ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ciencias Naturales y Matemáticas

Asignación automatizada de rutas a nivel nacional para una empresa del sector público

Previo a la obtención del Título de:

Matemático

Presentado por:

Alexsander Israel Játiva Macías

Frank Leonardo Meza Moina

GUAYAQUIL - ECUADOR

Año: 2024

DEDICATORIA

A todos los reales.

Alexsander Játiva M.

DEDICATORIA

Dedico esta tesis a mi mamá Teresa, a mi abuela Inés, a mis tíos Carlos y Danny junto a sus esposas Verónica y Alexandra.

Frank Meza M.

AGRADECIMIENTOS

Quiero expresar mi gratitud a nuestro tutor Ph.D. Xavier Cabezas por su retroalimentación y apoyo a lo largo del proyecto, del mismo modo que a la matemática Ph.D. Elimar Marchan por todos los consejos a lo largo de este proyecto.

Alexsander Játiva M.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a mi tutor, el Dr. Xavier, por su invaluable apoyo y orientación a lo largo de todo este proyecto. Asimismo, agradezco a la Dra. Elimar por sus consejos y su constante retroalimentación, los cuales fueron fundamentales para el desarrollo de esta tesis. A mis amigos, compañeros de la carrera, quienes han sido una fuente de motivación y apoyo, les agradezco de corazón; sin ustedes, probablemente habría tomado un camino distinto, como el de la Estadística. Extiendo mi gratitud a mis amigos de colegio: Jesús, Bruno, Luis, Melany y Jordana, quienes han estado a mi lado a lo largo de estos años, brindándome su amistad y consejos en cada etapa de mi vida. Finalmente, quiero agradecer a Shanesia, por su apoyo incondicional en momentos difíciles y por ayudarme de diversas maneras en una etapa complicada de mi vida.

Frank Meza M.

DECLARACIÓN EXPRESA

"Los derechos de titularidad y explotación, nos corresponde conforme al reglamento de propiedad intelectual de la institución; *Alexsander Israel Játiva Macías, Frank Leonardo Meza Moina,* y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Alexsander Játiva M.

Frank Meza M.

EVALUADORES

Firmado electrónicamente por:
LUZ ELIMAR MARCHAN MENDOZA

Luz Elimar Marchan Mendoza

PROFESOR DE LA MATERIA

Firmado electrónicamente por:
JOSE XAVIER CABEZAS GARCIA

José Xavier Cabezas García

TUTOR

RESUMEN

La asignación de redes de transporte es crucial para las empresas, pues una pobre planificación de las rutas incurre en un aumento en los gastos operativos. De esta forma, las empresas tienen la necesidad de herramientas que les permitan realizar dichas asignaciones de forma eficiente. En particular, una empresa pública del Ecuador asigna sus operaciones de transporte de forma manual, tardándose un promedio de dos días por implementación. En este trabajo se implementó un modelo de matemático orientado a asignar rutas con el objetivo de reducir las distancias recorridas. Para lograrlo, se usaron técnicas de clustering, seguido de la implementación de un problema de ruteo de vehículos que permite determinar las rutas óptimas. Usando estas técnicas se logró realizar la asignación de rutas en poco menos de dos horas, representando una reducción del 88.73% del tiempo con respecto a la implementación manual. Con este enfoque se busca no solo optimizar la distancia por el personal operativo, sino también mejorar la eficiencia operativa y reducir el impacto ambiental asociado al uso de la flota vehicular. En conclusión, se espera que esta implementación proporcione una mayor precisión en la planificación y ejecución de las rutas, facilitando una gestión más eficiente de los recursos disponibles y contribuyendo a la sostenibilidad operativa de la empresa.

Palabras Clave: Optimización de rutas, clustering, ruteo de vehículos, planificación logística.

ABSTRACT

The allocation of transportation networks is crucial for companies, as poor route planning leads to increased operating costs. Thus, companies need tools that allow them to make such assignments efficiently. In particular, a public company in Ecuador assigns its transportation operations manually, taking an average of two days per implementation. In this work, a mathematical model was implemented to assign routes with the objective of reducing the distances traveled. To achieve this, clustering techniques were used, followed by the implementation of a vehicle routing problem to determine the optimal routes. Using these techniques, route assignment was achieved in just under two hours, representing a reduction of 88.73% of the time compared to manual implementation. This approach seeks not only to optimize distance for operating personnel, but also to improve operational efficiency and reduce the environmental impact associated with the use of the vehicle fleet. In conclusion, this implementation is expected to provide greater precision in route planning and execution, facilitating more efficient management of available resources and contributing to the company's operational sustainability.

Translated with DeepL.com (free version)

Keywords: *Route optimization, clustering, vehicle routing, logistical planning.*

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

Muchas situaciones pueden ser modeladas a través de la teoría de Grafos. En particular, asignando valores o pesos a los arcos del grafo, se pueden abordar una gran variedad de problemas de optimización. Los problemas de optimización son frecuentes en la industria, siendo uno de los principales la determinación de la forma más óptima de disponer una flota de k vehículos para atender a m clientes, minimizando la distancia recorrida por la flota. Este tipo de problemas se conocen como problemas de ruteo vehicular, o VRP por sus siglas en inglés.

Formalmente, el VRP es un problema de optimización en grafos, clasificado como NP-Duro e incluye como caso particular el problema del agente viajero, o TSP por sus siglas en inglés (Toth & Vigo, 2012).

El VRP ha sido ampliamente estudiado, principalmente porque una implementación correcta del VRP o sus variantes permite minimizar los gastos operativos de transporte en las empresas. En este documento, se revisan algunas variantes del VRP aplicadas a una empresa pública que necesita realizar operaciones de ruteo a nivel nacional.

1.1 Descripción del problema

En una empresa pública de Ecuador, se realiza un programa de visita a clientes a nivel nacional. El período de visitas dura 20 días, dividido en cuatro bloques de cinco días de trabajo.

Para la tarea en cuestión, se cuenta con siete vehículos. Cada equipo de trabajo está compuesto por un vehículo para el transporte, un chofer, un jefe de vehículo y tres empleados, siendo estos últimos los encargados de realizar las actividades correspondientes.

Cada mes, se deben visitar 3200 manzanas del país. Las manzanas están agrupadas en conjuntos de entre cinco y siete elementos llamados conglomerados, los cuales se asignan de forma aleatoria. Los empleados deben completar todas las actividades en el período de 20 días.

La empresa ha estado realizando este proceso durante varios años. La asignación de rutas para cada vehículo se realiza de forma manual, lo que no solo consume una cantidad considerable de tiempo (entre tres y cuatro días para la planificación), sino que también es extremadamente ineficaz: No se consideran variables cruciales como la congestión vehicular y la distancia entre conglomerados, y se omite la posibilidad de que en las manzanas seleccionadas haya un bajo índice de respuestas.

Además, al basarse únicamente en la experiencia, las planificaciones no presentan métricas que permitan validar su efectividad ni un enfoque matemático que permita minimizar las distancias recorridas. Este método de asignación de rutas no validado puede conllevar un gasto de tiempo en congestiones de tráfico, tráfico en zonas urbanas, un aumento en el gasto de recursos para la flota y mayor contaminación ambiental debido al incremento en el número de kilómetros recorridos.

1.2 Justificación del problema

Este proyecto es relevante por varias razones. Implementar un modelo matemático que genere una asignación automática de rutas, enfocado en la optimización de la distancia recorrida, conducirá a una mayor eficiencia en las actividades de visitas. Esto permitirá reducir los tiempos

de viaje, minimizar los costos operativos y maximizar la productividad durante los períodos de recolección. Además, gracias a la minimización de la distancia recorrida, se reduce el consumo de gasolina, lo que disminuye la huella de carbono y contribuye a la sostenibilidad de las empresas. Esto es especialmente relevante ya que, según Ritchie (2020), cerca del 13% de los gases de efecto invernadero provienen de los automóviles.

En esta investigación, se propone un modelo matemático que implementa formulaciones de problemas de ruteo vehicular para lograr una asignación automatizada. Esta formulación tendrá un enfoque matemático orientado a minimizar los kilómetros recorridos por los vehículos de trabajadores.

1.3 Objetivos

1.3.1 Objetivo General

Implementar un modelo matemático para la optimización de rutas en una institución pública, que integre soluciones de problemas de ruteo vehicular y permita mejorar la eficiencia en la visita a clientes considerando factores como el tráfico, la disponibilidad de los clientes y los recursos limitados.

1.3.2 Objetivos Específicos

A continuación se detallan los diferentes objetivos específicos del trabajo.

1. Implementar un modelo matemático que optimice las rutas, minimizando el tiempo total de desplazamiento y maximizando la eficiencia operativa;
2. Evaluar la efectividad del modelo matemático propuesto con las soluciones actuales

utilizadas en la institución, utilizando métricas de desempeño como el tiempo de recorrido y el número visitas completadas;

3. Analizar los resultados obtenidos del modelo matemático para identificar áreas de mejora y proponer recomendaciones para la implementación del modelo en la planificación operativa de la institución.

1.4 Marco teórico

1.4.1 Introducción

Dantzig & Ramser (1959) definieron el problema de despacho de vehículos (Truck Dispatching problem), vagamente este es un problema de optimización que consiste en determinar la mejor forma de desplegar una flota de vehículos iguales de tal forma que se visiten un cierto número de clientes y se minimice la distancia recorrida; años más tarde en un paper publicado por Clarke y Wrigth se presentó una generalización a través de la teoría de programación lineal (Braekers, Ramaekers, & Van Nieuwenhuyse, 2016). Este documento es reconocido en el área de investigación de operaciones, pues dio lugar a lo que se conoce actualmente como problemas de ruteo vehicular, o VRP por sus siglas en inglés, (para una definición formal consúltese la sección 1.4.2).

El VRP es uno de los problemas más estudiados en el área de logística e investigación de operaciones, como se mencionó en la introducción [1], es conocido que el VRP es un problema NP-Duro, lo cual quiere decir que para un número elevado de ciudades no es posible usar métodos que encuentren una solución exacta, sin embargo, existen heurísticas que dan buenos resultados para problemas con un gran número de instancias.

La versión del VRP presentada por Clarke y Wright es altamente simplificada, actualmente existen una gran cantidad de variaciones del VRP para acoplarse más a la realidad, (véase Toth & Vigo (2012) y Drexl (2012)). En este documento, se describirán el VRP, una de ellas siendo la versión básica y la otra una extensión que incluye ventanas de tiempo. Con la primera formularemos el problema en la sección [1.1] y la se segunda se presenta a modo de mejora, provisto de que la empresa tenga los datos necesarios. [1]

1.4.2 Problemas de Ruteo

Problema de ruteo Vehicular clásico (CVRP)

AIMMS (2020a). Sea $G = (V, A)$ un grafo, con nodos $V = \{0, 1, \ldots, n\}$, donde los nodos $1, \ldots, n$ representan clientes y el nodo 0 representa el depósito, es decir, el lugar desde donde salen los vehículos. Cada cliente tiene una demanda d_i. Sea A el conjunto de arcos o aristas. Para cada arco (i, j) se tiene un costo no negativo $c_{ij} \geqslant 0$, en general, este costo suele provenir de una distancia. Sea m el número de vehículos todos con capacidad D. El problema consiste en encontrar un conjunto de rutas que minimicen el costo de cada vehículo, y que cumplan las siguientes restricciones:

- Cada cliente de $V - \{0\}$ debe ser visitado exactamente una vez.
- Todos los vehículos deben partir y regresar al depósito.[2]

Además, para que el problema tenga solución, se debe pedir que la demanda no supere la capacidad de cada vehículo. Bajo estas condiciones, formalmente el CVRP es formulado bajo el

[1] Actualmente, la empresa no posee ventanas de tiempo sobre sus clientes.

