilidad. Sin embargo, se encontró que sin importar los cambios que se realizaban, este generaba una respuesta de No value. Originalmente se contemplaba que era un error de restricciones, pero después de revisar detalladamente, y confirmar que era la misma lógica utilizadas para los otros casos, se concluye que es un error de escalabilidad.

Al estar utilizando el caso base pero con muchos más datos se evidencia que no tiene la capacidad para solucionar el problema. En casos anteriores se habpian utilizado estrategias de clustering para solucionar este problema, sin embargo como ahora se tiene un único depósito, asignar clusters implicaría casi que solucionar el problema de manera empírica que es algo que se quiere evitar en modelamiento.

### 3.3.2 Metaheuristica

Por el lado de la metaheurística se tuvieron resultados gratificantes. Por un lado como se ve en la figura 10 que hay diferentes vehículos llegando a los clientes, donde las turas en su mayoría parecen estar cerradas y ser consecuentes entre la menor distancia recorrida.

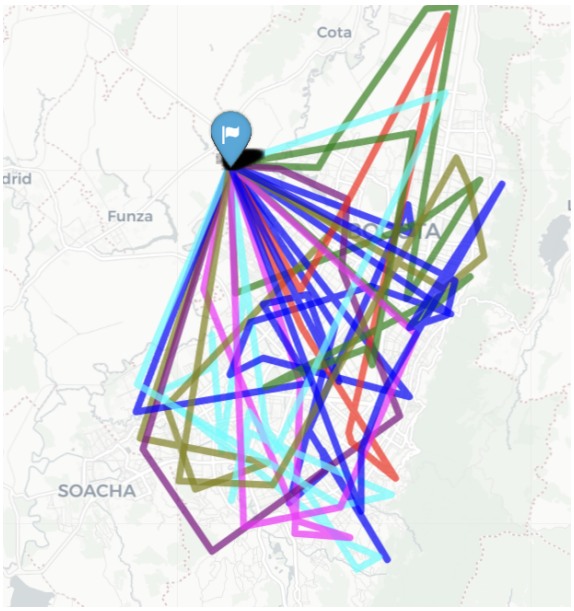


Figura 10: Solución de rutas Algoritmo

Asimismo se ve en la figura 11 que hay una evolución de la función fitness relevante hasta la generación 50. La falta de evolución se puede conectar con un momento donde falte diversidad en los individuos. Sin embargo, al revisar su comportamiento general, al ser completamente proporcionales a la distancia, este punto en la generación 70, todos los mejores individuos generan ese costo, explicando el porque deja de evolucionar desde la generación 70 y se queda estable por las otras generaciones.


Figura 11: Evolución fitness a través de las generaciones

Otra información relevante que se obtuvo de este modelo fue el comportamiento de los vehículos como se muestra en la figura 12 donde se ve el uso de aproximadamente 13 vehículos donde el vehículo 11 es el que más distancia recorre. Esto es evidente porque el costo esta directamente relacionado a la distancia, al necesitar más gasolina, mantenimiento y pago por kilometro recorrido. De esta información también es interesante que no asigna rutas a la mayoría de vehículos. Esto puede ser explicado por el comportamiento habitual de los algoritmos géneticos, donde si no existen casos donde la asignación de estos vehículos es más económica que la ya existente asignación, nunca perseverará ese individuo que propone más vehículos.

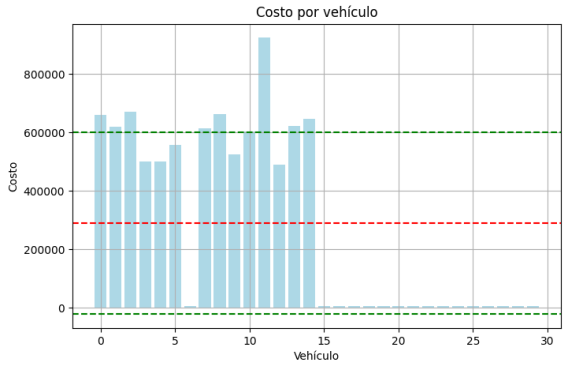


Figura 12: Costo por vehículo

Finalmente se hace su test de configuraciones para revisar cuales eran los mejores parámetros para ejecutar el algoritmo como se ve en la tabla 3

