UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

OPTIMIZACIÓN DE DISTRIBUCIÓN LOGÍSTICA DE PRIMERA MILLA PARA MARKETPLACE EN RETAIL

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

JOAQUÍN CUBELLI DE LEÓN

PROFESOR GUÍA:
ALFREDO VEGA CURIHUINCA

PROFESOR CO-GUÍA:
FRANCISCO RIVERA SERRANO

COMISIÓN:
IVÁN CASTRO OJEDA

SANTIAGO DE CHILE
2023

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: JOAQUÍN CUBELLI DE LEÓN
FECHA: 2023
PROF. GUÍA: ALFREDO VEGA CURIHUINCA

OPTIMIZACIÓN DE DISTRIBUCIÓN LOGÍSTICA DE PRIMERA MILLA PARA MARKETPLACE EN RETAIL

El presente trabajo se centra en el desarrollo de una plataforma que soluciona el problema de asignación de vehículos en rutas de primera milla para el *Marketplace* de una empresa de retail chilena, donde previamente la planificación de las rutas se hacía de forma manual.

Para lograr una solución que mejorase el uso de los camiones, se desarrollan aspectos desde la generación de los datos para definir correctamente el problema, pasando por la forma en la que se modela el problema y sus restricciones, metaheurísticas asociadas para obtener soluciones en tiempos cortos y, por último, la acción de poner en producción lo desarrollado a través de un *front end*.

Para este caso en particular, se obtuvieron soluciones al problema de ruteo de vehículos considerando restricciones de capacidad y ventanas horarias, resultando en la automatización de la toma de decisiones por parte de las personas encargadas de planificar, y en disminuciones de costos de hasta un 31 %.

Agradecimientos

Quiero expresar mi profundo agradecimiento a las personas que me han brindado su apoyo y motivación durante la realización de esta memoria.

En primer lugar, quiero agradecer a mis padres Ana y Martín y, a mi hermana Victoria por su amor incondicional y paciencia en todo momento. Gracias por estar siempre ahí para mí, por creer en mí y por brindarme el espacio y tiempo que necesitaba para enfocarme en este proyecto.

A mi polola Javiera, quiero agradecerle por estar conmigo en cada paso de este camino. Su amor y apoyo incondicional han sido fundamentales para alcanzar este logro.

A mis colegas, especialmente a Alfredo y Aldo, quiero agradecerles por brindarme su ayuda y enseñanza a lo largo de este proceso. Su guía y paciencia han sido invaluables para mí.

Nuevamente, quiero agradecer a todos los que han formado parte de este proyecto y espero que esto permita simplificar y hacernos disfrutar un poco más la vida.

Un fuerte abrazo para todos,

Joaquín.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Antecedentes Generales

El presente trabajo se desarrolló en una empresa dedicada al sector de *retail* y servicios, enfocada en brindar valor a lo largo de todo el proceso de compra del consumidor. Sus unidades de negocio proveen diversos servicios que incluyen: comercio tanto físico como *online* a través de un *marketplace*, financiamiento de personas y empresas, distribución logística de primera y ultima milla, entre otros.

Se trabajó en el área corporativa de la empresa, área de negocio que busca generar mejoras transversales a todas las otras unidades de negocio. En específico, en la gerencia de *Advanced Analytics*, para el equipo de *Supply Chain*, que busca mejorar diversos procesos en la cadena de suministro al utilizar técnicas de analítica avanzada.

La cadena de suministro para el *e-commerce* de la empresa está compuesta de forma simplificada de las siguientes áreas:

- Primera Milla: corresponde a los flujos de paquetes desde los proveedores y los vendedores a los distintos centros de distribución.
- Flota Troncal: se encargan de mover paquetes entre centros de distribución. Esta se presenta principalmente cuando se busca enviar un paquete a regiones.
- Ultima Milla: corresponde al último tramo del recorrido, desde el centro de distribución final hasta el domicilio del cliente.

En primera milla, se busca satisfacer los movimientos de productos desde los distintos proveedores o vendedores del *marketplace* al centro de distribución respectivo. Para esto, se tiene una flota heterogénea de camiones dedicada, con camiones de baja volumetría como también ramplas de más de 100 m3.

Hay dos centros de distribución principales: el de *Mini Ticket*, que está especializado en recibir paquetes de tamaños pequeños, y el de *Big Ticket*, que recibe paquetes de tamaños grandes. Esta diferenciación es importante porque ambos centros están en zonas geográficas distintas, por lo que los camiones de ambos flujos no se pueden mezclar. De igual manera, las rutas realizadas por los camiones de *Big Ticket* generalmente se ven limitados por el espacio disponible ocupado por los productos, mientras que los de *Mini Ticket* por el tiempo de ruta

total.

Asimismo, la empresa ha estado impulsando en el último tiempo el servicio de *Same Day Delivery*, lo que implica que los camiones deben llegar al centro de distribución antes de una hora acordada para permitir el procesamiento de los productos y realizar la entrega final en el mismo día. Este horario límite se convierte en una restricción crítica para la planificación de las rutas de camiones y se debe considerar cuidadosamente para garantizar la eficiencia del sistema logístico.

Un recorrido de primera milla se puede resumir en los siguientes etapas:

- Viaje de colocación: corresponde al recorrido entre el punto de partida del transportista y el primer proveedor a recolectar.
- Recolección de productos de proveedores: esta etapa se subdivide en varias sub-etapas, que se repiten para cada proveedor a recolectar:
  - Traslado entre proveedores: corresponde al viaje que realiza el transportista entre los proveedores.
  - Recolección en proveedor: hace referencia al acto de retiro, marcado y carga de los productos al camión. En la industria al tiempo asociado que toma al transportista retirar los productos se denomina el *Service Time* o tiempo de servicio.
- Traslado hacia el centro de distribución: una vez que se han recolectado todos los productos, el transportista se dirige hacia el centro de distribución para descargar los productos.

1.2. Identificación y Formulación del Problema

A la hora de planificar las distintas rutas de camiones, se tiene la oportunidad de mejorar la configuración de la plataforma corporativa de ruteo para la primera milla, al poder proponer mejoras basadas en tecnologías de analítica avanzada. Esto se debe a la necesidad por parte de la empresa de volver más eficiente la operación y bajar los costos asociados a la implementación del servicio de *Same Day Delivery*. Proceso que inherentemente aumenta los costos cuando se deben movilizar más recursos con el fin de disminuir los tiempos de entrega.

Por lo anterior, se busca disminuir los costos al incorporar algoritmos de ruteo que generen una asignación de las distintas paradas a lo largo de cada ruta de la manera más eficiente posible, considerando las distintas restricciones de la realidad. Sin embargo, para calcular esta asignación de paradas se debe tener información fidedigna respecto a distintas características del problema de ruteo a resolver. Ejemplos de estos datos incluyen: obtener estimados fidedignos del volumen ocupado en cada camión, incorporar ventanas horarias de funcionamiento para los puntos a retirar, así como también mejorar la estimación del factor de tráfico.

Asimismo, el aspecto más relevante en términos de información es obtener una estimación de los tiempos de ruta de cada camión, ya que para planificar rutas lo más certeras posibles se debe contar no sólo con un buen estimado del tiempo que el camión se demora en hacer el recorrido, sino también del tiempo que toma en retirar los productos de cada vendedor. Siendo este último, el tiempo de servicio, o en inglés *Service Time*. Actualmente, tanto en la

plataforma corporativa, como generalmente en la industria, se utiliza un tiempo de servicio estimado estático basado en el criterio de una persona experta en la operación [1].

Cuando se logra generar estimaciones del tiempo de servicio que se ajustan dinámicamente para obtener el menor error acumulado posible, se puede mejorar significativamente la descuadratura entre la duración estimada y la realidad. Lo que se traduce en un mejor cumplimiento de los tiempos de entrega, generando rutas más confiables y eficientes, lo cual a su vez reduce la necesidad de generar rutas manuales por parte de las personas que realizan la planificación, denominados *planner*.

Tomando en cuenta las condiciones anteriores, se puede inferir que desde la planificación se busca aprovechar estas nuevas tecnologías con el fin de generar rutas más eficientes y resilientes a errores. No obstante, por motivos de continuidad operacional, no se puede modificar directamente la plataforma corporativa de ruteo, por lo que se busca desarrollar una herramienta que resuelva el problema de ruteo de vehículos. Esta servirá como prueba de concepto para la incorporación de estas nuevas variables en la otra plataforma.

1.3. Objetivos del proyecto

Con lo anterior expuesto, se tiene como objetivo general de este trabajo, el *optimizar la planificación del transporte de primera milla al desarrollar una herramienta que permita resolver el problema de ruteo de vehículos*. Mejoras en la eficiencia en la planificación de esta etapa, con la restricción de no afectar la oferta, puede tener un impacto significativo en la rentabilidad e indicadores de servicio del proceso de primera milla.

Para lograr este objetivo general, se han planteado tres objetivos específicos:

1. Mejorar el proceso de ruteo, incorporando algoritmos que sistematicen la asignación de paquetes a cada camión de manera óptima, reduciendo el número de camiones necesarios.
2. Generar la estimación faltante de la operación asociada al *Service Time*, mediante el desarrollo de una metodología que permita estimar el tiempo de servicio que le tomará a un camión retirar los paquetes de un punto determinado.
3. Generar los datos necesarios para asegurar que el algoritmo pueda cumplir con las restricciones reales de tiempo y capacidad de los vehículos disponibles, permitiendo habilitar la planificación de rutas operacionalmente factibles.

Capítulo 2

Marco Teórico y Estado del Arte

2.1. Definición del Problema y Notación Básica

El problema en el cual se tiene que lidiar con la distribución de bienes entre un centro de distribución y distintos clientes, se denomina el problema de ruteo de vehículos, o en inglés el *Vehicle Routing Problem* [2]. En este, se busca modelar todas las distintas variables que estén relacionadas con el ruteo de los vehículos a los distintos clientes de tal forma que se pueda hacer un aproximado a lo que sería una potencial realidad y, al considerar esos datos, poder generar rutas de vehículos factibles dadas las características del problema.

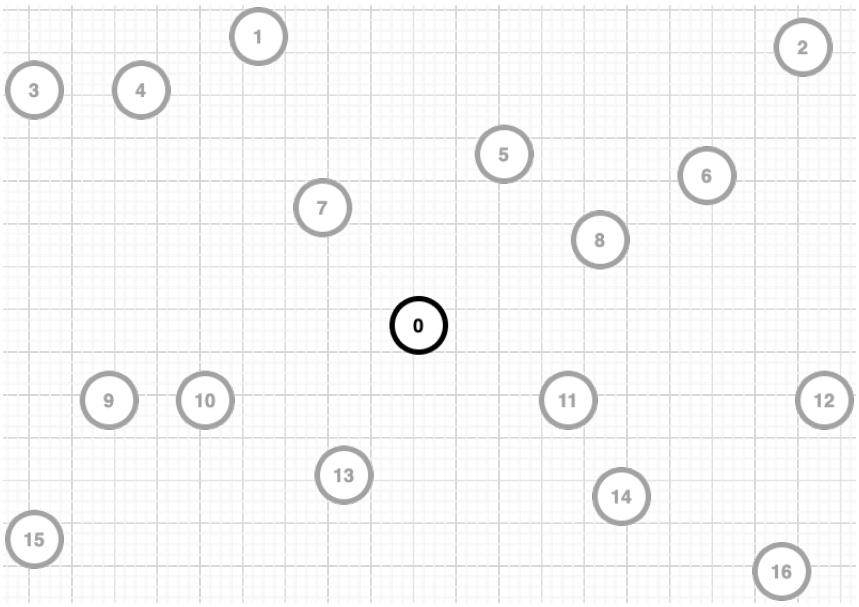

Figura 2.1: Red de nodos de ejemplo

2.2. Traveling Salesman Problem

Existen múltiples variaciones del *Vehicle Routing Problem*, el *Travelling Salesman Problem* corresponde la versión más básica. Esta consiste en un único viaje y, busca simplemente generar el camino para una red de nodos que minimice la distancia o tiempo total del recorrido. Para eso, el *Travelling Salesman* recorre todos los nodos una sola sola vez y, termina su

recorrido de vuelta en el nodo inicial. En la Figura 2.2 se puede apreciar un ejemplo de uno de estos recorridos, que completa un Ciclo Hamiltoniano. Detalles de notación se comentarán más adelante.