[2] En esta formulación los vehículos deben regresar al depósito pero existen versiones que quitan esta restricción.

siguiente problema de optimización:

$$\text{Minimizar: } \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} c_{ij}x_{ijk} \tag{1.1}$$

$$\text{Sujeto a: } \sum_{i=0}^{n} x_{ijk} = \sum_{i=0}^{n} x_{jik} \tag{1.2}$$

$$\sum_{k=1}^{m}\sum_{i=0}^{n} x_{ijk} = 1,\ j \in \mathbb{N}_n \tag{1.3}$$

$$\sum_{j=1}^{n} x_{0jk} = 1,\ k \in \mathbb{N}_m \tag{1.4}$$

$$\sum_{i=0}^{n}\sum_{j=1}^{n} d_j x_{ijk} \leqslant D \tag{1.5}$$

$$\sum_{i\in S, j\notin S} x_{ijk} \geqslant 2,\ S \subset V - \{0\}, 2 \leqslant |S| \leqslant n-2 \tag{1.6}$$

Donde, $x_{ijk} \in \{0,1\}$, $k \in \mathbb{N}_m$ y $i, j \in \mathbb{N}_n$. Aquí $x_{ijk} = 1$ representa que el vehículo k va del cliente i al cliente j, y $x_{ijk} = 0$ representa que el vehículo k no hace dicho recorrido, y la función (1.1) representa la función de costo. Las ecuaciones (1.2) a (1.5) representan las restricciones del problema, a saber:

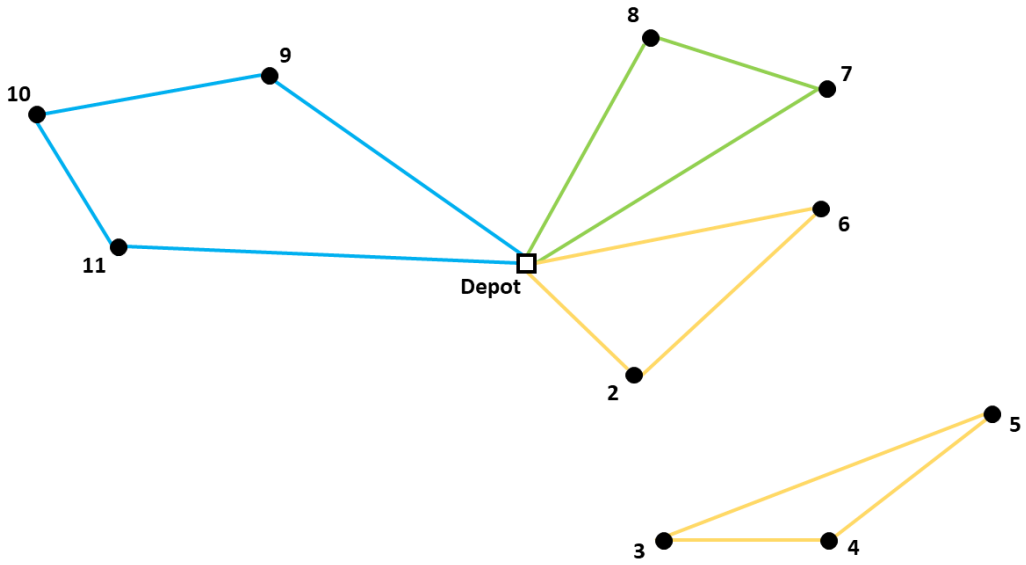
(1.2) Garantiza que cada vehículo sale desde cada cliente que visita.

(1.3) Garantiza que cada vehículo sale del depósito. (No hay vehículos sin utilizar).

(1.4) Garantiza que el costo de demanda de cada ruta no supere a la capacidad del vehículo.

La restricción (1.6) es interesante, pues las primeras cinco son obtenidas directamente del problema, sin embargo, solo con estas restricciones puede ocurrir que se obtenga una solución que presente subciclos. Por ejemplo, en la Figura 1.1 se aprecia una solución de un problema de ruteo con diez clientes y tres vehículos, denotados por el color verde, amarillo y azul, que presenta una ruta con un subciclo.

Figura 1.1.
Ejemplo solución con subciclo



Fuente: (AIMMS, 2020b)

La restricción (1.6), conocida como Formulación de Dantzig-Fulkerson-Johnson, garantiza que la solución no presenta subciclos (AIMMS, 2020b).

Problema de ruteo vehicular con ventanas de tiempo. VRPTW

En muchas aplicaciones prácticas, se requiere que los vehículos visiten a los clientes en una ventana de tiempo determinada. Al agregar esta restricción se obtiene una nueva versión del VRP conocida como *problema de ruteo vehicular con ventanas de tiempo*, o VRPTW por sus siglas en inglés. La formulación del VRPTW es una extensión del CVRP con la diferencia de que se deben añadir las restricciones de las ventanas de tiempo, además, en este caso no hace falta usar la formulación de Dantzing para eliminar subciclos, pues las nuevas restricciones de tiempo logran eliminar los subciclos.

Para cada cliente i se tiene un intervalo $[a_i, b_i]$ llamado la ventana de tiempo del cliente i. Un vehículo debe llegar al cliente i en un tiempo $t \in [a_i, b_i]$. Sea t_{ij} el tiempo que lleva llegar desde el cliente i al cliente j. En general, en esta variable se incluye el tiempo que se demorará en el cliente i. $s_i \in [a_i, b_i]$ el tiempo en que el cliente i empieza a ser atendido. Bajo la misma formulación del CVRP se deben añadir las siguientes restricciones:

- **Duración de las rutas**

$$s_i + t_{ij} - M(1 - x_{ijk}) \leqslant s_j, \ \forall i \in V, j \in V - \{0\}, k \in \mathbb{N}_m$$

- **El tiempo a ser servido debe estar dentro de la ventana de tiempo**

$$a_i \leqslant s_i \leqslant b_i, \ \forall i \in V,$$

donde $M = \max_{ij}\{b_i + t_{ij} - a_i\}$. Dado que el tiempo siempre va incrementando, se garantiza que no existan subciclos. (AIMMS, 2020c).

Problema del agente viajante (TSP)

El problema del agente viajante, o TSP por sus siglas en inglés, es un problema de optimización combinatoria. El problema consiste en lo siguiente, dado un conjunto de N ciudades determinar el camino de menor distancia que debe recorrer un agente, de tal forma que cada ciudad sea visitada sola una vez. El TSP es probablemente el problema más famoso de optimización, actualmente en Google Scholar (accedido en Agosto del 2024), existen cerca de 800,000 artículos que hacen mención a este problema. Note que todo TSP puede ser visto como un CVRP, basta con tomar a una ciudad cualquiera del conjunto como el depósito (pues la longitud del trayecto no va a cambiar), con una flota de un solo vehículo con capacidad $D \geqslant N$ y

cada cliente con demanda 1, entonces tenemos un TSP. Por lo tanto, en este documento cuando se refiere a soluciones del TSP se usarán los mismos métodos descritos para CVRP.

Dado que el TSP y el VRP son problemas NP-Duros, para una gran cantidad de puntos el problema se vuelve intratable, es decir, no es posible resolverlo en un tiempo razonable. De acuerdo con Toth & Vigo (2012) uno de los mejores algoritmos actuales fue propuesto por Fukasawa y colaboradores en un artículo del 2006, denominado Rama-Corte-Precio o BCP por sus siglas en inglés. El método BCP obtiene la solución en tiempo pseudo-polinomial, específicamente, se da la solución en $O(Cn^2)$ donde n es el número de vehículos y C la capacidad de cada vehículo. En ese mismo artículo, se determinó que para un problema con 121 ciudades y 7 vehículos, el algoritmo tarda 25678 segundos, lo cual corresponde a poco más de 7 horas (Fukasawa et al., 2006). Por motivos que serán explicados en la sección (2.1) esta comparativa es la que más se acerca a las realidad de los datos del problema a resolver, por tanto se debe usar una heurística para solucionar el problema en un tiempo razonable, en este trabajo se usará el método de Búsqueda Genética Híbrida (HGS).

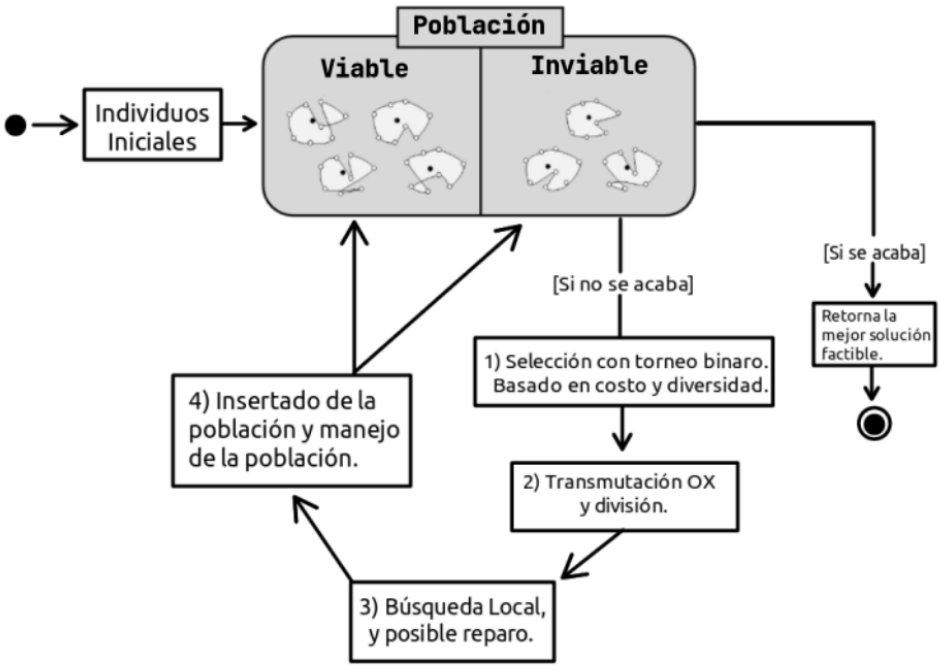
Búsqueda genética híbrida

La búsqueda genética híbrida es una heurística basada en los procesos genéticos de los seres vivos, el esquema general del método para problemas de ruteo es el siguiente: Se parte de un conjunto rutas llamadas individuos, con estos individuos se genera una población, dado que las rutas son elegidas sin ningún tipo de característica se pueden tener soluciones infactibles, es decir, con subciclos, por esto se divide la población en dos; una población de soluciones factibles y otra de soluciones infactibles. Después de determinar las poblaciones el algoritmo genera nuevas soluciones de la siguiente forma:

1. Se selecciona dos progenitores.
2. Se recombinan los progenitores para obtener una nueva solución.
3. Se mejora esta solución a través de una búsqueda local.
4. Se inserta la nueva solución en la población.

Este proceso se repite hasta un cierto criterio de parada, normalmente se suelen dar tres tipos de criterios de parada: El número de iteraciones (IT_{max}), el tiempo en segundos (T_{max}), o un máximo de iteraciones consecutivas sin mejoría en la solución (N_{IT}). El esquema presentado se puede apreciar en la siguiente imagen

Figura 1.2.
Proceso del algoritmo genético híbrido



Fuente : (Vidal, 2022)

Por supuesto, cada paso mostrado en la imagen anterior viene justificado a través de métodos matemáticos.