| ID | Generaciones | Población | Mutación | Cruce | Elitismo | Tamaño Torneo | Tiempo (min) |
|---|---|---|---|---|---|---|---|
| 43 | 400 | 70 | 0.1 | 0.8 | True | 5 | 8 11.95 |
| 42 | 400 | 70 | 0.1 | 0.8 | True | 3 | 8 12.26 |
| 41 | 400 | 70 | 0.1 | 0.6 | True | 5 | 8 10.72 |
| 40 | 400 | 70 | 0.1 | 0.6 | True | 3 | 8 8.44 |
| 34 | 400 | 50 | 0.1 | 0.8 | True | 3 | 8 5.17 |

Tabela 3: Comparación de configuraciones del algoritmo genético

En este caso se evidencia un comportamiento muy verosimil entre las primeras 2 configuraciones que contienen el mejor costo general, por lo que se elige la primera por su menor tiempo promedio. Estos parámetros son 400 generaciones, 70 por individuo, una probabilidad de mutación del 0.1, de cruzamiento del 0.8 y un tamaño de torneo de 5 individuos.

#### 3.3.3 Comparación entre soluciones

En este caso no es posible comparar entre los diferentes modelos por las limitaciones de pyomo, sin embargo se estudian los mismos parámetros evaluados en oportunidades anteriores como se ve en la tabla 4

| Estadística | Pyomo | Algoritmo Genético |
|---|---|---|
| Tiempo | - | 9.11 segundos |
| Memoria | - | 352.62 MiB |
| Costo Total | - | 1278329241368.162 |
| # vehículos | - | 13 |

Tabela 4: Comparación entre implementación de Pyomo y Metaheuristica

Esta tabla revela el mayor trade-off de los algoritmos genéticos, sobretodo a tan gran escala, donde se ve que aunque son muy veloces ocupan mucho espacio en memoria. Esto se debe a todas las estructuras que debe construir para llegar a una respuesta factible, lo más cercana a óptima que puede según sus configuraciones. De la misma manera, se ve que el genético puede llegar a una respuesta factible cuando el modelado tradicional, por sus restricciones no es capaz.

# 4 Análisis de Escalabilidad

Al revisar los diferentes resultados, se puede ver que la escalabilidad tiene problemas diferentes según el método que se utilice para solucionar.

## 4.1 Pyomo

Su escalabilidad es lo que define que sea o no sea factible un problema desde el modelado exacto. El único problema que se soluciona de manera completa por modelado es el caso 2. El caso

1 necesita requerir a timeouts y a ser flexibles con su brecha de optimalidad para que genere una respuesta factible. Por lo que también es una aproximación de optimalidad y no completamente óptimo. El caso 3 refleja que existen problemas tan grandes que sin las estrategias correctas (como el clustering) no genera soluciones por su capacidad de procesamiento.

## 4.2 Algoritmos Genéticos

En los algoritmos genéticos se ve mejor respuesta a la escalabilidad, la mayor evidencia siendo que los 3 casos pudieron ser resueltos por el algoritmo genético planteado en el momento inicial. Diferente al caso de pyomo se ve que entre más pequeño el problema (como es la situación con el caso 2) no tiene mucha oportunidada de evolucionar y explorar nuevos individuos que cumplen de mejor manera la tarea en cuestión. En el caso de casos más grandes, lo que pasará es que llegará una generación donde deje de mutar y evolucionar lo suficiente para ver diferencias evidentes entre generaciones. Esto es lo que pudo pasar en el caso 3, que al tener tantos datos, llega un punto donde el mejor puede dejar de explorar otras opciones. En el caso que mejor funciona los algoritmos genéticos son en los problemas de tamaño medio, donde tienen suficiente espacio para mutar, pero no son congelados por la cantidad de mutaciones.

En general se evidencia que para problemas grandes es mejor un algoritmo genético que puede garantizar una respuesta, aunque no óptima, si factible para la mayoría de problemas, en este caso el problema CVRP.

# 5 Discusión

## 5.1 Ventajas y Desventajas

### 5.1.1 Ventajas Pyomo

- Brinda una solución exacta y óptima al problema
- No requiere de una calibración ni búsqueda de parámetros.