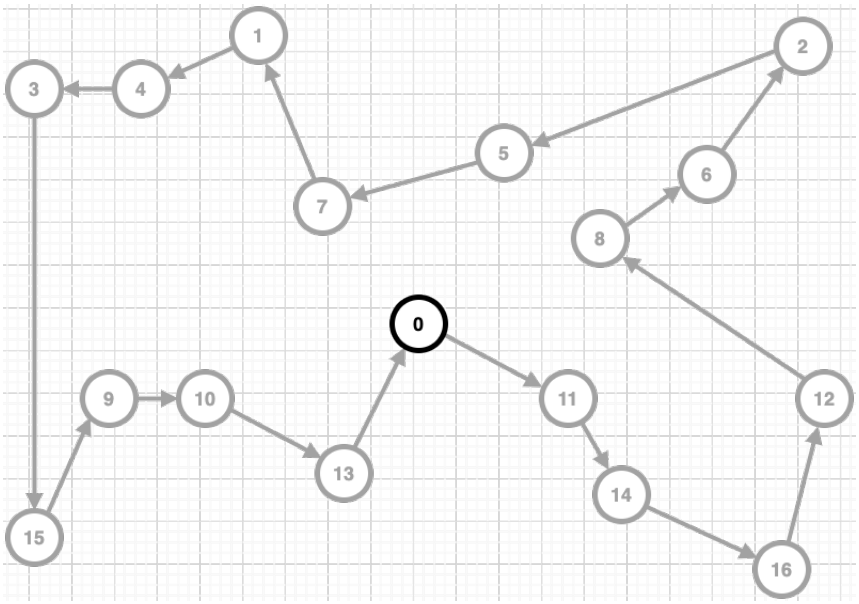

Figura 2.2: Solución del *Travelling Salesman Problem* para Figura 2.1

Por definición, este problema es *NP-Hard*, lo que significa que no existe un algoritmo exacto que pueda resolver el problema sin utilizar fuerza bruta para obtener la combinación que genere el mínimo global [3]. Asimismo, utilizar fuerza bruta se vuelve impráctico a medida que crece el número de nodos, puesto que la cantidad de posibles recorridos denotado con r crece factorialmente, siguiendo la siguiente ecuación:

$$r = \frac{(n-1)!}{2} \tag{2.1}$$

De la ecuación 2.1 se puede calcular la tabla 2.1 para un número n de nodos y, posible tiempo computacional estimado suponiendo que calcular un ciclo toma 1 micro segundo.

Tabla 2.1: Tiempos de cálculo estimados para generar todas las combinaciones posibles de ciclos cerrados para distinta cantidad de nodos.

n	**r**	**Tiempo cálculo**
3	1	1 μs
5	12	12 μs
8	2520	2.5 ms
10	180440	180 ms
20	$6.08x10^{16}$	1926 años
100	$4.66x10^{155}$	$1.47x10^{142}$ años

Existe un área extensa de estudio para buscar soluciones lo más cercanas posibles a este mínimo global en un tiempo razonable para problemas con números variados de nodos. Esta área estudia tanto Teoría de Grafos, como también distintas heurísticas y metaheurísticas que

pueden generar soluciones razonables en tiempos bajos [4]. Para seguir profundizando en el estudio de la resolución del *Travelling Salesman Problem*, es necesario hacer una introducción de Teoría de Grafos.

2.3. Teoría de Grafos

Para empezar a trabajar con grafos, es necesario establecer ciertas definiciones basadas en el paper de Brucato [3]:

Definición 1 (Grafo simple) Un grafo simple, G = (V,E) es un conjunto de objetos V llamados vértices o nodos, en conjunto con un posible conjunto de E subconjuntos de dos elementos V llamados arcos o ejes. Los nodos y arcos pueden tener pesos, denominados w, que corresponden a alguna característica virtual o física que posean.

Definición 2 (Adyacentes) Dos nodos son adyacentes si están conectados por un arco.

Definición 3 (Incidentes) El nodo u y el arco uv son incidentes entre ellos. Ocurre lo mismo para el nodo v con el arco uv.

Definición 4 (Grado de un nodo) El grado de un nodo v, es el número de vértices en G que son adyacentes a v.

Definición 5 (Mínimo y máximo grado de un grafo) El máximo grado de un grafo G, es el grado mayor de todos los nodos en G. Igualmente, el grado mínimo de un grafo G, es el grado menor de todos los nodos en G.

Definición 6 (Tamaño y orden de un grafo) El número de nodos en un grafo G, denominado $|V(G)|$, es llamado el orden de G y, se denota Ord(G). El número de arcos, denotado $|E(G)|$, se llama el tamaño de G.

Definición 7 (Grafo completo) Un grafo en el que todo par de nodos es adyacentes se llama un grafo completo.

Definición 8 (Paseo) Para dos nodos, no necesariamente distintos u y v, en un grafo G, un paseo u $\rightarrow$ v denominado W en G es una secuencia de vértices en G, empezando en u y, terminando en v.

Definición 9 (Paseo cerrado) Es un paseo en el cual los nodos iniciales y finales son los mismos.

Definición 10 (Camino) Es un paseo en un grafo G en el que no se repite ningún nodo.

Definición 11 (Ciclo) Un camino con nodos inicial y final iguales es llamado un ciclo.

Definición 12 (Envoltura convexa) Un grafo G tiene una envoltura convexa cuando se tiene un ciclo que contiene todos los nodos al interior del grafo, sin necesariamente ser estos nodos pertenecientes al ciclo.

Definición 13 (Ciclo Hamiltoniano) Corresponde al caso en el que un ciclo contiene cada nodo de un grafo G.

2.4. Heurísticas de Resolución

Considerando las definiciones anteriores, el *Travelling Salesman Problem* puede ser representado como un grafo $G = (V, E)$ donde cada destino, incluyendo su casa es un nodo y, si existe una ruta directa que conecta dos destinos distintos, entonces hay un arco entre esos dos nodos. El problema del vendedor ambulante se resuelve si existe una ruta, representada por un Ciclo Hamiltoniano, que sea la más corta, en donde se visita cada destino una vez y, regresa al nodo inicial.

Como se había mencionado anteriormente, no existe un método exacto que encuentre el Ciclo Hamiltoniano con soluciones globalmente óptimas, pero sí existen heurísticas que permiten generar soluciones que son óptimas para una vecindad cercana y, que de por sí son una solución factible generando rutas que llegan a un óptimo local.

En la siguiente sección, se discutirá respecto a distintas heurísticas que se han utilizado ampliamente durante las últimas décadas para resolver el *Travelling Salesman Problem* y, su extensión, el *Vehicle Routing Problem*. Para que el lector pueda entender y tener una visión general de cómo funcionan estas heurísticas, se explorarán, en una primera instancia, heurísticas que generan primeras soluciones en rutas, posteriormente, heurísticas que iteran por sobre estas primeras soluciones y, finalmente técnicas que permiten generar exploraciones más generales del espacio posible de búsqueda, evitando entrampamientos en óptimos locales.

2.4.1. Algoritmo del Vecino más Cercano

El Algoritmo del vecino más cercano, o el *Nearest-Neighbor Algorithm* en inglés, es un tipo de algoritmo generativo de tipo *greedy*, lo que significa que, va generando una ruta al elegir en cada iteración el arco correspondiente con menor peso. Hay que tener en consideración que la salida del algoritmo puede variar dependiendo del nodo inicial seleccionado, donde dos nodos distintos iniciales podrían dar como resultado dos Ciclos Hamiltonianos completamente distintos.

Algorithm 1 Algoritmo del vecino más cercano

(1) Inicializar todos los nodos como no visitados.
(2) Seleccionar un nodo arbitrario, defínase como el nodo actual u.
(3) El nodo u se marca como visitado.
(4) Encontrar el arco más corto que conecta el nodo actual u con alguno de los nodos sin visitar v.
(5) Seleccionar nodo v como nodo actual u.
(6) Cuando todos los nodos del dominio son visitados, se retorna la lista de nodos visitados, en caso contrario, volver al paso (3)

En términos de complejidad computacional, este algoritmo 1 toma en resolver el problema una cantidad de pasos del orden de $O(n^2)$, un nivel computable para combinaciones de nodos extensas. Esto se debe a que debe recorrer todos los nodos y, revisar para cada iteración, los

valores de cada uno de las conexiones de los arcos.

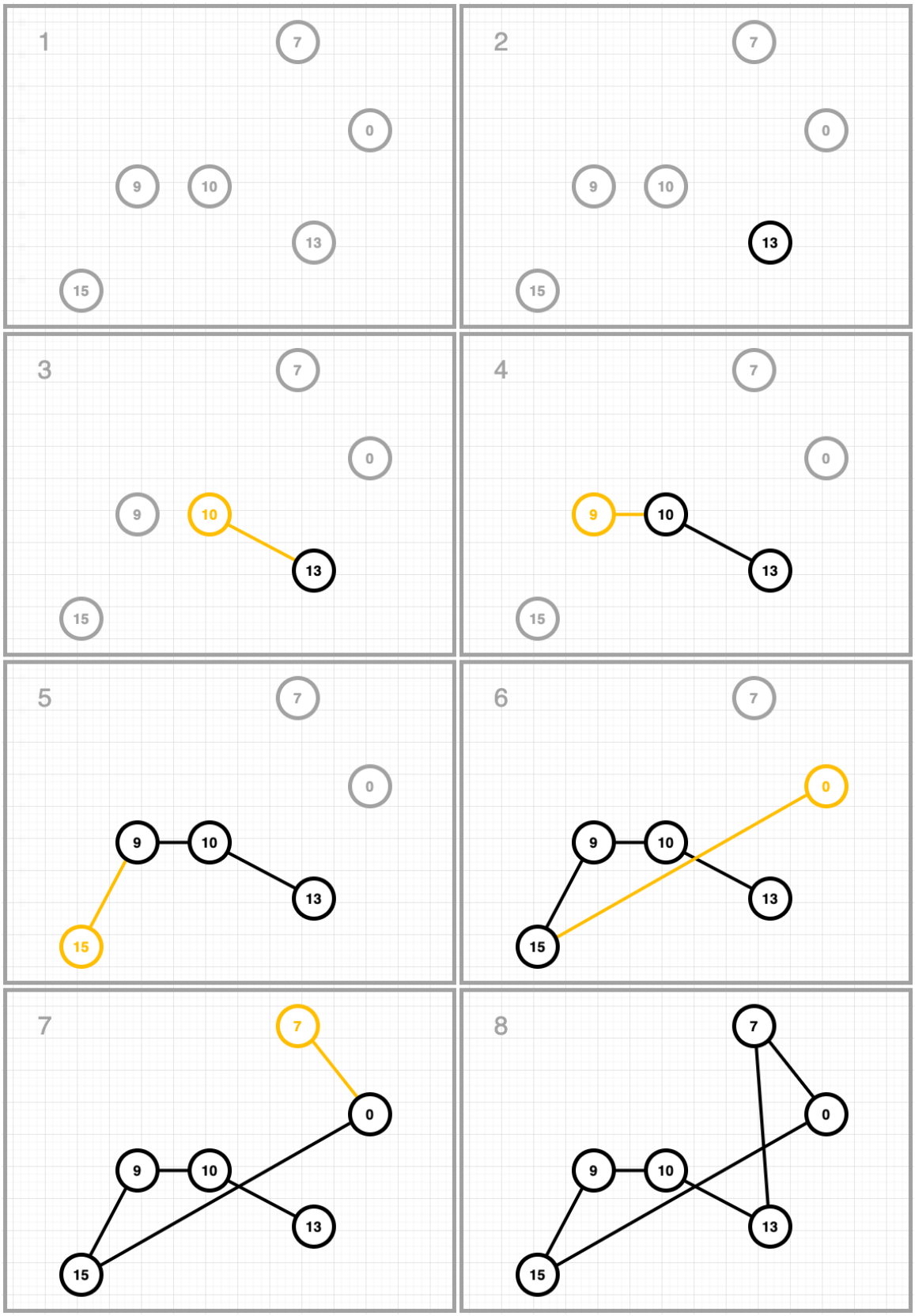

Figura 2.3: Algoritmo del vecino más cercano para caso ejemplo con 6 nodos.

Como se ve en la figura 2.3, se inicializa aleatoriamente en el nodo 13 y el algoritmo va generando conexiones en cada iteración con el vecino más cercano. Cuando se logra generar el ciclo hamiltoniano, se aprecia en una primera instancia que la solución generada no es óptima globalmente. Ejemplos de posibles mejoras se aprecian entre los nodos 0, 13 y 15, donde una ruta que recorra esos nodos y, otra conexión entre el nodo 7 y 10, darían como resultado el caso óptimo.