Selección de Progenitores: Para seleccionar los progenitores, el algoritmo realiza un torneo binario, esto es; se toman dos rutas de manera aleatoria a través de una distribución de probabilidad uniforme, luego estas rutas son comparadas, seleccionado como progenitor aquella con mayor costo, esto se hace dos veces para conseguir ambos progenitores. Es importante mencionar que cuando se refiere a costo de la ruta o solución, no se está hablando de la distancia recorrida en dicha ruta, en este caso el costo viene dado por

$$f_P(S) = f_P^{\varphi}(S) + \Big(1 - \frac{n^{\text{ELITE}}}{|P|}\Big) f_P^{DIV}(S) \qquad (*)$$

Donde $f_P^{\varphi}(S)$ es el rango de una solución (individuo) en una subpoblación con m individuos, con respecto al costo de penalización φ y $|P|$ es el tamaño de la población. El costo de penalización φ se define de la siguiente manera. Llámese subruta a cada una de las rutas que debe seguir cada vehículo en una solución. Se denota el conjunto $R(s)$ como el conjunto de todas las subrutas de una ruta s. Para cada subruta r, el costo de penalización viene dado por

$$\varphi(r) := C(r) + w^Q \,\text{max}\{0, q(r) - Q\} + w^T \,\text{max}\{0, \tau(r) - T\}$$

Donde, $C(r)$ representa la distancia de la subruta r, w^Q es el coeficiente de penalización de la capacidad, $q(r)$ la capacidad de la subruta, Q es la capacidad del problema, w^T es el coeficiente de penalización de la duración, $\tau(r)$ es la duración y T es la duración máxima de r, respectivamente. Es importante mencionar que en implementaciones donde no se considere el tiempo, se puede tomar $w^T = 0$. En particular, para la implementación que se propone no se considera el tiempo, por lo cual se puede asumir que $w^T = 0$. Al inicio del algoritmo, se toma

$w^Q = \overline{c}/\overline{q}$ donde $\overline{c}$ es la distancia promedio de entre clientes y $\overline{q}$ es la demanda promedio. Durante la ejecución del algoritmo, w^Q es ajustado para mejorar a los individuos aptos. El ajustamiento se produce a través de la siguiente regla. Sea ξ^{REF} el porcentaje de soluciones viables en 100 soluciones, y ξ^Q el porcentaje de soluciones viables con respecto a la capacidad, esto es, aquellas que no excedan la capacidad, entonces

1. Si $\xi^Q \leqslant \xi^{REF} - 0.05$ entonces $w^Q = 1.2w^Q$;
2. si $\xi^Q \geqslant \xi^{REF} + 0.05$ entonces $w^Q = 0.85w^Q$;

Nótese que la evolución de este coeficiente impacta directamente en la elección de individuos. Por otro lado, $f_P^{DIV}(S)$ es la contribución a la diversidad, de las n_{close} soluciones más cercanas agrupadas en el conjunto $\mathcal{N}$, este valor es calculado a través de la siguiente ecuación (Vidal, Crainic, Gendreau, Lahrichi, & Rei, 2012)

$$f_P^{DIV}(S) = \frac{1}{n_{close}} \sum_{P \in \mathcal{N}} \delta^H(P, S)$$

donde

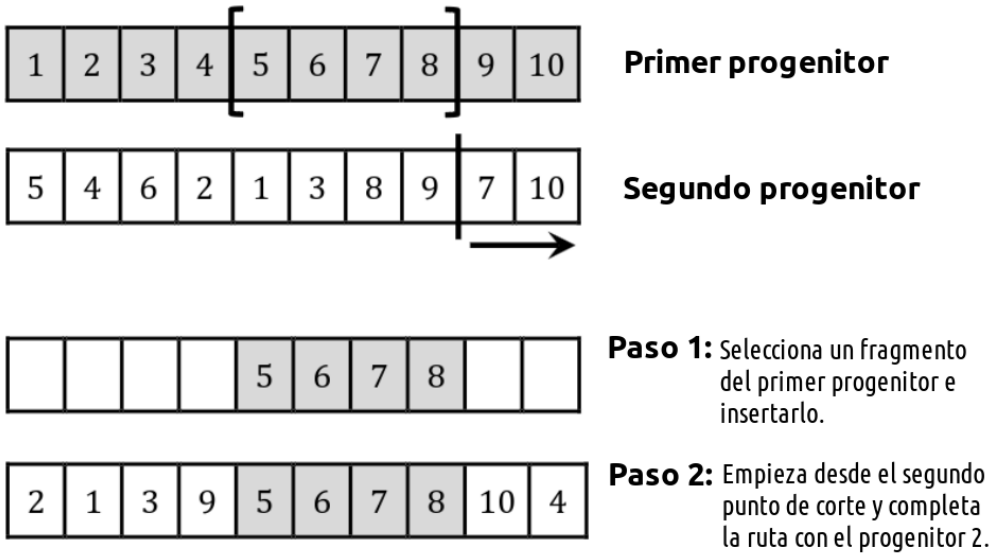
$$\delta^H(P_1, P_2) = \frac{1}{2n} \sum_{i=1}^{n} \mathbf{1}(\pi_i(P_1) \neq \pi_i(P_2))$$

En esta última ecuación, P_1 y P_2 son dos individuos (rutas), **1** da el valor de 1 si la condición es verdadera, o 0 caso contrario. Y la función π_i determina el patrón de cromosomas del cliente i, que en el contexto del CVRP se refiere al cliente anterior y posterior al cliente i en una ruta, y $\mathcal{N}$ es el conjunto de los n_{close} individuos más cercanos, es decir aquellos individuos cuyo patrón genético es más similar al de S. Es importante mencionar que en la versión propuesta por (Vidal, Crainic, Gendreau, Lahrichi, & Rei, 2012) la función δ^H incluye el término $\mathbf{1}(\delta_i(P_1) \neq \delta_i(P_2))$ que da 1 si los clientes tienen depósitos distintos y 0 caso contrario, dado que en el caso presentado en este

trabajo solo contiene un depósito podemos omitir este término. La ecuación (∗) tiene un mayor peso que otras implementaciones de algoritmos genéticos, esto se hace con el fin de garantizar una mayor calidad en la solución final.

Transmutación OX y división: El algoritmo usa una transmutación OX para crear un nuevo individuo a través de los progenitores, esto se hace de la siguiente manera: Se toma un fragmento aleatorio del primer progenitor y se lo reemplaza en la misma posición del segundo progenitor, y luego se completa a través del segundo progenitor una ruta. La aplicación de este procedimiento pude ser realizado en tiempo lineal.

Figura 1.3.
Transmutación OX



Adoptado de: (Vidal, 2022)

Es importante mencionar que esta representación omite la visita al depósito, pero existen algoritmos altamente eficientes que son capaces de reinsertar los depósitos para conseguir soluciones completas del CVRP.

Búsqueda local y posible reparo: Una vez que se obtiene el nuevo individuo a través de transmutación y división, se hace una búsqueda local, esta búsqueda es realizada a través del método SWAP*. Este consiste en cambiar dos clientes v y v' de rutas distintas r y r', sin insertarlos en el mismo lugar, es decir, el cliente v no reemplaza al cliente v' y viceversa. Como la búsqueda se hace en toda la población, puede darse el caso de que una solución sea infactible después de aplicar la búsqueda local. Para solucionar esto se aplica una operación de reparación con un 50% de probabilidad. Esto vuelve a ejecutar la búsqueda local con mayor penalidad, para tratar de conseguir una solución factible, si no se puede, se hace un manejo de la población. Este algoritmo de SWAP* genera mejorías con respecto a otros métodos de búsqueda genética, por ejemplo, el propuesto por (Vidal, Crainic, Gendreau, Lahrichi, & Rei, 2012).

Manejo de la población: En la ejecución de la búsqueda genética se generan dos poblaciones; factible e infactible. Después de realizar la búsqueda local podemos tener un individuo que corresponde a una de las dos poblaciones, este es introducido en la población correspondiente. Cada población contiene entre μ y $\lambda + \mu$ soluciones, donde μ y λ son enteros positivos dados. El parámetro μ determina la mínimo tamaño de la población y el valor λ es el tamaño de la generación. Al iniciar el algoritmo, se toman una población inicial de 4μ individuos, con esta población inicial y la búsqueda local se van creando las poblaciones factible e infactible, cuando una población llega a tener $\mu + \lambda$ individuos, se empieza un proceso para eliminar λ soluciones, este procedimiento se hace bajo dos métricas:

1. Se eliminan las soluciones iguales. Y de lo contrario;

2. Se eliminan las soluciones con menor costo de acuerdo con la ecuación $(*)$.

Por tanto, el algoritmo quedaría de la siguiente forma

Algoritmo 1.1. Búsqueda Genética Híbrida

```
Iniciar la población con soluciones aleatorias mejoradas por búsqueda local;
while número de iteraciones sin mejoría < IT_max y tiempo de ejecución < T_max do
    Seleccionar soluciones progenitoras P_1 y P_2;
    Aplicar transmutación a P_1 y P_2 para generar descendencia C;
    Mejorar la descendencia C a través de búsqueda local;
    Insertar C en la respectiva población;
    if C es infactible then
        Con un 50%, reparar C a través de búsqueda local e insertar en la respectiva población;
    end
    if Máximo tamaño de la población se alcanza then
        Seleccionar los supervivientes;
    end
    Ajustar la penalidad de los coeficientes para las soluciones infactibles;
end
Dar la mejor solución factible;
```

En pruebas realizadas con este algoritmo para instancias de VRP se tiene que la solución suele estar en un margen de error del 11% con respecto a la solución. (Vidal, 2022)

1.4.3 Algunas Técnicas de Clustering

K-means Clustering

El algoritmo K-means es una técnica que consiste en dividir un conjunto de datos en K grupos, minimizando la varianza dentro de cada grupo. Adaptándolo al VRP, K-means es utilizado para agrupar los clientes en clústeres que pueden ser atendidos por rutas individuales.