### 5.1.2 Ventajas Algoritmo Genetico

- Brinda una solución en un tiempo mucho menor.
- Tiene una complejidad espacial menor al modelo matematico

### 5.1.3 Desventajas Pyomo

- Tiene una complejidad temporal y espacial más alta que el algoritmo genetico
- Para problemas de gran tamaño dada su complejidad temporal y espacial es poco escalable, para problemas medianos o grandes no encuentra una solución de manera rápida
- El modelado del problema puede llegar a ser complejo y dificil de llegar. Hay una complejidad adicional en esto.

### 5.1.4 Desventajas algoritmo Genetico

- No brinda una solución exacta ni óptima al problema, ya que esta es una aproximación del mismo.
- Requiere de una búsqueda de parametros que puede llegar a ser muy extensa.

- Para este ejemplo el algoritmo genetico no tiene en cuenta todas las restricciones. Si bien penaliza por ello, puede que el mejor individuo no cumpla con todas las restricciones dadas.

## 5.2 Recomendaciones

Para problemas que son de un tamaño pequeño o hasta mediano, el modelado matematico y solución por pyomo son la opción a usar. Esto porque como visto en el proyecto, el modelado matematico brinda una solución exacta cuando el tamaño del problema no excede un límite.

Mientras que para problemas de un tamaño mayor la solución aproximada llega a ser una mejor opción. Ya que si no se posee el solucionador comercial Gurobi. Los solucionadores gratuitos no llegan a una solución de estos problemas más extensos. Es por esto que se opta por una solución aproximada como el algoritmo genetico. Ya que este al menos llega a dar con una solución aproximada, que si bien no es la solución óptima puede estar cercana a ella. Y para el problema brinda una solución logística.

## 5.3 Lecciones Aprendidas

- El modelo matematico brinda una solución exacta, optima y confiable. Sin embargo hay casos en que aplicar este algoritmo resulta dificil. Ya que el modelado puede resultar confuso, trasladar un problema de la naturaleza hacia un modelo matematico exacto optimizable no siempre es una tarea fácil. cuando esto sucede puede que una solución más "natural"sea conveniente.

- La solución metaheuristica es útil cuando el problema presentado por la naturaleza resulta dificil de modelar matematicamente. Adicional a esa dificultad, el modelo tarda mucho en resolver o no resuelve. En estos casos es mejor optar por una solución aproximada inspirada en la naturaleza. Como es la metaheuristica. El comportamiento de la genetica y como esta se adapta al entorno para sobrevivir es un comportamiento ideal y facilmente modelable para solucionar estos problemas extensos.

- Identificar cuando es factible aplicar una solución u otra es una lección aprendida de este proyecto. Aprender que para un mismo problema no solo hay diversas maneras de modelarlo matematicamente. Pero diferentes enfoques que pueden brindar soluciones para cumplir parcial o totalmente los requerimientos del problema. Saber las ventajas y desventajas de cada enfoque y saber utilizarlas para obtener el mayor beneficio o la mejor satisfacción al extraer una solución del problema.

# 6 Conclusiones

El objetivo principal de la practica era implemntar una solución via modelo matematico (pyomo) y una solución via metaheuristica (algoritmo genetico) y compararlas en la mayoria de aspectos posibles: Escalabilidad, complejidades (temporal, teorica, espacial, practica). Para así determinar las ventajas y desventajas de las mismas. Y con esto analizar los escenarios en donde una solución se desempeña mejor que la otra. Y con esto concluir cuando optar por una solución exacta (como lo es el modelo) o una solución aproximada (como lo es la metaheuristica).

Respecto a este gran objetivo se puede concluir que se cumplio a totalidad. Ya que en todas las etapas del proyecto se estuvieron comparando las dos soluciones. Empezando por el planteamento

inicial de ambos en las secciones uno y dos. Obteniendo unos resultados como se evidencian tanto en el codigo anexo como en la sección 3. Realizando un análisis de escalabilidad entre ambas soluciones en la sección 4. Para así extraer las ventajas y desventajas de cada una de las implmentaciones en la primera parte de la sección 5. Y con esto tener un critero de cuando utilizar cada uno de los algoritmos. Se puede concluir que se evidenció con exito la diferencia entre lo exacto y lo aproximado y cuales son los trade-off que cada enfoque implica.