Es por lo narrado en el caso anterior, que existen otras heurísticas de búsqueda local que buscan mejorar la primera solución generada, al explorar la posibilidad de realizar intercambios de arcos entre nodos. Existe una multitud de algoritmos que iteran por sobre la primera solución, pero destacan los que hacen permutaciones de k nodos, como son el caso del 2-opt y el 3-opt.

2.4.1.1. 2-opt

El algoritmo 2-opt es un algoritmo de búsqueda local propuesto por Flood en 1956 [5]. La idea principal de este es tomar pares de arcos que no compartan un mismo nodo, eliminarlos y cruzar las rutas de los nodos expuestos, tal y como se muestra en la figura 2.4. Si la función objetivo resultante de la operación disminuye, se toma esta nueva combinación como válida. En términos de complejidad temporal, cada ejecución única de este algoritmo tiene un orden de $O(n^2)$ [6].

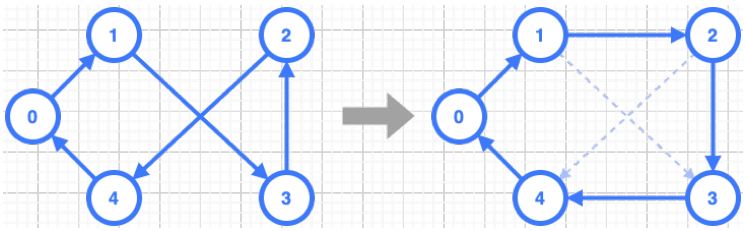

Figura 2.4: Ejemplo de permutación 2-opt.

2.4.1.2. 3-opt

Similarmente al caso anterior, 3-opt es un algoritmo de búsqueda local que, a diferencia del caso anterior, involucra eliminar 3 arcos y, generar 3 nuevas conexiones de arcos. Con esto, existen 7 formas distintas de generar las nuevas conexiones, en caso de haber una disminución en la función objetivo, se utiliza la que genera la mayor disminución. La figura 2.5 muestra la permutación desde una de las posibles combinaciones hacia un escenario con una menor función objetivo percibida. Una ejecución única del algoritmo tiene una complejidad temporal de $O(n^3)$ [6].

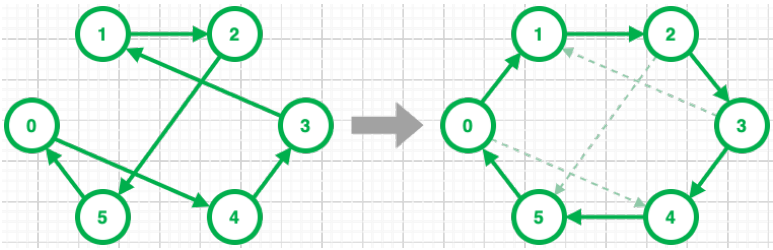

Figura 2.5: Ejemplo de permutación 3-opt, donde se eliminan 3 arcos y se generan 3 conexiones nuevas, obteniendo una ruta con menor distancia aparente.

Esta permutación se puede utilizar en conjunto con búsquedas generativas para mejorar el resultado inicial, tal y como se muestra en la figura 2.6.

2.4.2. Greedy Local Search

La búsqueda local *greedy*, o codiciosa, es una técnica de optimización heurística que se utiliza para encontrar soluciones buenas y casi óptimas a problemas difíciles. Esta combina métodos generativos como el *Nearest Neighbours* con métodos que hacen *swap* de nodos como es el caso de 2-opt y 3-opt. Al igual que en los casos anteriores, es un algoritmo de búsqueda local, lo que significa que comienza con una solución inicial y luego la mejora, iterativamente, haciendo pequeños cambios locales.

La idea clave detrás de la búsqueda local codiciosa es tomar siempre la mejor decisión en cada paso, según el estado actual de la solución. Esto significa que el algoritmo selecciona el movimiento que parece ser el mejor en ese momento, sin considerar las consecuencias a largo plazo de la decisión. Esto puede conducir a soluciones buenas, casi óptimas, pero también puede quedarse atascado en los óptimos locales [7], [8], donde no es posible seguir mejorando sin hacer un movimiento subóptimo.

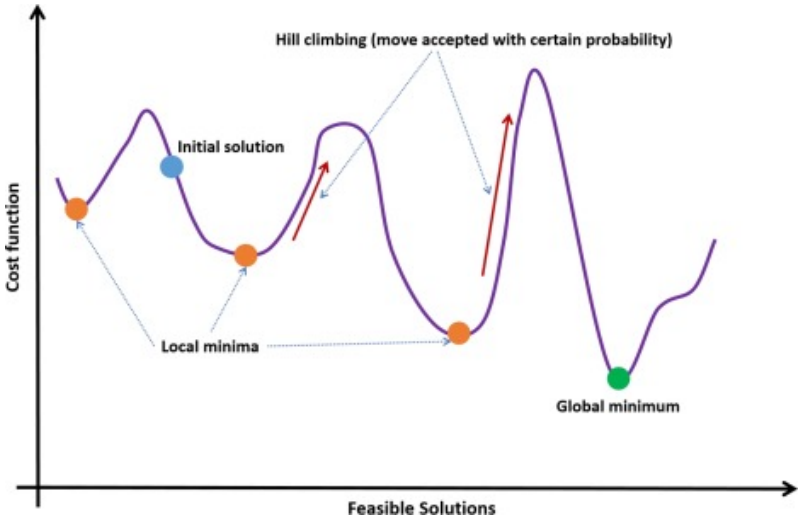

Figura 2.7: Función objetivo respecto al posible espacio de búsqueda. Obtenido de [7].

Para superar este problema, los algoritmos de búsqueda locales codiciosos a menudo usan

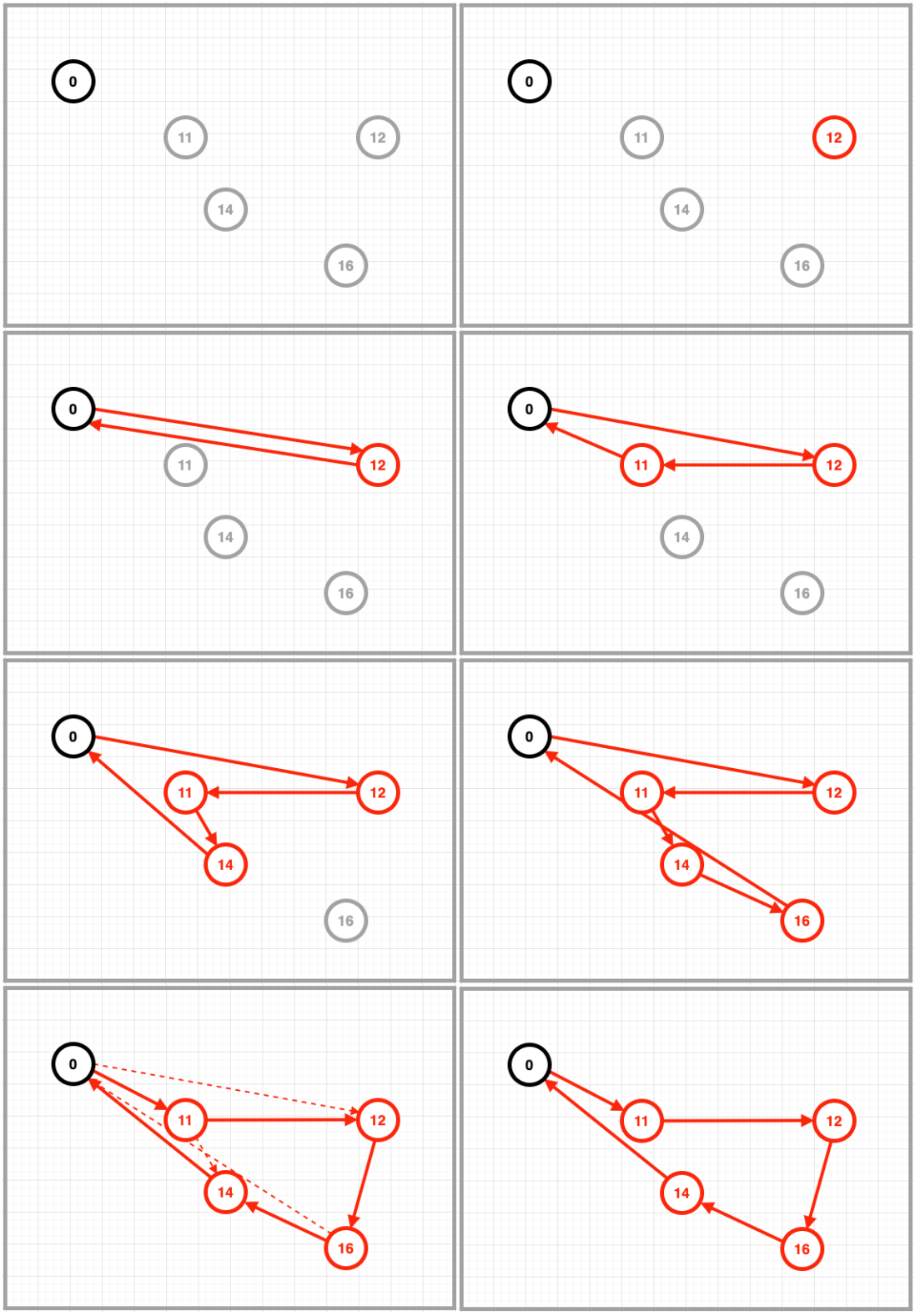

Figura 2.6: Ejemplo de búsqueda local *greedy*, con una primera iteración generativa basada en un algoritmo similar al de *Nearest Neighbours* y usando permutaciones 3-opt que se mejora la primera solución hasta llegar al óptimo global.

una variedad de técnicas, como la aleatorización, la perturbación y la diversificación, para escapar de los óptimos locales y explorar el espacio de búsqueda más extensamente. Estas técnicas pueden ayudar al algoritmo a encontrar mejores soluciones, pero también aumentan el costo computacional y el riesgo de quedarse atascado en soluciones subóptimas. En la figura 2.7, se aprecia cómo expandir el espacio de búsqueda con soluciones que salgan del mínimo y puedan encontrar mejores soluciones.

2.4.3. Guided Local Search

La búsqueda local guiada (GLS, por sus siglas en inglés), es una metaheurística de optimización que modifica algoritmos de búsqueda local con penalizaciones y costos asociados a las distintas características del problema. Con esto, se explora una mayor región del espacio de búsqueda con mayor eficacia, navegando hacia regiones más prometedoras y posibilitando encontrar mejoras en la función objetivo. Este método fue desarrollado por Voudouris y Tsang y, la siguiente explicación del funcionamiento es basado en [8].

Para generar las penalizaciones, se define una función objetivo penalizada $h(s)$ que reemplaza a la función original $g(s)$

$$h(s) = g(s) + \lambda \sum_{i=i,f} p_i \cdot I_i(s) \tag{2.2}$$

Donde s es la solución candidata, λ es un parámetro del algoritmo GLS, f es el número de características, p_i corresponde a la penalización de la característica i, que se inicializa con 0 para todas las características e I_i, es una función indicatriz que indica si la solución s contiene a la característica i.

Cuando, durante el uso de esta metaheurística, se llega a un óptimo local, se incrementa la penalización de las características asociadas a esa solución. Esto cambia de cierta forma la función objetivo por sobre la cual se itera, incrementando el costo en la vecindad y, permitiendo al algoritmo de búsqueda local explorar nuevos segmentos del espacio de búsqueda, tal y como se aprecia en la figura 2.8.

La forma de generar esta penalización se realiza con una función guía, denominada la *utilidad de penalizar una característica*. Se define esta función $util_i$, sobre un óptimo local M como lo siguiente:

$$util_i(M) = I_i(M) \times \frac{c_i}{1 + p_i} \tag{2.3}$$

Donde c_i es el costo de la característica i. Con esto, si una característica no aparece en la solución que genera el óptimo local, entonces la utilidad de penalizarla es 0. Entre más alto es el costo de la característica, tiene mayor utilidad penalizarla. Por último, la mayor cantidad de veces que la característica ha sido usada, mayor se vuelve el denominador $(1 + p_i)$ y, por lo tanto, disminuye la utilidad de penalizarlo nuevamente.