Dado un conjunto de observaciones $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, donde cada observación es un vector real de d dimensiones, el clustering K-means tiene como objetivo dividir las n observaciones en k conjuntos $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ para minimizar la suma de cuadrados intra-clúster(varianza). El objetivo es encontrar:

$$\arg\min_{\mathcal{C}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathcal{C}_i} \|\mathbf{x} - \mu_i\|^2,$$

donde μ_i es la media (también llamada centroide) de los puntos en $\mathcal{C}_i$, es decir:

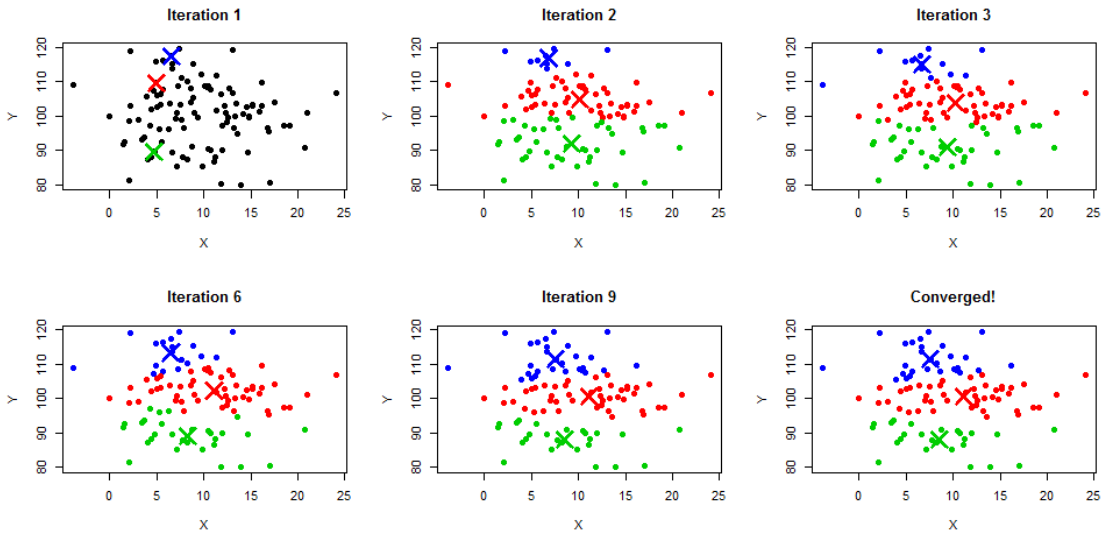
$$\mu_i = \frac{1}{|\mathcal{C}_i|} \sum_{\mathbf{x} \in \mathcal{C}_i} \mathbf{x},$$

$|\mathcal{C}_i|$ es el tamaño de $\mathcal{C}_i$, y $\|\cdot\|$ es la norma euclídea, aunque puede cambiar la formulación con otra norma.(Kriegel, Schubert, & Zimek, 2017)

La formulación anterior es una versión más utilizada y expuesta en la actualidad, pero también se puede definir de la siguiente manera: Tenemos un conjunto de datos que consiste en N puntos en algún espacio $\mathcal{W}$. Se denota el n-ésimo de estos puntos como x_n, por lo que se puede escribir los datos como el conjunto $\{x_n\}_{n=1}^{N}$. Los algoritmos de clustering asignan cada uno de estos datos a uno de los K clústeres. Estas asignaciones se representan dando a cada uno

de los N datos un vector de responsabilidad binario r_n, dicho vector tiene todas sus componentes nulas excepto en la componente que corresponde al clúster al que está siendo asignado. En otras palabras, si x_n está asignado al clúster k, entonces $r_{nk} = 1$ y todos los otros elementos en r_n son cero. (Adams COS, n.d.) El siguiente algoritmo es un ejemplo en el que un entero entre 1 y K se codifica como un vector binario de longitud K que es cero en todas partes excepto en un lugar.

Figura 1.4.
Ejemplo del algoritmo K-means



Fuente: (*K-Means Clustering – What it is and How it Works – Learn by Marketing*, n.d.)

Algoritmo 1.2. Clustering K-Means (Algoritmo de Lloyd)

Vector de datos $\{x_n\}_{n=1}^N$, número de clústeres K `// Vectores de datos y número de clústeres`

for $n \leftarrow 1 \dots N$ **do**

  $r_n \leftarrow [0, 0, \cdots, 0]$ `// Inicializar todas las responsabilidades.`

  $k' \leftarrow \text{RandomInteger}(1, K)$

  $r_{nk'} \leftarrow 1$ `// Hacer que una de ellas sea uno aleatoriamente para inicializar.`

end

repeat

  for $k \leftarrow 1 \dots K$ **do**

    $N_k \leftarrow \sum_{n=1}^N r_{nk}$ `// Calcular el número asignado al clúster` k.

    $\mu_k \leftarrow \frac{1}{N_k} \sum_{n=1}^N r_{nk} x_n$ `// Calcular la media del clúster` k`-ésimo.`

  end

  for $n \leftarrow 1 \dots N$ **do**

    $r_n \leftarrow [0, 0, \cdots, 0]$ `// Reiniciar las responsabilidades.`

    $k' \leftarrow \arg\min_k \|x_n - \mu_k\|^2$ `// Encontrar la media más cercana.`

    $r_{nk'} \leftarrow 1$

  end

until *ninguno de los* r_n *cambie*;

return asignaciones $\{r_n\}_{n=1}^N$ para cada dato, y medias de los clústeres $\{\mu_k\}_{k=1}^K$

MeanShift Clustering

El algoritmo MeanShift es una técnica que consiste en encontrar los modos de una función de densidad de probabilidad para agrupar los datos en clústers. MeanShift itera moviendo cada punto de datos hacia la media de los puntos en su vecindad definida por una ventana de tamaño fijo. Adaptándolo al VRP, MeanShift se utiliza para identificar áreas de alta densidad que pueden ser atendidas por rutas individuales (Comaniciu & Meer, 2002).

Dados n puntos de datos $x_i \in \mathbb{R}^d$, la estimación de la densidad del kernel multivariante utilizando un kernel simétrico radial (por ejemplo, kernels de Epanechnikov y Gaussianos), $K(\mathbf{x})$, se define como,

$$\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right),$$

donde h (llamado parámetro de ancho de banda) define el radio del kernel. El kernel simétrico radial se define como,

$$K(\mathbf{x}) = c_k k(\|\mathbf{x}\|^2),$$

donde c_k representa una constante de normalización.

Un kernel es una función que satisface los siguientes requisitos:

1. $\int_{\mathbb{R}^d} \phi(x) = 1$,
2. $\phi(x) \geqslant 0$

Algunos ejemplos de kernels incluyen:

1. Kernel rectangular $\phi(x) = \begin{cases} 1 & \text{si } a \leqslant x \leqslant b \\ 0 & \text{de otro modo} \end{cases}$
2. Kernel Gaussiano $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$
3. Kernel de Epanechnikov $\phi(x) = \begin{cases} \frac{3}{4}(1-x^2) & \text{si } |x| \leqslant 1 \\ 0 & \text{de otro modo} \end{cases}$

Tomando la derivada de:

$$\hat{f}_K = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right),$$

Se obtiene:

$$\nabla \hat{f}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)\mathbf{x}_i\right] - \mathbf{x}\left[\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)\right],$$

donde $g(x) = -k'(x)$ denota la derivada del perfil del kernel seleccionado.

Finalmente, el *mean shift* desde un punto dado es (Comaniciu & Meer, 2002):

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}\right] - \mathbf{x}$$

Algoritmo 1.3. Procedimiento de Mean Shift

Puntos de datos $x_i \in \mathbb{R}^d$, ancho de banda h Puntos de datos trasladados $\mathbf{x}_i^{t+1}$

El algoritmo sería el siguiente: **for** *Cada punto de datos* $\mathbf{x}_t$ **do**

    repeat

        for $i \leftarrow 1$ **to** n **do**

            `// 1. Calcular el vector de mean shift` **m**

            $$\mathbf{m} \leftarrow \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\mathbf{x}-\mathbf{x}_i}{h} \right\|^2 \right)} \right] - \mathbf{x};$$

            `// 2. Trasladar la ventana de densidad`

            $\mathbf{x}_i^{t+1} \leftarrow \mathbf{x}_i^t + \mathbf{m}(\mathbf{x}_i^t);$

        end

    until *convergencia*;

end

`// Iterar hasta que`

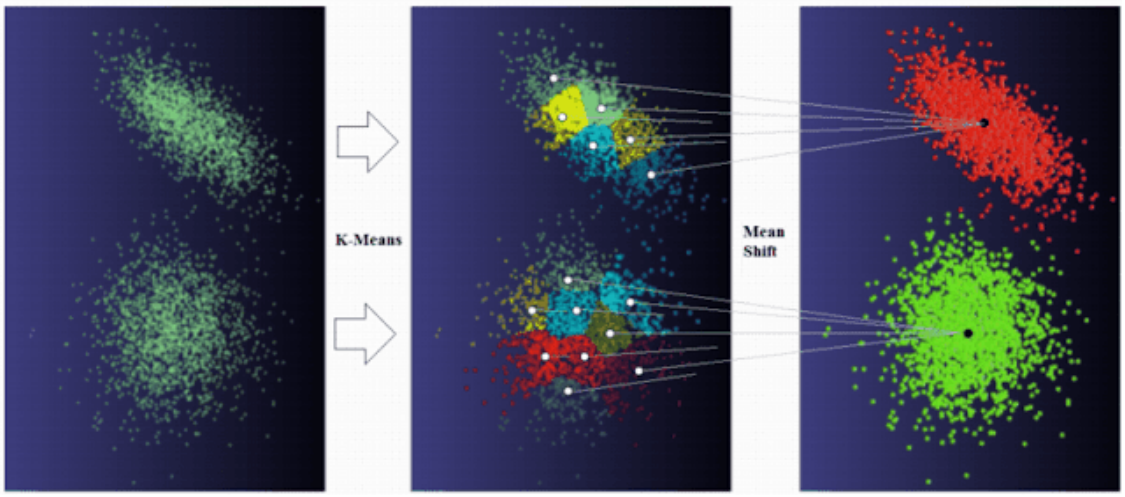
$\nabla f(\mathbf{x}_i) = 0;$

Spectral Clustering

El algoritmo Spectral Clustering utiliza las propiedades del espectro (valores propios) de la matriz de similitud de los datos para realizar el agrupamiento. Este algoritmo convierte el problema de clustering en un problema de partición de grafos, donde los puntos de datos se representan como nodos y las conexiones entre ellos se representan como aristas ponderadas (Ng, Jordan, & Weiss, 2002).

Dado un conjunto de observaciones $(x_1, x_2, \ldots, x_n)$, donde cada observación es un vector

Figura 1.5.
Ejemplo de comparación entre K-means y MeanShift



Fuente: (Li, n.d.)

real de d dimensiones, el Spectral Clustering tiene como objetivo encontrar una partición de los datos que minimice el corte entre los clústers mediante la resolución del problema de valores propios de la matriz de Laplace normalizada L:

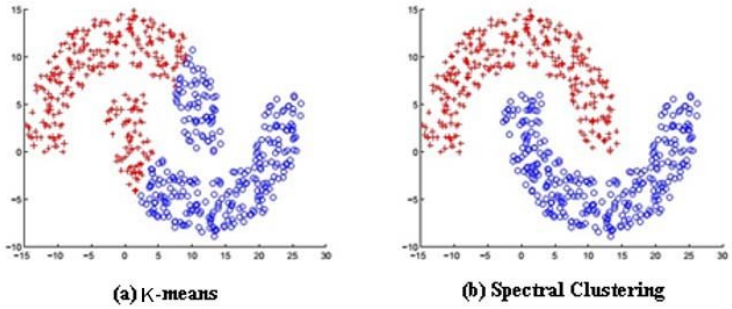
$$L = D^{-1/2} W D^{-1/2},$$

donde W es la matriz de similitud y D es la matriz diagonal de grados. Los k vectores propios correspondientes a los k menores valores propios de L forman una matriz U que se utiliza para particionar los datos aplicando un algoritmo de clustering, como K-means, sobre U.

Agglomerative Clustering

De acuerdo con Johnson (1967) el algoritmo Agglomerative Clustering es un tipo de clustering jerárquico que construye una jerarquía de clústers mediante un enfoque ascendente.

Figura 1.6.
Ejemplo de comparación entre K-means y Spectral clustering



Fuente: (Ben Ayed, Ben Halima, & Alimi, 2017)

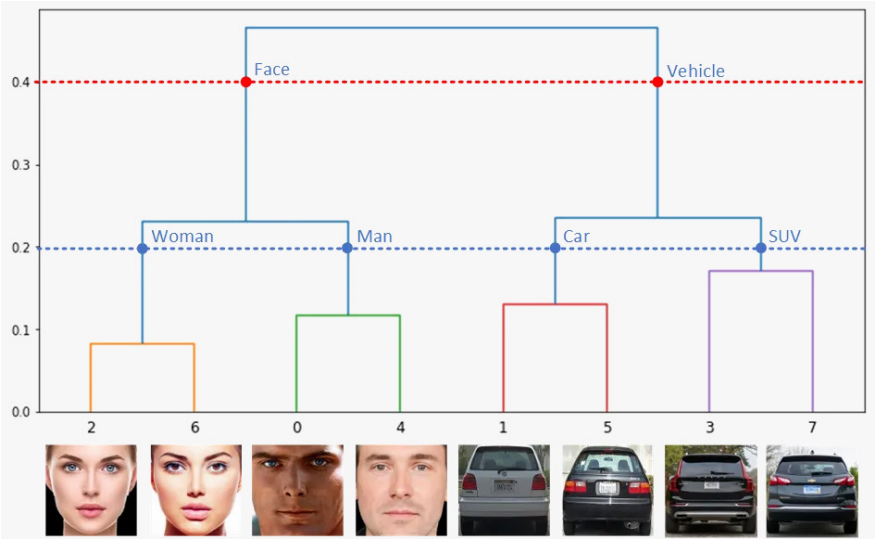
Comienza tratando cada punto de datos como un clúster individual y, en cada paso, fusiona los dos clústers más cercanos hasta que todos los puntos estén en un solo clúster o se cumpla un criterio de parada.

Dado un conjunto de observaciones $(x_1, x_2, \ldots, x_n)$, donde cada observación es un vector real de d dimensiones, el Agglomerative Clustering tiene como objetivo construir una dendrograma que represente la jerarquía de clústers. La distancia entre dos clústers C_i y C_j se define mediante un criterio de enlace, como el enlace completo:

$$d(C_i, C_j) = \max_{x \in C_i, y \in C_j} \|x - y\|$$

donde $\|\cdot\|$ es la norma euclídea, aunque puede cambiar la formulación con otra norma. Por tanto, el algoritmo para el aglomerative clustering es el siguiente.

Figura 1.7.
Ejemplo de Agglomerative Clustering



(Nemutlu, 2024)

Input: Conjunto de puntos $S = \{s_1, \ldots, s_n\} \subseteq \mathbb{R}^d$, número de clusters k
Output: Asignaciones de cluster para cada punto s_i

// 1. Formar la matriz de afinidad $A \in \mathbb{R}^{n \times n}$
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** n **do**
    if $i \neq j$ **then**
      $A_{ij} \leftarrow \exp\left(-\frac{\|s_i - s_j\|^2}{2\sigma^2}\right)$;
    else
      $A_{ij} \leftarrow 0$;
// 2. Definir D como la matriz diagonal
for $i \leftarrow 1$ **to** n **do**
  $D_{ii} \leftarrow \sum_{j=1}^{n} A_{ij}$;
Construir la matriz $L \leftarrow D^{-1/2} A D^{-1/2}$;
// 3. Encontrar los k mayores eigenvectores de L
Calcular los k mayores eigenvectores de L: $x_1, x_2, \ldots, x_k$;
Formar la matriz $X \leftarrow [x_1\ x_2\ \ldots\ x_k] \in \mathbb{R}^{n \times k}$;
// 4. Formar la matriz Y
for $i \leftarrow 1$ **to** n **do**
  for $j \leftarrow 1$ **to** k **do**
    $Y_{ij} \leftarrow \frac{x_{ij}}{\left(\sum_{j=1}^{k} x_{ij}^2\right)^{1/2}}$;
// 5. Tratar cada fila de Y como un punto en $\mathbb{R}^k$
Aplicar K-means u otro algoritmo de clustering a las filas de Y para formar k clusters;
// 6. Asignar los puntos originales a los clusters
for $i \leftarrow 1$ **to** n **do**
  Asignar el punto original s_i al cluster j si y sólo si la fila i de Y fue asignada al cluster j;

CAPÍTULO 2

2. METODOLOGÍA

2.1 Introducción y tratamiento de datos

En esta investigación, se utilizaron datos proporcionados por una empresa pública sobre la división territorial del país que comprende 201,995 lugares definidos. En estos datos, se tiene la ubicación geográfica de cada manzana del país. Por otro lado, se tienen 12 muestras aleatorias correspondientes a cada mes del año. Cada mes contiene 3,200 lugares designados para visitas mensuales por parte de los empleados. Dado que la muestra es aleatoria, puede incluir a las islas Galápagos como lugar donde los empleados deben ir. La empresa cuenta con un representante adicional exclusivamente para el archipiélago, por tanto no se tomó en cuenta para el desarrollo de los algoritmos.

El proceso metodológico se inició con la obtención de los datos en formato shapefile. Posteriormente, se empleó el software de código abierto QGIS para quitar las islas Galápagos y la isla Puná de la muestra, la primera por los motivos presentados en el párrafo anterior y la segunda por no existir conexiones terrestres directas hasta dicha isla. Como existían problemas en la geometría de los datos por lo que se implementó la función `Check Geometry`. Además, se tomaron los centroides de cada manzana para obtener sus coordenadas geográficas, esto fue realizado a través de las funciones `centroids` y `Add X/Y fields to layer`. Las coordenadas

de estos centroides se extrajeron en el formato WGS 84 (EPSG:4326) , también conocida como web Mercator, que es una adaptación de la proyección de Mercator de S^2 a $\mathbb{R}^2$. El motivo para la elección de este formato es por ser el estándar en servicios de mapeo y ruteo como Google maps y Azure maps. Las coordenadas fueron almacenadas en un archivo con formato CSV, que incluye un código de identificación único para cada lugar junto con sus coordenadas geográficas correspondientes, por motivos de confidencialidad con la empresa, no se explicará dicha codificación. Es importante señalar que la muestra proporcionada carecía de información crucial: los tiempos de desplazamiento de los empleados a los lugares designados, una estimación del tiempo de visita para cada lugar y el tiempo que les tomó completar la planificación de rutas para un mes, es decir, no hubo datos acerca de si se logró completar la planificación o no. Esta información serviría como punto de comparación para los resultados del modelo. Esta ausencia de datos es la principal razón por la que aplicar una modelo más cercano a la realidad, como el VRPTW, fue imposible, además, limitó la capacidad de evaluar el rendimiento del modelo propuesto para futuras mejoras.

Distancias y tiempos

El modelo requiere una matriz de distancia y tiempo. Inicialmente, se consideró utilizar la distancia de Manhattan o de norma $p = 1$, pero se identificaron dos inconvenientes principales:

1. La distancia de Manhattan solo permite movimientos horizontales y verticales. En situaciones donde el siguiente lugar a visitar está en la misma manzana, esta medida sobrestima la distancia real en comparación con métodos más precisos, como los proporcionados por la API de Google Maps.

2. Para poder calcular calcular la matriz de tiempos se tendría que suponer una velocidad constante del vehículo, y omitir consideraciones logísticas como los semáforos y el tráfico, esto es válido para un acercamiento puramente académico, sin embargo, para operaciones reales no es conveniente.

Debido a estas limitaciones, se exploraron alternativas como las APIs de Google Maps y Azure Maps. Aunque ambos servicios ofrecen un crédito mensual gratuito de 200 dólares, la escala del problema, que requiere calcular una matriz de distancia de al menos 140x140, excede significativamente este presupuesto. Como solución, se optó por Open Route Service (ORS), una alternativa de código abierto. Al igual que Google maps, esta API permite obtener la matriz de distancias de manejo, es decir, la distancia que debe recorrer un automóvil para llegar de punto A a punto B, y la matriz de tiempo representa el tiempo de conducción de dicha ruta, considerando el tráfico[1]. Inicialmente, se consideró utilizar la API gratuita de ORS, pero esta presentaba restricciones, limitando el cálculo de la matriz de distancia a 50x50 y estableciendo límites en las peticiones. Para superar estas limitaciones, se implementó ORS de manera local a través de un servidor, lo que permitió eliminar las restricciones en el cálculo de las matrices de distancia y tiempo. Es importante señalar que la implementación local de ORS puede presentar desafíos en cuanto a los requisitos de hardware. Para datos a nivel mundial, se recomienda un servidor con al menos 128 GB de RAM. Sin embargo, para este estudio, que se centra exclusivamente en Ecuador, un servidor con 32 GB de RAM resulta suficiente. Los datos necesarios se obtuvieron de Open Street Map. A continuación, se detallan las especificaciones del vehículo utilizado como servidor para ORS.

[1] El tráfico es considerado en el instante que se hace la petición al servidor.

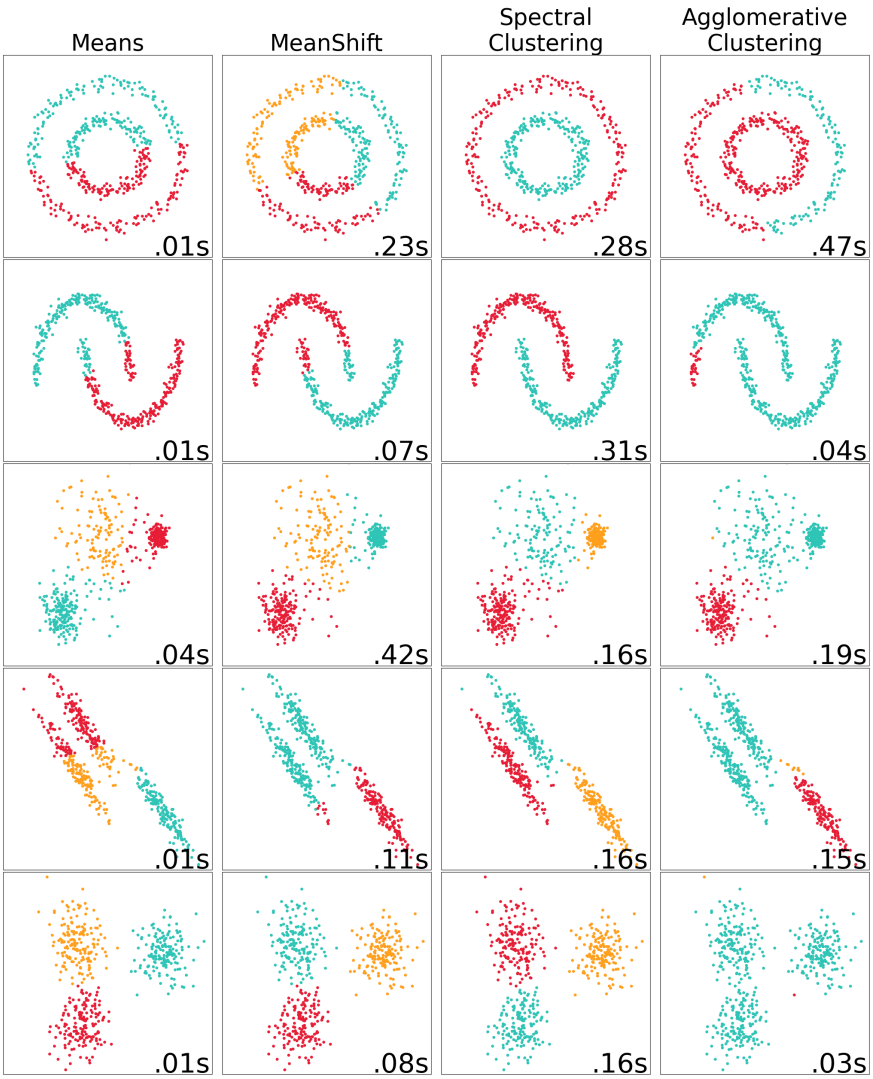
Componente	Especificación
Procesador	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Núcleos	4
Tarjeta gráfica	Intel(R) UHD Graphics 620 NVIDIA GeForce MX110
Memoria RAM	31,92 GB

Tabla 2.1.
Especificaciones del sistema

Esquema de la solución

Como se mencionó en la sección 1.1 se trabajó con una muestra de 3200 puntos, que después de retirar las islas, y arreglar la geometría consta de 3040 puntos. Ahora, el problema consiste en determinar las rutas que deben seguir siete vehículos para lograr visitar estos 3040 puntos en 20 días laborables, debido a esta restricción de tiempo se ha decidió organizar los clientes en $140 = (20)(7)$ grupos, esta agrupación ha sido realizada a través del método de k-means clustering, las razones para la elección de este método de clustering son descritas a continuación: Dado que el método busca minimizar la varianza de los elementos en un clúster, es ideal para conjuntos de datos donde los elementos no tengan una forma o distribución no corpuscular, es decir, una que no presente una similitud con bolas en $\mathbb{R}^2$, esto puede ser apreciado en la Figura 2.1.