En un óptimo local, la característica con mayor valor de utilidad es penalizada. Esto se genera al agregar 1 a su valor de penalización.

$$p_i = p_i + 1 \tag{2.4}$$

Existen muchos ejemplos en la literatura en el que se utiliza este algoritmo, como el desarrollado por Tsang y Voudouris para la asignación de trabajos de una compañía de

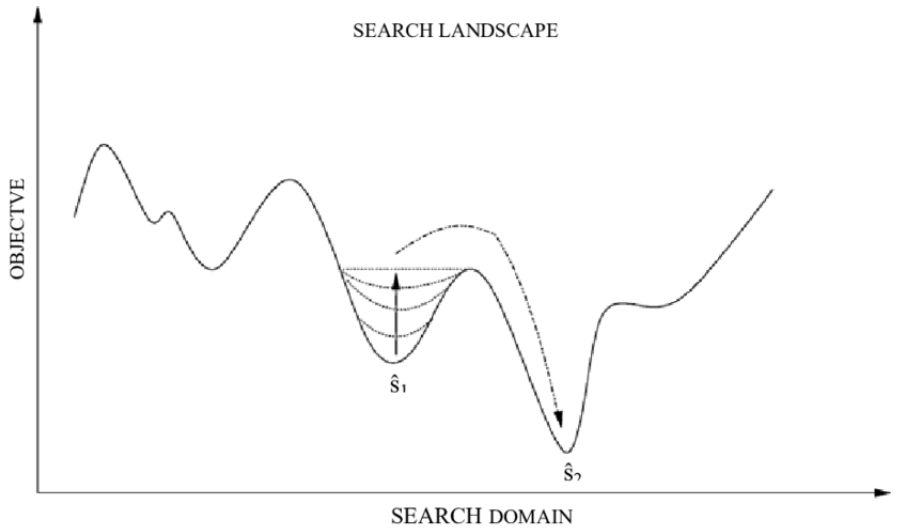

Figura 2.8: Cambios en función objetivo $h(s)$ basados en penalizaciones de características considerando cierto espacio de búsqueda. Obtenido de "Metaheuristics in combinatorial optimization: Overview and conceptual comparison". Blum, Roli. [9].

Algorithm 2 Algoritmo Guided Local Search

Input: una función objetivo g y, una estrategia de búsqueda local L.
(1) Generar una solución inicial heurísticamente o aleatoriamente.
(2) Inicializar los valores de penalización p_i en 0.
while Repetir hasta llegar a cierto número de iteraciones, o cumplir cierto requerimiento de finalización **do**
(3) Realizar búsqueda local de acuerdo a la función objetivo h, hasta encontrar un óptimo local M.
(4) Cuando se llega al óptimo local, calcular la utilidad de cada característica i.
(5) Penalizar todas las características con función de utilidad máxima.
end while
(6) Retornar la mejor solución calculada considerando la función objetivo g.

telecomunicaciones [10], en la que se utiliza el *Guided Local Search*, usando una función generativa basada en el *Fast Local Search*, una heurística similar al *Greedy Local Search*, en conjunto con permutaciones 2-opt para realizar exploraciones del espacio de búsqueda.

2.4.4. Tabu Search

La Búsqueda Tabú es un algoritmo de optimización metaheurística que utiliza un enfoque basado en recordar los movimientos realizados anteriormente para explorar el espacio de búsqueda y evitar quedarse atascado en los óptimos locales [11]. Funciona manteniendo una lista de soluciones prohibidas o tabú, que ayuda al algoritmo a evitar volver a estados visitados anteriormente.

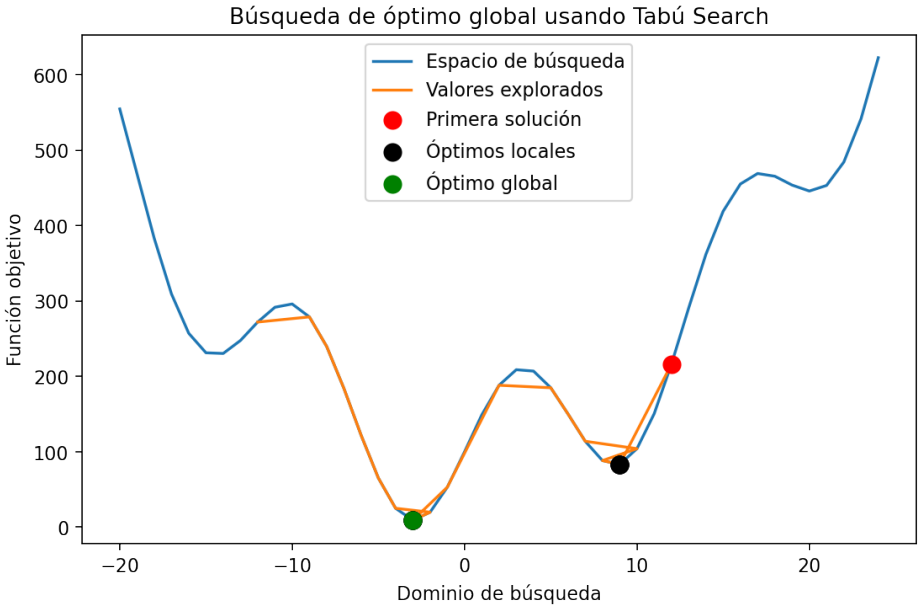

Figura 2.9: Búsqueda tabú realizando una exploración del espacio de búsqueda para ejemplo unidimensional basado en funciones sinusoidales sobre una parábola. Pese a iniciar en un punto arbitrario que podría quedar atrapado en un óptimo local, decide explorar la vecindad, encontrando un mínimo global.

La idea básica detrás de la Búsqueda Tabú es mejorar iterativamente una solución haciendo movimientos que son "tabú" o prohibidos, con el objetivo de escapar de los óptimos locales y explorar el espacio de búsqueda de manera más efectiva, tal como se observa en la figura 2.9. El algoritmo utiliza una función objetivo predefinida para evaluar la calidad de las soluciones candidatas y seleccionar la mejor para moverse.

Una de las características clave de la Búsqueda Tabú es el uso de técnicas de intensificación y diversificación para escapar de los óptimos locales y explorar el espacio de búsqueda de manera más efectiva. La intensificación se refiere a la aplicación de operadores de búsqueda más agresivos para mejorar la solución actual, mientras que la diversificación implica realizar movimientos más aleatorios para explorar diferentes partes del espacio de búsqueda.

Otro aspecto importante de la Búsqueda Tabú es el uso de una lista tabú, que ayuda al algoritmo a evitar volver a los estados visitados anteriormente y evita que se quede atascado en los óptimos locales. La lista tabú se actualiza dinámicamente a medida que avanza el algoritmo, con el objetivo de guiar la búsqueda hacia mejores regiones del espacio de búsqueda.

Algorithm 3 Algoritmo Búsqueda Tabú

```
Input: Se inicializa problema con heurística generativa o de forma aleatoria (s0).
sMejor ← s0
mejorCandidato ← s0
listaTabu ←
listaTabu.agregar(s0)
while !condicionTérmino() do
    sVecindario ← obtenerVecindario(mejorCandidato)
    mejorCandidato ← None
    for sCandidato en sVecindario do
        if sCandidato no en listaTabu then
            if g(sCandidato) < g(mejorCandidato) then
                mejorCandidato ← sCandidato
            end if
        end if
    end for
    if g(sCandidato) < g(sMejor) then
        sMejor ← sCandidato
    end if
    listaTabu.agregar(sCandidato)
    if listaTabu.tamaño > tamañoTabuMáximo then
        listaTabu.pop()
    end if
end while
```

2.4.5. Simulated Anneaning

El *Simulated Anneaning* es un algoritmo de optimización metaheurística que se utiliza para encontrar el óptimo global de una función. Se basa en los principios del templado en metalurgia, que consiste en calentar y luego enfriar lentamente un material para reducir sus defectos y aumentar su estabilidad estructural [12].

En este caso, el algoritmo funciona perturbando aleatoriamente la solución actual y evaluando la función objetivo en la nueva solución candidata. Generalmente, si la solución candidata es mejor que la solución actual, se acepta como la nueva solución actual. Sin embargo, si la solución candidata es peor que la solución actual, se puede aceptar con una probabilidad que depende de la diferencia en los valores de la función objetivo y la temperatura actual. Esto permite que el algoritmo escape de los óptimos locales y continúe buscando el óptimo global.

A medida que la temperatura se reduce gradualmente, es menos probable que el algoritmo acepte peores soluciones y eventualmente converge al óptimo global. Este proceso de enfriamiento gradual se conoce como programa de templado y es un parámetro importante del algoritmo. En el recuadro 4 se encuentra el algoritmo de *Simulated Annealing*:

Algorithm 4 Algoritmo de Simulated Annealing

(1) Establezca una temperatura inicial y un programa de templado.
(2) Establezca el estado inicial del sistema en una solución aleatoria.
(3) Evaluar el estado inicial utilizando la función objetivo.
(4) Genere una solución candidata aleatoria haciendo una pequeña perturbación al estado actual.
(5) Evalúe la solución candidata utilizando la función objetivo.
(6) Calcule la probabilidad de aceptación de la solución candidata.
(7.1) Si la solución candidata es mejor que el estado actual, acéptela como el nuevo estado actual.
(7.2) De lo contrario, acepte la solución candidata con una probabilidad igual a la probabilidad de aceptación.
(8) Si la temperatura sigue siendo alta, volver a (4), en caso contrario, entregar estado final del sistema.

2.5. Vehicle Routing Problem

Hasta esta sección, se ha escrito exclusivamente respecto a la problemática y forma de solucionar el problema de ruteo de vehículos en los casos donde un vehículo único pueda realizar una ruta única que cubra todos los nodos del mapa. Sin embargo, en la vida real, existen múltiples restricciones que impiden que un único vehículo pueda realizar una sola ruta [2], tal y como se observa en la figura 2.10.

Al igual que con el *Travelling Salesman Problem*, este problema tiene como complejidad mínima *NP-Hard*, por lo que no existe un algoritmo que lo resuelva que llegue siempre al óptimo global, salvo para el caso en el que se usa fuerza bruta, con costos temporales incrementales del orden factorial. Es por esto, que las heurísticas de resolución señaladas anteriormente también aplican, con ciertas modificaciones menores, a la resolución de este problema.

Existen múltiples variantes de este problema basado en las características o restricciones del problema, entre ellas destacan:

- Capacitivo: denotado por la letra C, genera como una limitante dura al problema de optimización la capacidad de los vehículos a la hora de hacer las rutas. Impidiendo a los vehículos que queden rebalsados en capacidad.
- Ventanas horarias: denotado por las siglas TW, limita las rutas a ciertas duraciones máximas, o ente distintos horarios, para todos los nodos.
- Flota Heterogénea: denotado por la letra H, ocurre cuando se tiene un problema capacitivo y, al mismo tiempo, las capacidades de la flota no son homogéneas.

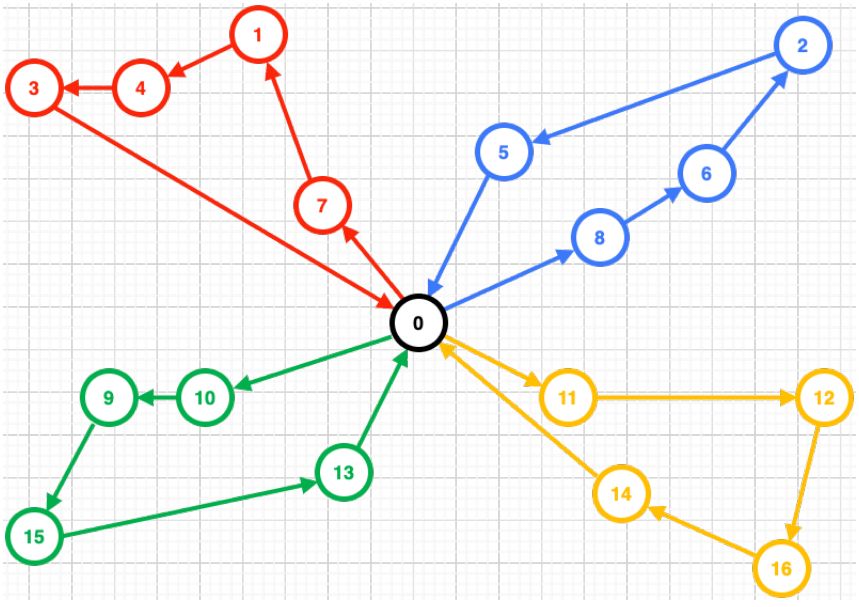

Figura 2.10: Figura de ejemplo donde se muestran 4 rutas distintas dadas las restricciones del problema.