Figura 2.1.
Diferencias entre algoritmos de clustering para distintos datasets.



Fuente: Elaboración propia.

Por ejemplo, para la primera, segunda y cuarta fila los datos no presentan una estructura similar a crepúsculos, y por tanto el k-means realiza una mala clasificación, mientras que el mean shift da excelentes resultados. Por otro lado, para datos que están más dispersos pero con una tendencia a tener lugares de acumulación (o estructura corpuscular), como la tercera y

quinta columna, el k-means realiza una clasificación mucho mejor. Si bien es cierto los otros métodos también presentan clasificaciones similares para esta distribución de datos, el k-means tiene una ejecución considerablemente menor, de hecho el k-means tiene una complejidad de tiempo de $O(Ikdn)$ donde I es el número de iteraciones, n el número de puntos, k el número de clústers y d la dimensión de los datos Sachinsoni (2023), en otras palabras, el algoritmo tiene un tiempo de ejecución lineal, lo cual es muy eficiente. En particular, la distribución de los datos es observable en la Figura 2.2, por tanto, la mejor opción de clustering es el k-means.

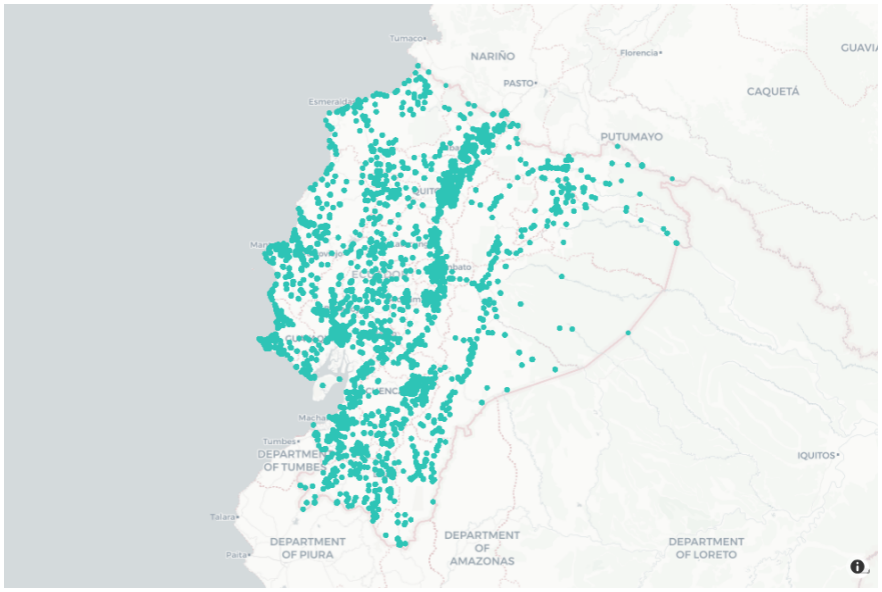


Figura 2.2.
Distribución de los datos a nivel territorial.

Fuente: Elaboración propia.

Una vez se realizó el clustering se obtuvieron 140 lugares a ser visitados, estos representan el cliente más cercano al centroide del clúster encontrado por el k-means. La razón de tomar el cliente más cercano al centroide del clúster viene dada porque el centroide no necesariamente pertenece a la muestra, y puede darse el caso de que para dicho punto no exista una ruta

automovilística que lo conecte. Después de hacer esta corrección se realiza un CVRP con los 140 lugares y 7 vehículos, la distancia entre los lugares es proporcionada por la API de open route service. Para garantizar que cada vehículo utilice los 20 días de servicio en operación se le otorga a cada lugar una demanda de 1 y a cada camión una capacidad de 20. Esto nos da 7 rutas, una para cada vehículo, donde cada lugar representa la ubicación geográfica donde se deben realizar las operaciones en cada día, por supuesto, cada lugar representa una colección de clientes a visitar, en promedio se tienen 18 clientes por cada lugar. Luego, se aplica un TSP en cada clúster para determinar la ruta que se debe seguir cada vehículo en cada día, es decir, se realiza un CVRP con 140 puntos y 7 vehículos y luego 140 TSPs (Uno por cada clúster).

Software y recursos utilizados

Para la solución del CVRP (y por consiguiente del TSP), se eligió la heurística de búsqueda genética híbrida, propuesta por Vidal (2022) las razones para usar esta heurística sobre otras viene justificada por su alta eficiencia. La eficiencia del algoritmo es medido a través del valor $GAP = 100(c - c_{MSE})/c_{MSE}$, donde c es la solución encontrada por el algoritmo y c_{MSE} es la mejor solución encontrada dicha instancia [2], en otras palabras, el GAP determina el porcentaje de error entra la mejor solución encontrada y la solución encontrada por el algoritmo. Las instancias vienen dadas en el siguiente formato $X - nN - kK$, donde N es el número de clientes y K el número de vehículos. Para la ejecución de las heurísticas se tomó un criterio de parada de tiempo máximo dado por $T_{max} = n * 2.4$ segundos, cada algoritmo fue ejecutado 10 veces y se obtuvo su GAP, estos datos se ven reflejados en la siguiente tabla, adaptada de los datos presentados por Vidal (2022).

[2] Dichos valores pueden ser encontrados en http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

Tabla 2.2.
Comparativa del GAP entre distintos solvers y heurísticas.

Instancia	OR-Tools	LKH-3	HILS	KGLS	SISR	FILO	HGS-2012	HGS-CVRP
X-n110-k13	0.86	0.15	0.00	0.00	0.01	0.00	0.00	0.00
X-n115-k10	0.48	0.00	0.00	0.00	0.00	0.00	0.00	0.00
X-n120-k6	1.27	0.01	0.01	0.00	0.00	0.00	0.00	0.00
X-n125-k30	2.37	0.66	0.55	0.36	0.04	0.28	0.00	0.00
X-n129-k18	2.70	0.50	0.11	0.11	0.03	0.03	0.00	0.00
X-n134-k13	2.34	0.50	0.29	0.22	0.20	0.11	0.00	0.00
X-n139-k10	1.11	0.48	0.01	0.00	0.00	0.00	0.00	0.00
X-n143-k7	2.77	0.43	0.23	0.19	0.18	0.15	0.00	0.00
X-n148-k46	2.65	0.16	0.00	0.32	0.04	0.07	0.00	0.00

Fuente: (Vidal, 2022)

Donde,

- **OR-Tools:** Un solver proporcionado por Google.
- **LKH-3:** Una versión modificada de la heurística de Lin-Kernighan.
- **HILS:** Búsqueda local híbrida iterada.
- **KGLS:** Búsqueda local de conocimiento guiado.
- **SISR:** Inducción de holgura mediante remoción de cuerdas.
- **FILO:** Optimización localizada iterativa rápida.
- **HGS-2012:** Implementación de búsqueda genética híbrida.
- **HGS-CVRP:** Re-implementación de búsqueda genética híbrida optimizada para CVRP.

Así, de la tabla 2.2 se tiene que el $HGS-CVRP$, en promedio, siempre alcanza la mejor solución, mientras que las otras heurísticas presentan errores en las mismas instancias, siendo en muchos casos la única que alcanza a la mejor solución. Es importante mencionar que en el artículo de Vidal Vidal (2022) se presentan más instancias con mayor número de datos, nos hemos limitado a estas instancias por ser las que más se acercan al problema, sin embargo, para instancias con mayores datos la tendencia descrita se mantiene. Por estos motivos, se optó por implementar HGS para resolver el CVRP. El autor de la versión de HGS optimizada para CVRP también pone a disposición una implementación del algoritmo en `C++`; sin embargo, se optó por una implementación en `Python` a través de la librería `PyVRP` propuesta por Wouda, Lan, & Kool (2024), los motivos de usar `Python` sobre `C++` son los siguientes:

1. La empresa comentó que tenía predilección por implementaciones en `Python`, sobre todo por su familiaridad con Jupyter Notebooks, que es una instancia web para ejecutar fragmentos de códigos en `Python`.

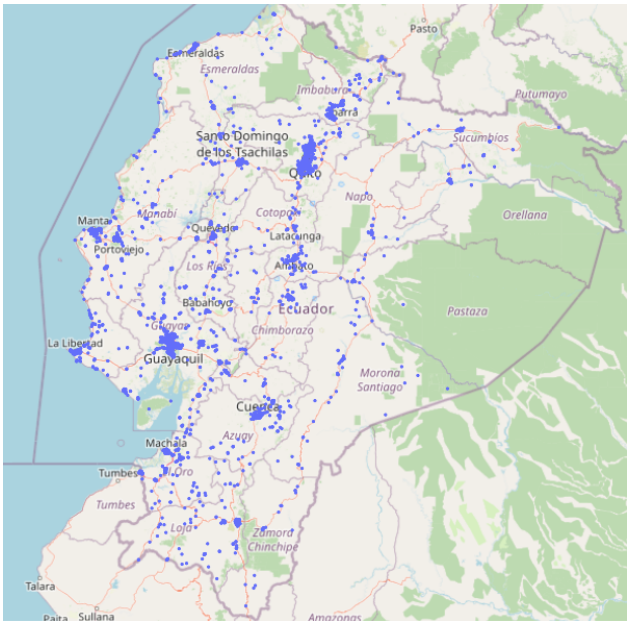
2. La gran cantidad de librerías de `Python` permite lograr una implementación sencilla, y fácil de entender por personas con un nivel básico del lenguaje.

3. Existen gran cantidad de librerías que permiten realizar gráficos con interactividad, perfectos para una implementación en la que se requieren obtener rutas.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS Para la realización del ruteo se realizó un programa en Python con ayuda de la librería PyVRP. Las pruebas fueron realizadas en una computadora con las siguientes características: Intel (R) Core (TM) i7-4770 @ 3.40GHz 8.00 GB RAM en el sistema operativo Fedora Workstation 38. Para determinar la utilidad de la solución, se obtuvo una muestra de 3200 manzanas en todo el país, sin incluir las regiones insulares, de forma aleatoria. La distribución de las manzanas en el mapa del Ecuador se observa en la Figura 3.1.