2.5.1. Restricciones de Capacidad

Es necesario tener en cuenta las restricciones de capacidad, en caso de que los camiones se vean frecuentemente limitados por la cantidad máxima de bienes a cargar. Esto asegura que los vehículos se utilicen de manera eficiente y efectiva, evitando problemas operacionales ligadas al desbordamiento de los vehículos. Las restricciones de capacidad especifican la cantidad máxima de bienes o materiales que puede transportar cada vehículo y juegan un papel crucial en el VRP porque determinan la viabilidad y eficiencia de las rutas.

Debido a la naturaleza incierta de los datos asociados a la capacidad de los bienes a cargar, su posible variabilidad y, factores de empaquetamiento, la capacidad máxima utilizable de los vehículos es menor a la capacidad máxima real, por lo que se tiende a utilizar un factor porcentual máximo de carga.

2.5.2. Ventanas Horarias

Las ventanas horarias son inherentes al funcionamiento de tanto los distintos proveedores, como de los centros de distribución. Como es natural en la mayor parte de los rubros, las ubicaciones tienen ciertos horarios acotados en los que se puede visitar y, estando fuera de horario, el camión no puede retirar o descargar los paquetes de esa ubicación.

Por lo anterior, es necesario tener una buena estimación de tiempo para satisfacer las restricciones horarias y asegurar que las rutas sean eficientes y factibles. Si las estimaciones de tiempo son precisas, es posible programar las visitas de manera que satisfaga las ventanas de tiempo y minimice la distancia total o el costo de las rutas. Esto puede ayudar a evitar retrasos y tiempo de inactividad y, puede garantizar que los vehículos se utilicen de manera eficiente y eficaz.

Generalmente, existen dos factores que determinan el estimado de tiempo total en ruta. Estos son el tiempo de recorrido y el tiempo de servicio.

2.5.2.1. Tiempos de Recorrido

El tiempo de recorrido corresponde al tiempo que le toma al camión en realizar el trayecto entre varios puntos, considerando únicamente el tiempo empleado en el desplazamiento del vehículo. Estimar estos tiempos para una ruta puede ser un proceso complejo y hay muchos factores que pueden afectar el tiempo que lleva viajar de un lugar a otro. Algunos de estos factores incluyen: la distancia de la ruta, el modo de transporte, la velocidad del transporte, el número de paradas o transbordos y las condiciones del tráfico.

Una forma de estimar los tiempos de viaje es utilizar un servicio que puede proporcionar un tiempo estimado basado en la distancia de la ruta, la velocidad y aceleración promedio asociadas a un vehículo y las condiciones de tráfico locales.

Otra forma de estimar los tiempos de viaje es usar datos históricos sobre los tiempos de viaje para rutas iguales o similares. Estos datos se pueden utilizar para proporcionar una estimación del tiempo de viaje para una ruta determinada.

En ambos casos, es importante tener en cuenta que el tiempo de recorrido estimado es una estimación, factores aleatorios como accidentes, tráfico, o incluso el desfase temporal en semáforos pueden causar que el tiempo real varíe. Por lo que generalmente es una buena idea

permitir tiempo adicional al planificar un viaje.

2.5.2.2. Tiempo de Servicio

El tiempo de servicio es la cantidad de tiempo que le toma a un transportista en una parada recolectar los paquetes de un proveedor determinado. Es esencial calcular correctamente este tiempo, puesto que una mala estimación puede cambiar el horario de término de una ruta significativamente, quedando, al subestimar, con atrasos en la llegada al centro de distribución y, al sobreestimar, con capacidad ociosa.

La forma más usual de realizar estimaciones de tiempo de servicio, es usar una predicción que es generalmente entregada de forma manual por un *planner*. El *planner* generalmente se basa en experiencias pasadas que considera factores como la ubicación, la cantidad y tipo de paquetes a cargar. Como menciona Song et al. [1], *tal práctica requiere mucho tiempo y las estimaciones también suelen ser inexactas, por lo que generalmente no se realiza. Esto es de esperar ya que los planificadores tienden a ser más conservadores en sus predicciones, ya que los trabajos perdidos se penalizan fuertemente.*

Existen formas de estimar los tiempos de servicio utilizando soluciones basadas en datos que automatizan la predicción del tiempo de servicio. Un ejemplo de esta, es la desarrollada por Song et al. [1], que propone un método que utiliza datos de GPS históricos y, al cruzarlos con información de las rutas, se puede realizar una reconstrucción temporal de los distintos eventos. Con información de los tiempos de servicio históricos y, diversas *features* asociadas, se puede entrenar un modelo regresivo que genere predicciones de tiempo de servicio para nuevas rutas.

2.5.3. Desbalance de Rutas

Además de cumplir con las distintas restricciones consideradas en la literatura, existen objetivos que son diseñados para suavizar las diferencias entre las rutas. Estos objetivos se introducen con el fin de incorporar un elemento de rutas justas en efecto. Para definir un objetivo que balancee rutas, es necesario definir una métrica que defina la carga de una ruta. Ejemplos de estas métricas pueden ser, la distancia, tiempo, o cantidad de paradas de cada ruta. En este caso en particular, se tiene en consideración el desbalance de cantidad de puntos entre rutas.

Equilibrar rutas en una red de transporte o logística puede tener una serie de beneficios. El principal es que puede ayudar a mejorar la confiabilidad y la consistencia del proceso de transporte [13]. Al garantizar que las rutas se planifiquen y ejecuten de manera equilibrada, es posible evitar la congestión y otros problemas que pueden causar demoras e interrupciones.

En la literatura existen múltiples técnicas para propiciar el balance de rutas. Entre ellas destacan la penalización sobre la función objetivo de rutas desbalanceadas [14], en donde se toma la diferencia entre el valor esperado de puntos a retirar, entre las distancias recorridas o tiempo total y, en base a esto, se agrega una penalización proporcional a la función objetivo. También existe el caso de funciones objetivo multivariables [13], en donde, usando métodos de generación de columnas, se prueban múltiples combinaciones de rutas que cumpla tanto la función objetivo principal como, adicionalmente, el requerimiento de rutas balanceadas.

Capítulo 3

Diseño de la Solución

3.1. Propuesta de Solución

Basado en los requerimientos desde la planificación mencionados previamente en la introducción, el siguiente trabajo de memoria se va a estructurar en dos etapas principales.

La primera, está asociada a la definición del problema, al identificar todos los vehículos, sus capacidades, las ubicaciones de los clientes y las ventanas de tiempo en las que se debe realizar el retiro de cada cliente. Adicionalmente, se debe generar una estimación de los tiempos y distancias para cada par de clientes y una estimación del tiempo de servicio.

Como segunda etapa, se busca crear un modelo matemático que defina la función objetivo y las distintas restricciones. En este caso, se busca minimizar el tiempo total en ruta, teniendo en cuenta como restricciones principales no sobrepasar las capacidades de los vehículos y las ventanas de horarias para cada retiro. Para resolverlo, se utilizan distintas metaheurísticas disponibilizadas a través de la librería de Python *Google OR-Tools* [15].

Todo el proceso anterior, se encapsula en una página web que permite a los *planner*, colaboradores que planifican rutas de camiones, calcular las rutas utilizando el modelo. Los detalles de esta interfaz no se describirán en el presente trabajo, pero sí los datos generados por su uso.

3.2. Generación de Datos

3.2.1. Datos de Entrada

Desde la operación, se entregan los siguientes atributos que nos permiten modelar y resolver el problema:

- Dirección, comuna y región de proveedor.
- Coordenadas del Centro de Distribución.
- Cantidad y volumen de productos a cargar.
- Capacidad de los camiones.

- Ventanas horarias para visitar tanto a los proveedores como al Centro de Distribución.

Estos atributos son entregados en un archivo Excel de forma diaria, que debe ser ingresado y procesado por por la herramienta desarrollada en este trabajo.

3.2.2. Geocodificación de Direcciones

El proceso de geocodificación corresponde a la conversión desde una dirección física, como la dirección de una calle, o el nombre de un lugar, a las coordenadas geográficas correspondientes. Esto es particularmente útil, porque es más fácil trabajar usando las coordenadas geográficas que con direcciones físicas, puesto que pueden ser usadas de forma más precisa para ubicar locaciones en un mapa. Es más, para el proceso posterior de cálculo de tiempos y distancias, es necesario conocer la ubicación exacta de los distintos proveedores, por ende, se requiere geocodificar las direcciones correctamente.

Como cada proveedor tiene una dirección asociada, con comuna y región, se puede usar la API de Google para calcular sus coordenadas asociadas. Para esto, se desarrolló una librería denominada *easygeo*, que permite mandar *requests* a la API de Google, con los siguientes parámetros.

- Dirección completa: está compuesta por la concatenación de los elementos de una dirección (Calle, Número, Comuna, Región, País).
- API Key: corresponde a la llave autentificadora para poder obtener una respuesta válida. Esta es necesaria porque el servicio de geocodificación de Google es un servicio pago, que cobra por *request*.

Para aumentar las chances de que la API de Google encuentre las coordenadas para la dirección, se implementa una lógica de pre-procesamiento de direcciones, que elimina partes del texto que no aportan información importante, y la concatena con otros datos descriptivos como la comuna y la región.

Al mandar un *request* al servicio de Google, si encuentra una dirección válida, obtenemos de vuelta un *response* que contiene la siguiente información:

- Latitud.
- Longitud.
- Dirección asociada a geocodificación.

Para asegurar que los contenidos del *response* sean válidos, y se tenga la ubicación de las coordenadas geográficas exactas, se implementa una lógica que analiza la dirección asociada a las coordenadas geocodificadas y se buscan los siguientes atributos:

- Número de calle: puesto que se busca que el *response* contenga una dirección georeferenciada de forma exacta, y no sea, por ejemplo, el nombre de una calle.
- Estar en la Región Metropolitana: este atributo está asociado a una de las características del problema. Todas las rutas se realizan en la Región Metropolitana, por lo tanto, si se obtiene una geocodificación en otra región, aunque fuese exacta, es inválida.

Si alguno de los dos atributos no se encuentra, entonces la *response* es catalogada como inválida, por lo tanto, se pasa la dirección asociada a revisión manual. Esta revisión manual está incorporada como una vista en donde aparece la dirección original, un mapa del punto geocodificado incorrectamente, y su dirección asociada. Se le permite al *planner* realizar, en caso de que fuese necesario, correcciones a la dirección, o bien, validar que el punto está en una ubicación correcta.

3.2.3. Cálculo de Matriz de Tiempos y Distancias

Como se había mencionado anteriormente, para tener un control preciso de los tiempos de ruta, es necesario conocer los tiempos y distancias de traslado entre los distintos puntos, cumpliendo con las restricciones de ventanas horarias y de distancias máximas.

Esta estimación de tiempos y distancias se tiene que hacer para cada par de puntos con el fin de generar todas las posibles combinaciones de rutas, y con eso, permitir al algoritmo de optimización elegir las mejores rutas. Es con eso, que se generan una matriz de tránsito, en donde cada par de índices representa un origen-destino, y se computa una matriz por cada atributo de la ruta asociado a ambos puntos. En este caso, como se busca medir el tiempo y la distancia, se calculan dos matrices, respectivamente.

Para realizar la estimación de distancias, existen múltiples métodos, se puede calcular la distancia *haversine* entre cada par de puntos, y en base a esta distancia y una velocidad promedio, se puede estimar asimismo un tiempo asociado.

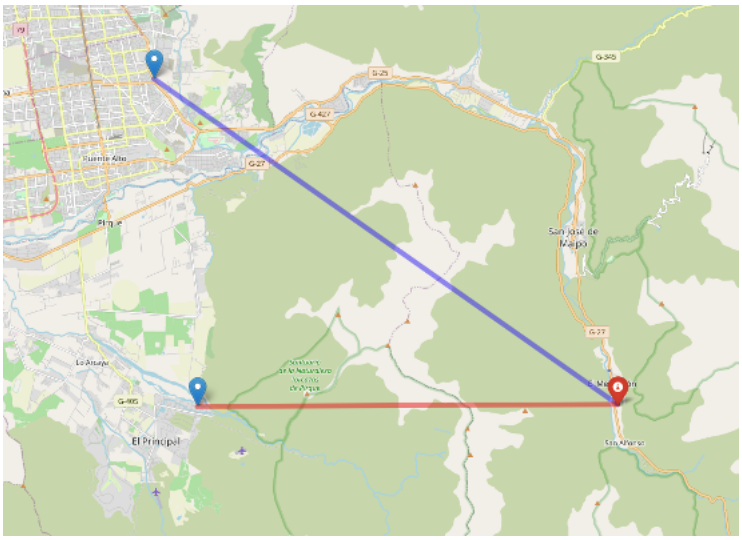

Figura 3.1: Comparación de punto con las dos ubicaciones más cercanas, basado en cálculos de distancia *haversine*. En rojo se aprecia la ruta más cercana bajo esta distancia.

Como la tierra puede ser modelada como una esfera, la distancia se puede calcular de la siguiente manera:

$$d = 2 * R * \sin^{-1} \sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1) * \cos(\varphi_2) * \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)} \tag{3.1}$$

Donde φ_1 y φ_2 son las latitudes de los dos puntos, λ_1 y λ_2 son las longitudes, R es el radio de la tierra, y d es la distancia entre los dos puntos.

Como se aprecia en la figura 3.1, el problema de este enfoque es que las rutas entre puntos no son generalmente en una línea recta, puesto que pueden existir accidentes geográficos como ríos, cerros, o montañas que impidan un trayecto en línea recta. En ese caso en particular, le es más corto realizar la ruta más larga al punto del camino azul, pese a que la distancia *haversine* diga lo contrario.

Es por eso, que existen múltiples motores de búsqueda que calculan las distancias más cortas entre puntos basándose en el camino que le tomaría a un camión realizar el recorrido. Destaca en estos servicios el motor de búsqueda *Open Sourse Routing Machine*, o *OSRM*. Estos servicios aprovechan mapas como los de *Open Street Map*, y en conjunto con algoritmos como el de Dijkstra [16], es posible obtener el camino real más corto. Asimismo, se pueden generar estimaciones de tiempo basado en cálculos que incorporen aceleraciones y velocidades máximas de vehículos para cada tipo de camino. Con esto, se genera un estimado mucho más fidedigno que el de la primera instancia, en especial para distancias más cortas [17].

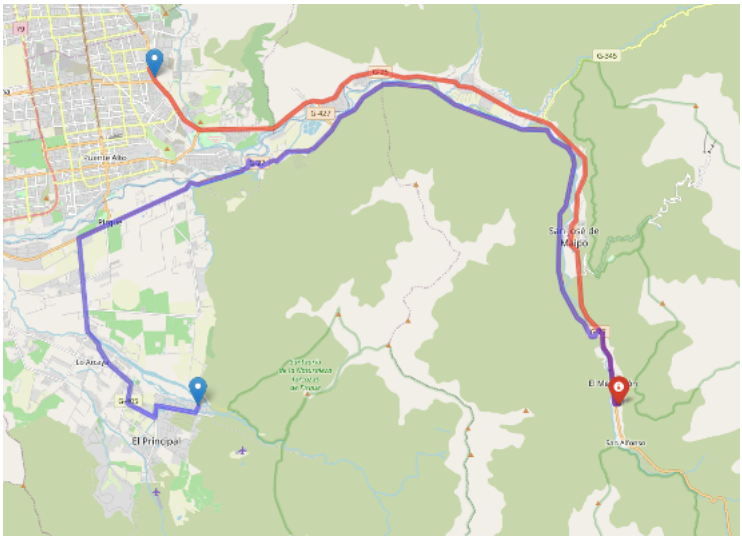

Figura 3.2: Comparación de punto con las dos ubicaciones más cercanas, basado en el camino más corto. En rojo aparece el camino con menor distancia asociada.

Al mandar *requests* a un servidor de OSRM montado en un servidor en *Google Cloud Platform*, es posible generar el cálculo de forma asíncrona de la matriz de distancias. Para esto, es necesario conocer el par de coordenadas que se quiera consultar. Como respuesta, se obtiene el tiempo y distancias estimados en realizar el recorrido y el camino como una polilínea de coordenadas. En la figura 3.2, se aprecian los caminos reales para la combinación de puntos.

Si se busca comparar en términos de cantidad de cómputos por segundo, se obtiene que el cálculo de tiempos y distancias basados en la distancia *haversine* es varios ordenes de magnitud más rápidos que al realizar llamados a la API de OSRM, por lo que existe un compromiso entre velocidad de cálculo y precisión. Como se tiene que generalmente las matrices requieren del orden de miles de cálculos, y la API de OSRM toma alrededor de 500 *requests*

por segundo, se hace plausible calcular absolutamente todos los elementos de las matrices usando el servicio más preciso.

3.2.4. Cálculo de Tiempos de Servicio

El desarrollo de esta etapa requirió un esfuerzo significativo respecto a las anteriores puesto que no existían tanto datos del tiempo de servicio previos y como una estimación manual por parte de un experto.

Por lo anterior, se diseño una encuesta para los transportes que operan en rutas de primera milla y se les hicieron algunas preguntas sobre el tiempo que necesitarían para recoger los productos de cada punto en una determinada ruta. Entre las preguntas que se les hizo, se encontraban la cantidad de productos, el tamaño de los productos y el tiempo que necesitarían para recoger los productos. Se recopilaron más de 800 respuestas que posteriormente se utilizaron para entrenar un modelo regresor del tiempo de servicio.

En base a esas respuestas, se dividió el *dataset* en dos partes, entrenamiento y testeo. Se entrenó un regresor basado en el *Multilayer Perceptron*, con dos capas ocultas de 100 neuronas. El input de la red son 3 parámetros, si corresponde al *Big Ticket* o *Mini Ticket*, como variables binarias y el número de paquetes. Se probó el incluir adicionalmente como atributo el nombre del proveedor, pero por falta de datos, no se pudo hacer una estimación tan granular. En la salida, se tiene como *target* el tiempo de servicio esperado.

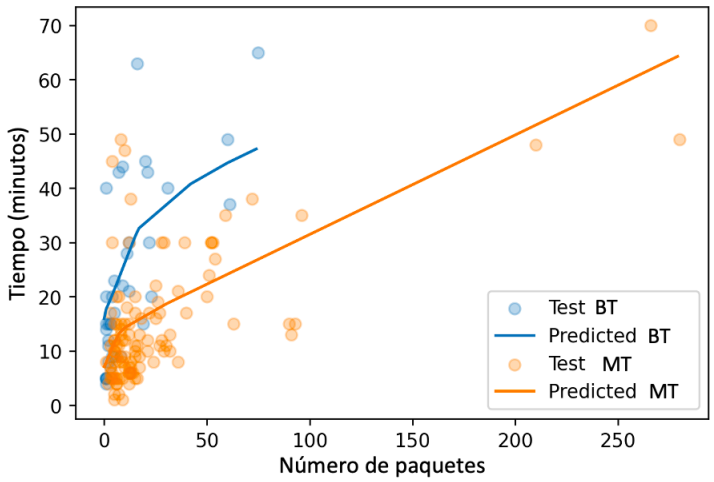

Figura 3.3: Regresor de tiempos de servicio basado en datos de encuestas.

El objetivo de este estimador es minimizar la varianza del error del conjunto de validación para distinta cantidad de paquetes. En una primera instancia, se calcula el error al restar el valor esperado con el valor real. Si el error asociado a esto, es positivo, entonces se está frente a un escenario en donde se atrasó el retiro de los paquetes, asimismo, cuando el valor es negativo, se produce un adelanto.

Con el cálculo del error despejado, se puede calcular la varianza de los errores para un viaje suponiendo que cada parada es una muestra aleatoria independiente de las otras. Para eso, se calculan todas las permutaciones posibles de la suma entre los errores para cierta cantidad de puntos esperados y se obtienen los errores agregados para el percentil 10 y 90,

respectivamente. Con esto, se busca obtener residuales que asemejen ruido blanco, por lo que se busca generar una gaussiana con media en cero.

El problema con este enfoque anterior, es que la gausiana obtenida no necesariamente puede estar centrada en cero. Por lo anterior, se introduce una constante a cada predicción con el objetivo que esta influya en la media para rutas de cierta cantidad de puntos. Resultados de la distribución de errores y de posibles rutas, incorporando las constantes, se analizarán en el siguiente capítulo.

3.3. Diseño del Algoritmo de Optimización

Después de una búsqueda exhaustiva de distintas librerías de optimización en *Python*, con distintos métodos, se decidió utilizar *Google OR-Tools* [15] y sus módulos asociados que permiten resolver el *Vehicle Routing Problem*.

3.3.1. Función Objetivo

En este caso, la función objetivo se diseño basado en requerimientos del negocio.

- Se busca minimizar el tiempo total de ruta
- Se penalizan rutas desbalanceadas
- Se penaliza volumen por sobre cierto umbral

Se parte modelando las rutas desbalanceadas. Se busca generar una penalización por la diferencia de puntos asignados. Por lo tanto, se genera la siguiente función b_k que representa el número de puntos desbalanceados. Donde p_e es el número de puntos estimados y s_{ik} es la función que retorna 1 si el punto i pertenece a la ruta k.

$$b_k = [p_e - \sum_{i}^{n} s_{ik} > 0] \tag{3.2}$$

Por otra parte, el volumen penalizable se modela con la función p_k.

$$p_k = [(\frac{\sum_i^n s_{ik} w_i}{W_k}) - p_v > 0] \tag{3.3}$$

Donde w_i es la capacidad a retirar del seller i, W_k es la capacidad del camión y p_v es el porcentaje de volumen que empieza a ser penalizado.

$$\min f(x) \tag{3.4}$$

$$f(x) = \sum_{k}^{m} (b_k q_b + p_k q_v + \sum_{i}^{n} \sum_{j}^{n} x_{ijk} t_{ij}) \tag{3.5}$$

Donde k corresponde al camión, de m camiones, q_b es la penalización por desbalance, q_v es la penalización por volumen adicional y t_{ij} es el tiempo entre los puntos i y j, incluyendo el tiempo de servicio.

3.3.2. Restricciones

Al modelo se le adicionaron las siguientes restricciones:

1. Ventanas horarias proveedores y centro de distribución
2. Porcentaje máximo de ocupación del camión.
3. Distancia máxima: en este caso, refiere a la distancia desde el primer punto hasta llegar al centro de distribuición.
4. Altura máxima: un camión no puede pasar por un proveedor que tenga productos más altos que el camión. Esto es extremadamente importante cuando se mueven refrigeradores.

3.3.3. Hiperparámetros Modificables

Con el fin de dar flexibilidad a la hora de generar las rutas, y permitir generar rutas más o menos conservadoras, se establecieron ciertos parámetros que pueden ser modificados por los *planner*.

1. Factor de tráfico: este es un multiplicador a los tiempos de traslado.
2. Distancia máxima: corresponde a la distancia máxima de la ruta para los camiones, considera la distancia desde el primer punto al centro de distribución.
3. Porcentaje de ocupación máximo: es el porcentaje del volumen del camión disponible para llenar. Usualmente está en un rango entre 50 y 70 %.
4. Penalización por desbalance de puntos: es un tiempo en minutos que se agrega para cada ruta por la diferencia de puntos bajo el umbral, permite al optimizador buscar otras opciones de rutas que tengan un número similar.
5. Porcentaje de ocupación penalizado: es el porcentaje sobre el cual empieza a dar penalizaciones de tiempo para evitar rutas con camiones extremadamente llenos, se usa con umbrales menores al porcentaje de ocupación máximo, apunta a equilibrar en lo posible los volúmenes retirados.
6. Penalización por volumen adicional: es el tiempo en minutos que se aplica por estar encima del porcentaje de ocupación penalizado.
7. Ventana horaria CD: es el tiempo en el cuál los camiones pueden hacer rutas, se tiene que llegar antes del horario de cierre.

3.3.4. Selección de la Metaheurística de Resolución

Al igual a lo mostrado en el transcurso del marco teórico, en donde se mostraba cómo al utilizar metaheurísticas se puede resolver el *Travelling Salesman Problem*, para el caso del *Vehicle Routing Problem*, se pueden utilizar las mismas técnicas. En una primera instancia, se utiliza una heurística generativa de una primera solución, y posteriormente, se itera en la solución buscando disminuir la función objetivo. En este caso están directamente implementadas en la librería de *Google OR Tools*, y se utilizó como heurística generativa una variación

del algoritmo del vecino más cercano que tiene en consideración las distintas restricciones del problema.

Una vez generada una primera solución, se tiene la opción de ir iterando usando la búsqueda local codiciosa, guiada, tabú, o el templado simulado, todos métodos descritos en la sección anterior. La selección de la última, se dejó a criterio del algoritmo optimizador con el objetivo que este seleccione la meta heurística que le sea más efectiva.

3.3.5. Supuestos a Tener en Consideración

Durante el proceso de generación de datos y diseño del algoritmo se realizaron algunos supuestos operacionales, los cuales se detallarán a continuación:

En una primera instancia, cabe destacar que el problema fue modelado de tal forma que el nodo inicial y el final para todo camión es el centro de distribución y los nodos intermedios representan las distintas paradas de la ruta. Con esto, los camiones que quedan sin utilizar poseen rutas con origen y destino en el centro de distribución, sin nodos intermedios.