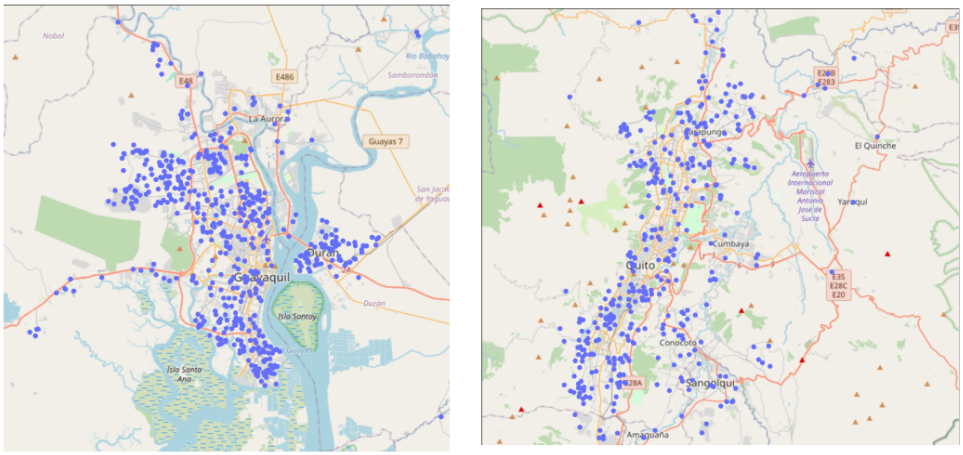
Figura 3.1.
Distrubución de las 3200 manzanas.



Fuente: Elaboración propia.

Como se puede apreciar, existen agrupaciones naturales que corresponden a las ciudades. Por ejemplo, los dos lugares con mayor concentración de puntos son la ciudad de Guayaquil y la ciudad de Quito. En la siguiente imagen se puede ver la distribución de las manzanas en dichas ciudades:

Figura 3.2.
Comparativa entre Quito y Guayaquil.



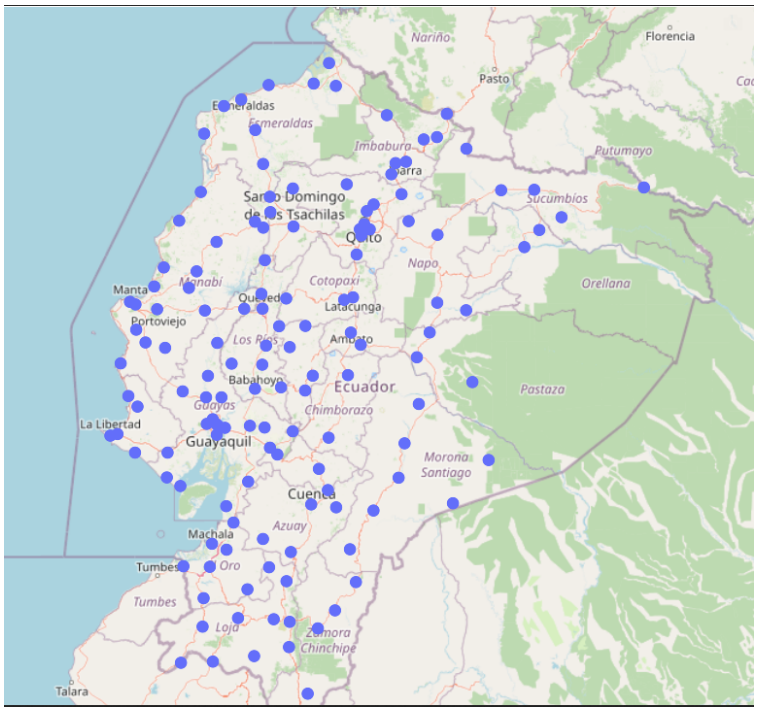
Fuente: Elaboración propia.

Por tanto, es de esperar que aquellos vehículos que deban operar en las ciudades tendrán el mayor número de manzanas a visitar. Esta suposición se vio verificada cuando se realizó el problema micro. Con la distribución presentada, se procedió a realizar la solución del problema macro y el problema micro.

3.0.1 Problema Macro

Para la solución global o problema macro, primero se hizo una agrupación en 140 clústers. Como se apreció en la Figura 3.1, existen agrupaciones naturales. Tras usar el algoritmo de K-means se obtuvo la agrupación mostrada en la Figura 3.3.

Figura 3.3.
Centroides de los 140 clústers.

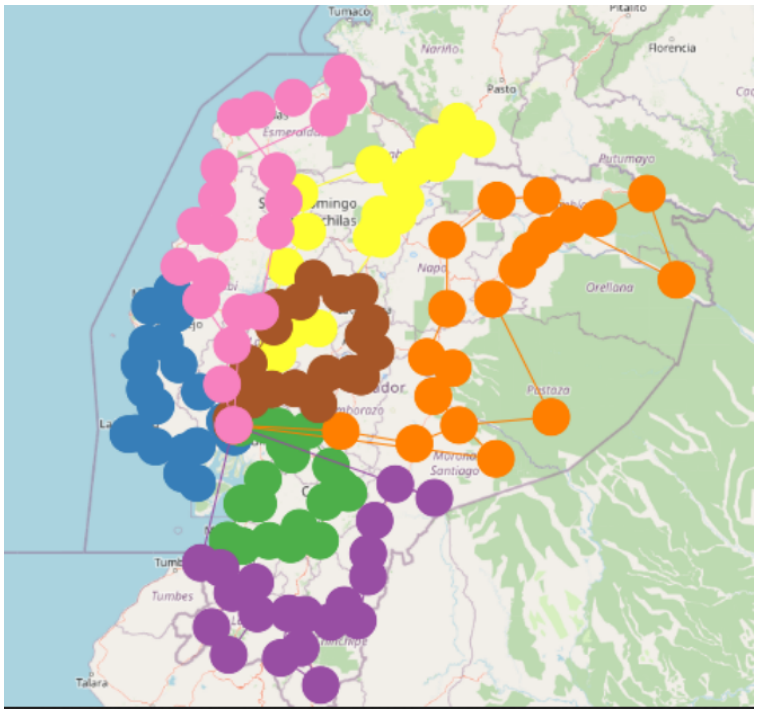


Fuente: Elaboración propia.

Luego, se implementó el VRP para determinar las rutas a recorrer por cada vehículo en los 20 días de trabajo. La planificación se puede ver en la Figura 3.4. Donde los colores azul, verde, morado, naranja, amarillo, café y rosa representan los vehículos 1,2, 3, 4, 5, 6 y 7, respectivamente. En particular, se pude fijar un vehículo para observar su solución. Por ejemplo, en la Figura 3.5 se puede observar una comparativa entre la ruta del vehículo 1 y el vehículo 4. El vehículo 1 recorre las provincias del Guayas, Santa Elena y Manabí. Para este vehículo los lugares a visitar se encuentran relativamente cerca, por ejemplo la distancia entre el depósito y el primer día de trabajo para el equipo 1 es 2.5 km, pues ambos están en la ciudad de Guayaquil, mientras que la distancia del depósito al primer día de trabajo del equipo 4 (color naranja) es 406km, representado un viaje de poco menos de 7 horas. En otras palabras, un día de trabajo

se pierde solo en partir del depósito al lugar donde se deben hacer las actividades del primer día, esto se debe a que la empresa debe hacer labores en todo el país pero solo tiene un depósito, obligando a que sus vehículos deban partir desde la ciudad de Guayaquil a distintas provincias. Una forma de solucionar esto, por ejemplo, sería abrir un depósito desde la ciudad de Quito para contemplar toda la sierra y Amazonia ecuatoriana.

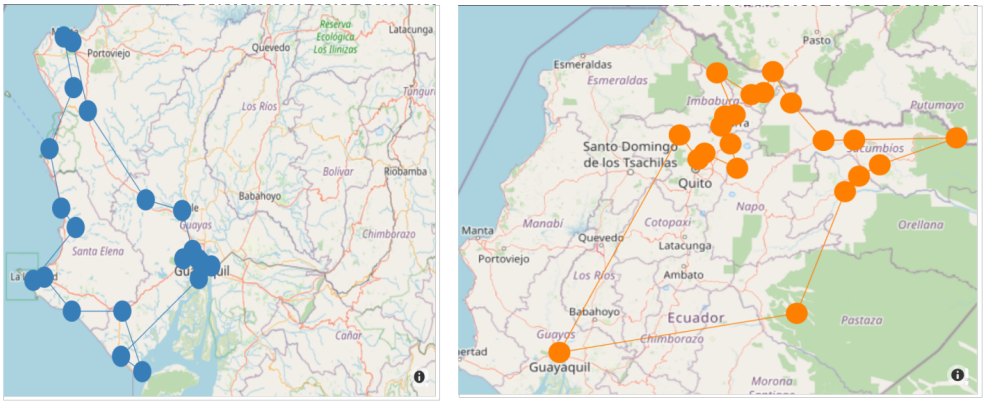
Figura 3.4.
Planificación de cada uno de los vehículos.



Fuente: Elaboración propia.

Como se mencionó antes, en las ciudades existe mayor número de manzanas a ser visitidas, en particular, el 32% de las manzanas se encuentran en la ciudad de Guayaquil. Luego de hacer la agrupación se obtuvieron 5 clústers en esta ciudad. En la figura 3.6 se ven representadas las manzanas, donde los colores representan cada uno de los clústers a los que pertenecen. Esto quiere decir que existen varios vehículos que deberán hacer rutas dentro de la

Figura 3.5.
Comparación vehículo 1 y vehículo 4.

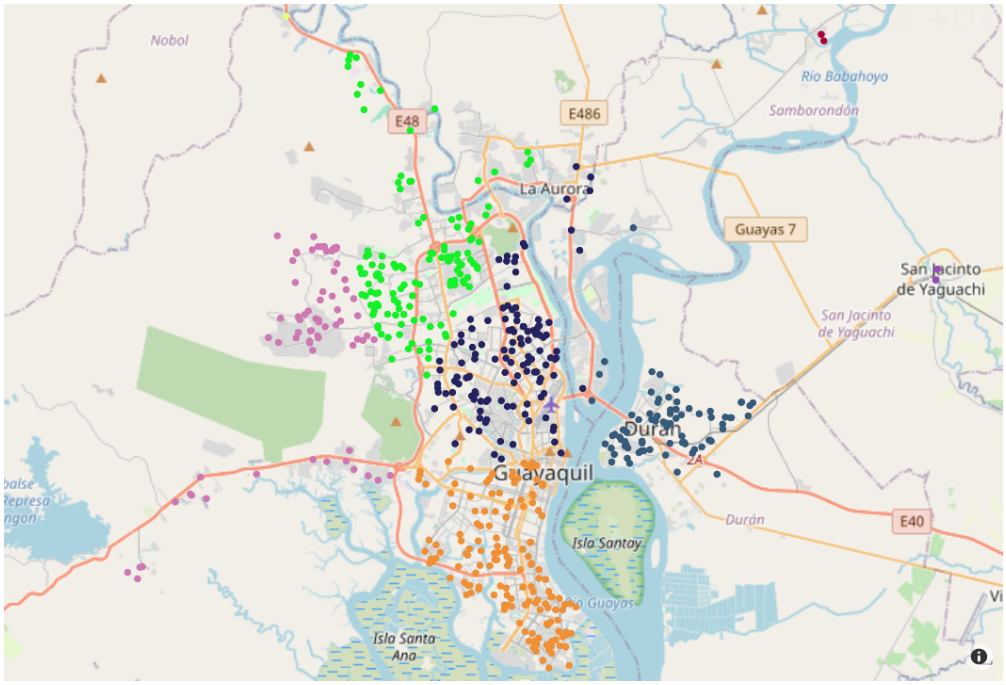


Fuente: Elaboración propia.

ciudad de Guayaquil. Por supuesto, estas agrupaciones tienen un gran número de manzanas. Por ejemplo la manzana de color azul oscuro tiene 63 manzanas, esto implica una carga elevada de trabajo para los vehículos. Esto ocurre porque el k-means hace la agrupación sin tener en consideración la carga de trabajo, una posible solución a esto es tomar las distancias de todos los puntos de la muestra y hacer una agrupación equitativa para cada clúster. Es decir, cada clúster tendría 3200/140 puntos donde los puntos dentro del clúster son los más cercanos de acuerdo a una distancia. Por supuesto, al fijar el número de puntos en cada clúster se pierde calidad en la agrupación, pues hay manzanas que están alejadas de otras, por estar en zonas rurales del país.

Para determinar las rutas de cada día y calcular los tiempos de las rutas así como las distancias, se resolvió el problema micro.

Figura 3.6.
Agrupaciones de manzanas en Guayaquil y Durán.



Fuente: Elaboración propia.

3.0.2 Problema Micro

En la resolución del problema micro se obtuvieron 140 rutas, cada una representando las actividades a seguir de un vehículo determinado en un día determinado. Por ejemplo, las agrupaciones de manzanas vistas en la figura 3.6 corresponden al vehículo 1 en los días 1, 2, 3, 19 y 20. La ruta que debe seguir el vehículo en cada uno de esos días se puede ver en la figura 3.7.

De acuerdo con la validación realizada y el GPS de la API de Open Route Service, se obtuvo la duración de cada ruta y la distancia de cada ruta. Para las planificaciones de la figura 3.7 se tiene la tabla 3.1.

Tabla 3.1.
Comparativa de cuatro días para el equipo 1.

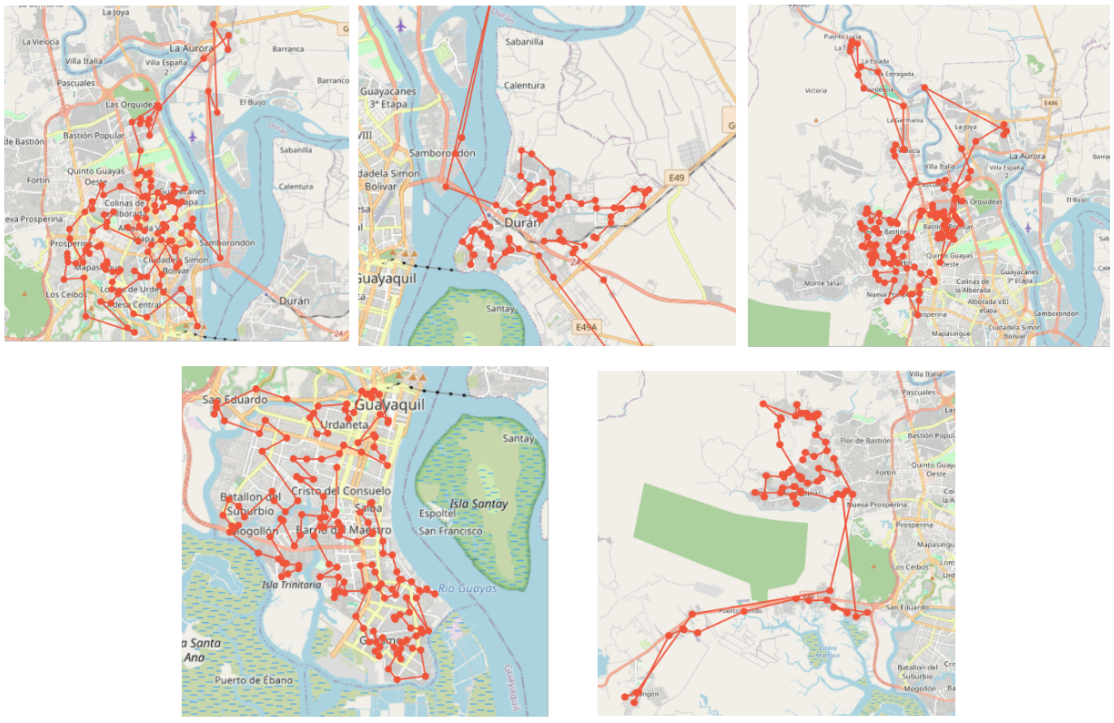
Día	Duración (Horas)	Distancia (Km)
1	6.82	180.6
2	5.22	155.14
3	6.79	167.65
19	6.43	139.95
20	4.39	121.27

Para la duración de las rutas, la API usa el tráfico, pero solo en el momento que se hace la petición. Por supuesto, esto es una aproximación modesta a la realidad, pues el tráfico varía dependiendo de la hora del día, además, para aquellos puntos donde no se encontró ruta de manejo, se optó por usar la distancia geodésica [1] y se supuso una velocidad de 70km, lo cual es correcto para carreteras según la legislación de tráfico [2], pero falla en ciudades, pues estas tiene distintas limitaciones de velocidad, variando desde los 30km hasta los 70km, dependiendo de la zona. Por tanto, se recomienda tomar estos resultados como base para mejor la implementación a través de la experiencia adquirida en las anteriores planificaciones, o mejor aún, proponer una heurística específica para este problema, con base a las otras planificaciones. Es importante mencionar que si una ruta tiene un duración menor a la jornada laboral, la empresa puede decir si el vehículo descansa o avanza con el siguiente día.

[1] El equivalente a una línea recta.

[2] Los vehículos pesados pueden conducir hasta 90km.

Figura 3.7.
De izquierda a derecha. Rutas para el vehículo 1, día 1, 2, 3, 19 y 20.



Fuente: Elaboración propia.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Se implementó una heurística de búsqueda genética híbrida para desarrollar un programa que asigna rutas de forma automatizada, con el objetivo de minimizar la distancia recorrida por los vehículos. Como resultado, se logró reducir el tiempo de planificación de dos días de trabajo manual, a realizarla en poco más de una hora sin supervisión, representado una mejoría del 89%;
- Se evaluó la efectividad del modelo con base a la literatura científica, lo cual indica que los métodos utilizados tienen un margen de error menor al 0.1% en tiempos de ejecución menores a dos horas. Por otro lado, a través de API de código abierto se validó la viabilidad de las rutas, demostrando que pueden realizarse dentro de un horario laboral estándar de ocho horas;
- Se analizaron los resultados y se concluyó que la principal limitante del modelo fue la falta de datos. Por tanto se recomienda recopilar datos de los clientes, como horarios de visitas, probabilidad de estar en dicho horario de vista y coordenadas precisas de cada cliente, no solo del sector. Estos datos permitirán usar modelos más precisos y obtener mejores rutas.

Recomendaciones

- Hacer uso de aplicaciones GPS con licencia comercial, que permitan obtener tiempos de conducción que consideren el tráfico de manera dinámica, y que no tengan límite de peticiones de tal forma que puedan ejecutar el programa varias veces para realizar ajustes a las rutas, según sea necesario. Además, se recomienda obtener datos de los clientes para incluir variables importantes en el modelo como: disponibilidad de visita, duración promedio de las visitas, probabilidad de visita conseguida.

- Verificar las coordenadas de los clientes, de lo contrario, podrían existir clientes para los que no se disponga de rutas vehiculares conectadas. Sin esto, se afecta la calidad de la solución, ya que sería necesario deben usar distancias geométricas que no representan de manera correcta la distancia de conducción.

- Permitir a los conductores decidir si variar la ruta de acuerdo a las condiciones de tráfico se ese día y la longitud de la ruta de ese día, pudiendo pasar localizaciones a distintos días.

- Abrir otros depósitos en zonas estratégicas como Quito, para cubrir la zona toda la sierra y costa norte, sin tener que hacer viajes desde el depósito de Guayaquil, evitando perder días de trabajo en la movilización diaria.

REFERENCIAS

Adams COS, R. P. (n.d.). *K -means clustering and related algorithms.* https://www.cs.princeton.edu/courses/archive/fall18/cos324/files/kmeans.pdf.

AIMMS. (2020a, 8). *Capacitated vehicle routing problem formulation.* Retrieved from https://how-to.aimms.com/Articles/332/332-Formulation-CVRP.html

AIMMS. (2020b, 8). *Explicit dantzig-fulkerson-johnson formulation.* Retrieved from https://how-to.aimms.com/Articles/332/332-Explicit-Dantzig-Fulkerson-Johnson-formulation.html

AIMMS. (2020c, 6). *Time windows.* Retrieved from https://how-to.aimms.com/Articles/332/332-Time-Windows.html

Ben Ayed, A., Ben Halima, M., & Alimi, A. (2017, 07). Adaptive fuzzy exponent cluster ensemble system based feature selection and spectral clustering. In (p. 1-6). doi: 10.1109/FUZZ-IEEE.2017.8015721

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers Industrial Engineering*, *99*, 300-313. Retrieved from https://www.sciencedirect.com/science/article/pii/S0360835215004775 doi: https://doi.org/10.1016/j.cie.2015.12.007

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603–619.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80-91. Retrieved from https://doi.org/10.1287/mnsc.6.1.80 doi: 10.1287/mnsc.6.1.80

Drexl, M. (2012). Rich vehicle routing in theory and practice. *Logistics Research*, *5*, 47-63. doi: https://doi.org/10.1007/s12159-012-0080-2

Fukasawa, R., Longo, H., Lysgaard, J., Poggi, M., Reis, M., Uchoa, E., & Werneck, R. (2006, 07). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Math. Program.*, *106*, 491-511. doi: 10.1007/s10107-005-0644-x

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*(3), 241–254.

K-Means Clustering – What it is and How it Works – Learn by Marketing. (n.d.). Retrieved from https://www.learnbymarketing.com/methods/k-means-clustering/

Kriegel, H. P., Schubert, E., & Zimek, A. (2017, 8). The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems*, *52*, 341-378. doi: 10.1007/s10115-016-1004-2

Li, J. (n.d.). *On Mean Shift and K-Means Clustering.* Retrieved from https://jamesxli.blogspot.com/2012/03/on-mean-shift-and-k-means-clustering.html

Nemutlu, D. (2024, 7). Hierarchical Clustering of Images with Python - Dahi Nemutlu - Medium. Retrieved from https://medium.com/@dnemutlu/hierarchical-clustering-of-images-with-python-f99e92855069

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, *14*, 849–856.

Ritchie, H. (2020, 3). *Cars, planes, trains: where do co emissions from transport come from?* Retrieved from https://ourworldindata.org/co2-emissions-from-transport

Sachinsoni. (2023, 12). *The art and science of k-means clustering: A practical guide.* Retrieved from https://medium.com/@sachinsoni600517/the-art-and-science-of-k-means-clustering-a-practical-guide-e71b11638867

Toth, P., & Vigo, D. (2012). *Vehicle routing problems, methods, and applications* (second ed.). SIAM.

Vidal, T. (2022). Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers Operations Research*, *140*, 105643. Retrieved from https://www.sciencedirect.com/science/article/pii/S030505482100349X doi: https://doi.org/10.1016/j.cor.2021.105643

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012, 06). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, *60*, 611-624. doi: 10.1287/opre.1120.1048

Wouda, N. A., Lan, L., & Kool, W. (2024). PyVRP: a high-performance VRP solver package. *INFORMS Journal on Computing*. Retrieved from https://doi.org/10.1287/ijoc.2023.0055 doi: 10.1287/ijoc.2023.0055