Adicionalmente, se busca desde la operación tomar como supuesto que las rutas empiezan en el primer punto en la hora inicial, por lo que el origen real para el inicio de la ventana horaria es el primer proveedor a retirar. Así, toda combinación en la matriz de tiempos y distancias, que considere viajes desde el origen, tiene pesos configurados en cero. En una futura iteración del proyecto, se espera incorporar los orígenes de los transportes para generar rutas más eficientes en los tiempos y distancias de colocación para el transportista.

Un aspecto importante a la hora de generar las rutas, es que se tomaron en consideración márgenes de seguridad para que las estimaciones generadas por los distintas fuentes de datos fuesen suficientemente conservadoras con el fin de evitar problemas operacionales. Estos márgenes de seguridad son configurados de forma manual de tal manera que son de cierta forma hiperparámetros a la hora de planificar las rutas. Ejemplos claros de márgenes de seguridad son: configurar un factor de tráfico basal en 1,5, utilizar a lo más el 50 % de la capacidad del camión, poner como umbral penalizable basal en el 35 % de la capacidad del camión. Es decir, si se toma un proveedor que se encuentra entre el 35 y 50 % de la capacidad, y no tiene una mejor opción, se acepta, pero con penalización. En tanto, si se busca agregar un proveedor a un camión que ya está ocupado por sobre el porcentaje de penalización porque era la mejor opción en términos de tiempo, puede que la penalización influencie al algoritmo a buscar otras opciones.

Por último, el modelamiento del problema no considera tiempos de descanso asociados para las rutas puesto que estas están divididas en horarios AM y PM, ya que no tienen duraciones superiores a 4 horas. Donde en el presente trabajo se aplicó la metodología principalmente a las rutas AM para realizar una prueba de concepto, y posteriormente, expandir el proceso al conjunto de las rutas de primera milla.

3.4. Disponibilización de Métricas y Mapas de Rutas Generadas

Al resolver las distintas rutas de manera diaria, los *planner* requieren de información acerca de ellas para determinar si las decisiones tomadas por la herramienta son adecuadas.

Es por lo anterior que se generaron distintos tipos de reportes automatizados que mostraban métricas por camión como: distancias totales, tiempos total en ruta, volumen ocupado, porcentaje de ocupación, cantidad de puntos visitados, cantidad de paquetes cargados, entre otros. De igual manera, se desarrolla un reporte más granular que muestra, para cada ruta, los proveedores a retirar en orden con, volúmenes, cantidades, dirección y la placa patente del camión asignado.

Para dar certezas al planner que las rutas siguen una trayectoria adecuada y razonable, en cada planificación, se genera un mapa que permite visualizar de mejor manera las rutas creadas. El mapa se desarrolló utilizando la librería *Folium* de Python.

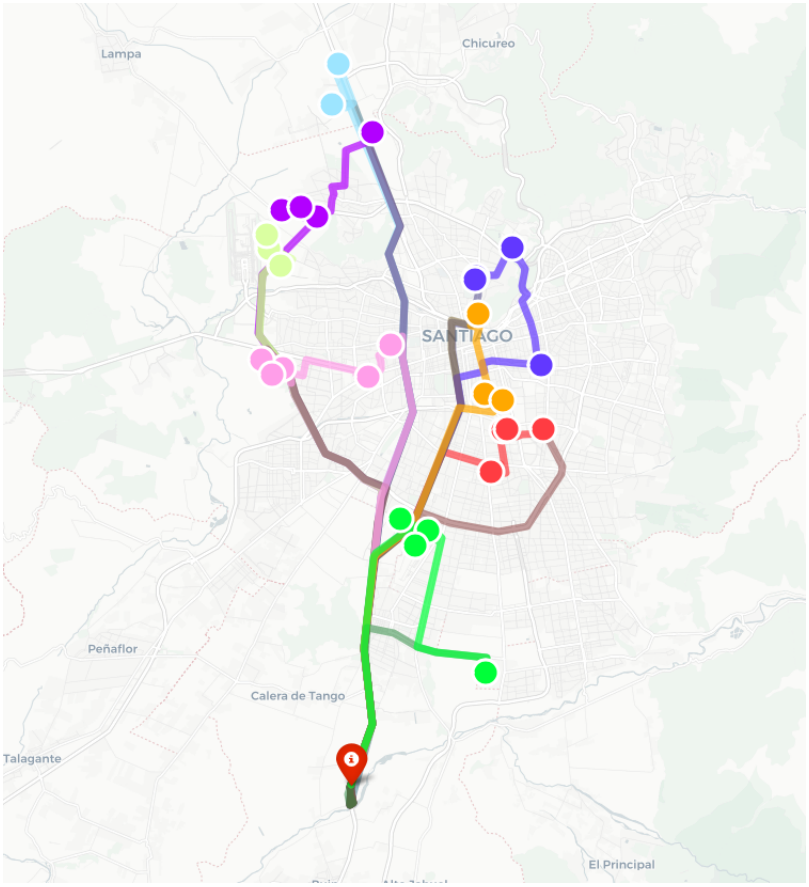

Figura 3.4: Mapa de ruta de ejemplo generada con la plataforma de ruteo.

En la figura 3.4, se aprecian 8 rutas representadas por distintos colores retirando múltiples proveedores, y se generó a modo de ejemplo para ilustrar lo que visualizan los *planner*. Este mapa es interactivo y aporta información al apretar cada elemento, por ejemplo, los caminos entregan distancia y tiempo de ruta estimado, y los puntos el volumen a retirar y el tiempo

de servicio, en conjunto con el nombre del proveedor. Por razones de confidencialidad, no se mostrarán detalles adicionales.

3.5. Evaluación del Impacto de la Herramienta de Ruteo

Uno de los desafíos más complejos a la hora de llevar a cabo un proyecto es poder lograr medir sus resultados y contrastarlos respecto a un punto basal, o denominado en la jerga del negocio, un *baseline*.

A la hora de diseñar las rutas, el indicador clave del negocio, que termina influyendo de mayor manera el éxito del proyecto, es el costo monetario que implica mantener funcionar la flota diaria de camiones, y de forma secundaria, los indicadores de calidad a la hora de realizar las rutas. Esto ocurre porque, al utilizar los camiones de forma más eficiente, bajan los costos asociados al funcionamiento, y asimismo, cuando ocurre que empeoran los indicadores de nivel de servicio, se suele necesitar más camiones para cubrir posibles imprevistos por problemas operacionales, utilizando más camiones de lo planificado, y por ende, incrementando de todas formas los costos. Es por eso que existe, a la hora de diseñar un experimento, algunas variables del problema que son independientes del uso o no de la herramienta desarrollada. Se busca que para estas variables, se mantengan estadísticamente similares para los dos grupos que se muestrean. Estas serán denominadas variables de control y que, en este caso son:

- Número de proveedores a retirar
- Número de paquetes a cargar
- Paquetes a retirar por proveedor promedio

Las variables dependientes del método de ruteo, en tanto, tienen relación con la aplicación o no de la herramienta desarrollada y busca probar de forma estadísticamente significativa que la aplicación de la herramienta tiene un efecto en términos de eficiencia en las rutas. En este caso, estas variables son:

- Número de camiones utilizados
- Productos retirados por camión en promedio
- Proveedores visitados por camión en promedio.

Para hacer la prueba estadística, se toman dos muestras en rangos de fecha horaria independientes que comparten ser parte de períodos valle de venta. Detalles del tamaño de muestra y valores se discutirán en la sección de análisis de resultados. Se determina el nivel de significancia en 5 %. Con esto en cuenta, se tiene el siguiente listado de niveles de significancia y su interpretación para determinados p valores:

- $p_{val} >= 0.1$: diferencia no significativa.
- $0.1 > p_{val} > 0.05$: diferencia ligeramente significativa.
- $p_{val} <= 0.05$: diferencia significativa.

- $p_{val} <= 0.01$: diferencia extremadamente significativa.

Dependiendo de la forma en la que distribuyan los datos, se pueden realizar dos tipos de test estadísticos distintos que permitan comparar las medias de las muestras y determinar si existe, o no, una diferencia significativamente estadística. Si las muestras distribuyen normalmente, tienen varianzas similares, son independientes y poseen un tamaño de muestra cercano a 30, el *t-test* puede ser adecuado. En tanto, si no se cumple, por ejemplo, el requerimiento de normalidad, es recomendable utilizar el *Mann-Whitney U-test* [18], teniendo en consideración que es recomendable que las muestras tengan varianza similar.

Capítulo 4

Análisis de los Resultados

4.1. Estimación de Tiempos de Servicio

El tiempo de servicio, o *Service Time* corresponde al tiempo que le toma al transportista retirar los paquetes de un proveedor. Como esta estimación de tiempo tuvo que generarse en el trascurso de este trabajo, se expandirá en el estudio de los resultados obtenidos.

Como fue descrito en la sección anterior, se entrenó un modelo regresivo basado en el *Multi Layer Perceptron* en base a resultados de más de 800 respuestas, habiendo entrenado el modelo con una fracción de estos y, con el restante, se realizan algunos ajustes para centrar los errores asociados a las rutas alrededor del cero.

En la figura 4.1, se aprecia cómo para los dos tipos de combinaciones, de rutas de *Big Ticket* y *Mini Ticket*, el modelo logra generar dos regresiones considerando cierta cantidad de paquetes y, estas se acercan a los puntos vinculados con la muestra de datos de validación. De todas maneras, existe, especialmente para las rutas de *Big Ticket*, puntos profundamente más alejados de la curva.

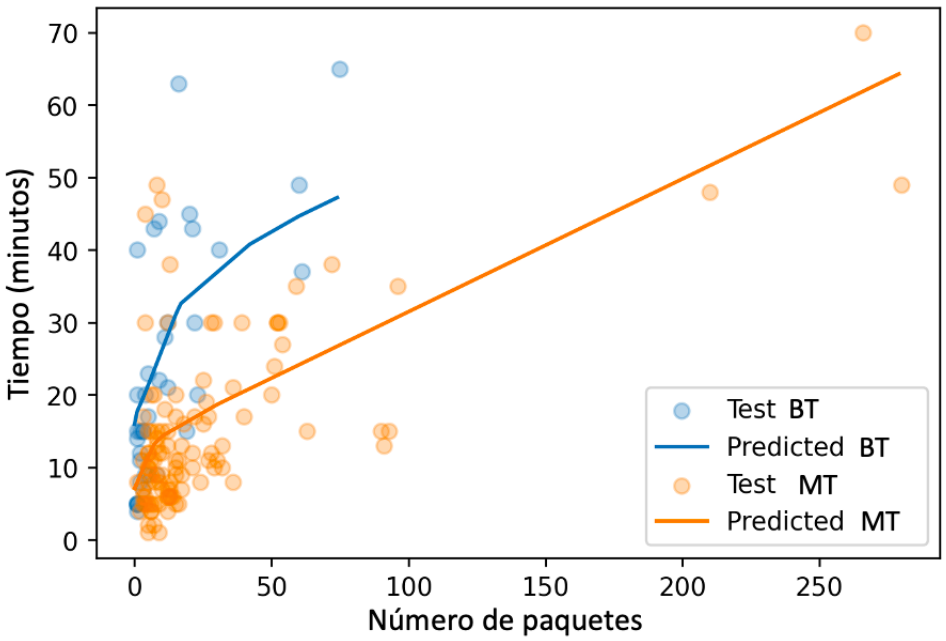

Figura 4.1: Primeros resultados de regresión

4.1.1. Estudio de Residuales en Big Ticket

Basado en los datos iniciales del modelo, se busca entender cómo distribuyen los errores basados en cada una de los tipos de ruta. Como se había mencionado antes, el error corresponde a la resta entre el valor generado por la regresión dado el tamaño del paquete y del número de paquetes; y el valor real. En este caso, se hace el análisis de la distribución del error para rutas con un solo punto de *Big Ticket*.

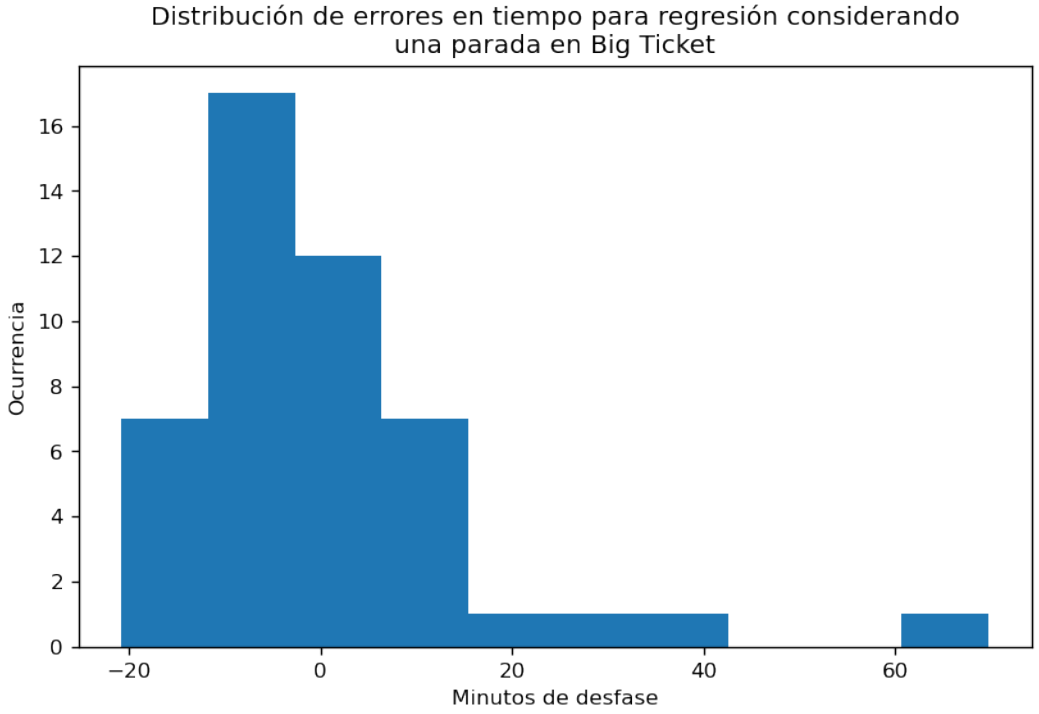

Figura 4.2: Distribución de errores de estimación de *Service Time* para rutas *Big Ticket* considerando una sola parada.

Como se aprecia en la figura 4.2, se tiene que la gran parte de los valores relacionados con el error están centrados en cero, con una media con leve sesgo hacia minutos negativos de desfase. Sin embargo, la distribución asemeja a una gausiana positivamente asimétrica, con un valor positivo de atraso de 60 minutos, por lo que puede que la combinatoria en suma de errores para simular rutas pueda estar ligeramente inclinada hacia el atraso. Con esto, se asume en que las distintas paradas son estadísticamente independientes entre ellas y, que el error de atraso de una parada no depende de otras, por consiguiente, se espera que la varianza de la distribución para rutas vaya incrementándose, pero a un ratio similar al de lo expuesto en el teorema central del límite, por lo que la desviación estándar dividida en el número de puntos a retirar debería disminuir.

Desde la operación y basado en pruebas de ruteo previas, se llegó a otro supuesto: el número de paradas esperado en una ruta debe ser igual a 4 para este tipo de rutas y para las ventanas horarias de funcionamiento de la ruta. Con esto y, debido al otro supuesto de independencia entre los errores muestreados, se puede utilizar el teorema central del límite (TLC) para hacer muchos muestreos de paradas independientes, generando múltiples paradas y, con ello, obteniendo la figura 4.3.

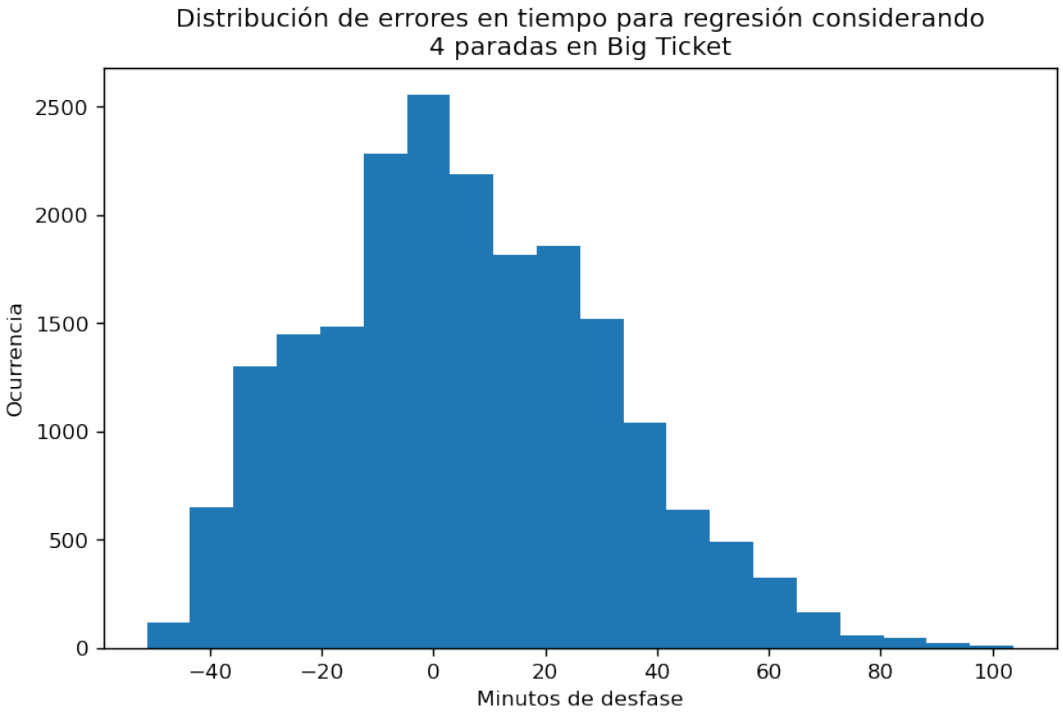

Figura 4.3: Distribución de errores para rutas de 4 paradas basado en supuestos del TLC.

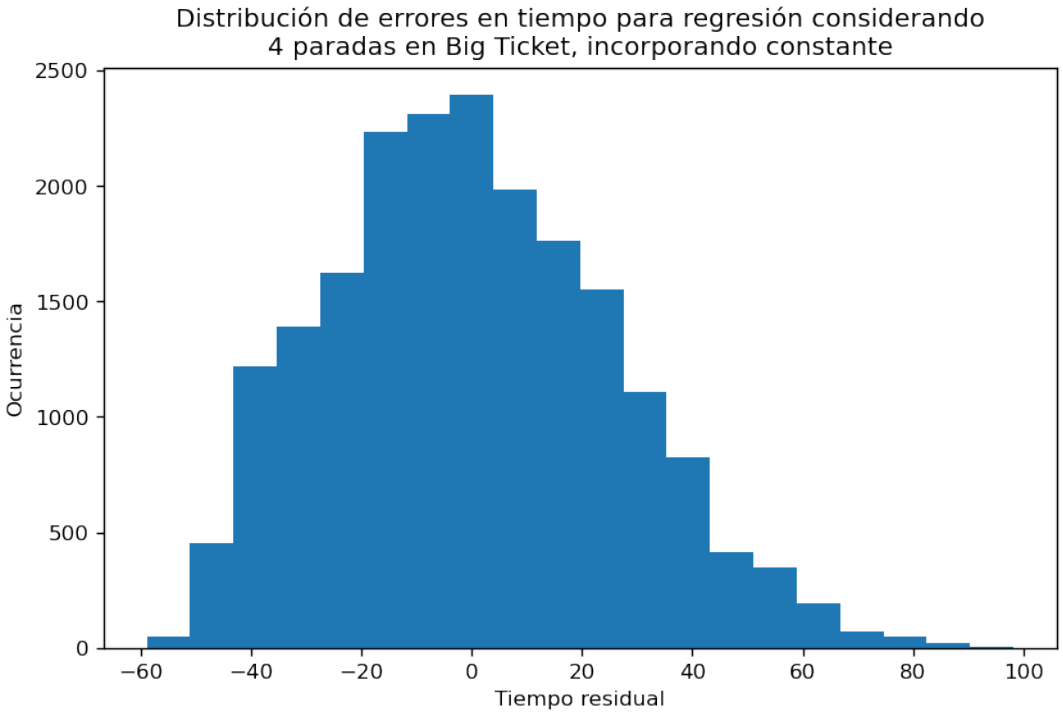

Figura 4.4: Distribución de errores para rutas de 4 paradas considerando tiempo adicional constante.

En la figura 4.3, se aprecia la distribución de errores para simulaciones de rutas *Big Tic-*

ket, considerando 4 muestras de paradas a lo largo de la ruta. Se aprecia cómo tiene una distribución normal ligeramente desfasada del origen. Esto representa un problema porque significa que, en promedio, las rutas presentan un desfase hacia el atraso, por lo que se puede argumentar que se está subestimando el tiempo necesario para retirar los productos. Con eso, se pueden producir posibles problemas operacionales. Para compensar aquello, a la salida del modelo, se le agrega un tiempo adicional que cancela el desfase, ajustándolo específicamente al número de paradas esperado.

En la figura 4.4, se incorpora ese desfase a la salida del regresor y tiene el efecto inmediato de centrar la gausiana en cero. Con esto, se puede medir los tiempos esperados de adelanto y atraso para las rutas obteniendo los percentiles 10 y 90. Obteniendo que, el 90 % de las rutas se adelanta menos de 30 minutos y se atrasa menos de 32 minutos. Con estos datos, es posible generar ajustes a las ventanas horarias de las rutas y, basado en previa coordinación con el equipo de planificación, poner un horario de término para la ruta suficientemente conservador como para que no sea usual la ocurrencia de atrasos, como también no adelantar significativamente las rutas y, tener capacidad ociosa en los camiones.

4.1.2. Estudio de Residuales en Mini Ticket

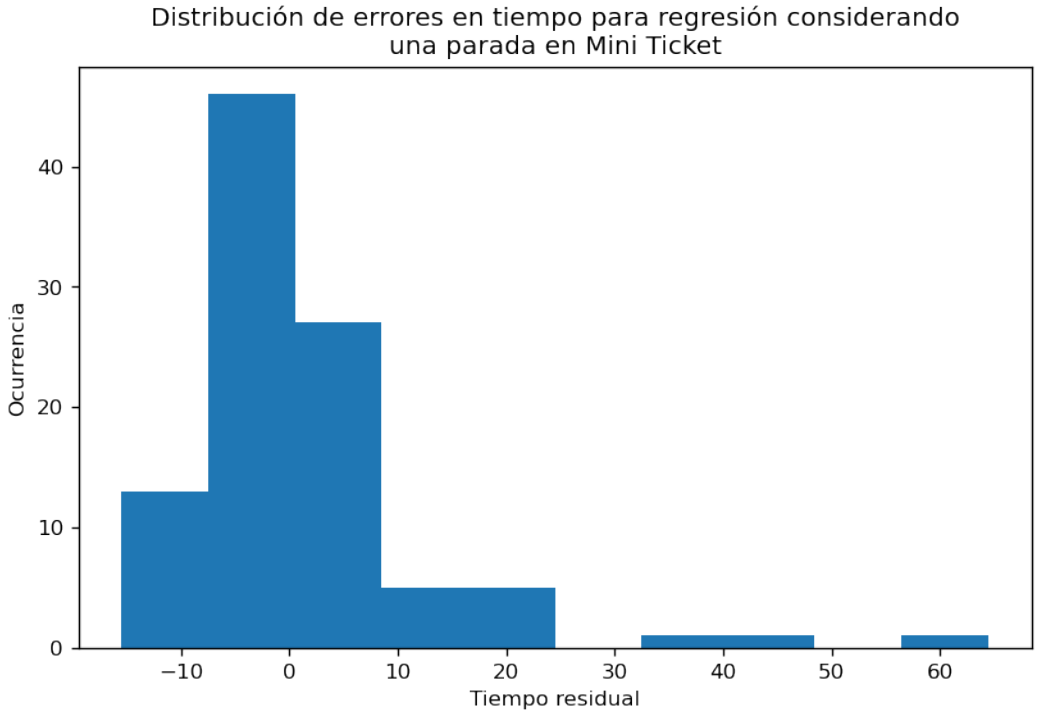

Figura 4.5: Caption

Al igual que en el caso de las rutas de *Big Ticket*, se obtiene en la figura 4.5 una campana de gauss positivamente asimétrica. Los valores con errores mayores a la media hora son extraños, pero pueden ser atribuibles a problemas operacionales asociados a distintos factores, como que el camión tenga que esperar al proveedor que esté listo, como también otros problemas operacionales. El tomar estos *outliers* posibilita al estimador y a la generación del sesgo, robusteza frente a posibles problemas operacionales y tener en cuenta su frecuencia de

ocurrencia para la generación de las rutas.

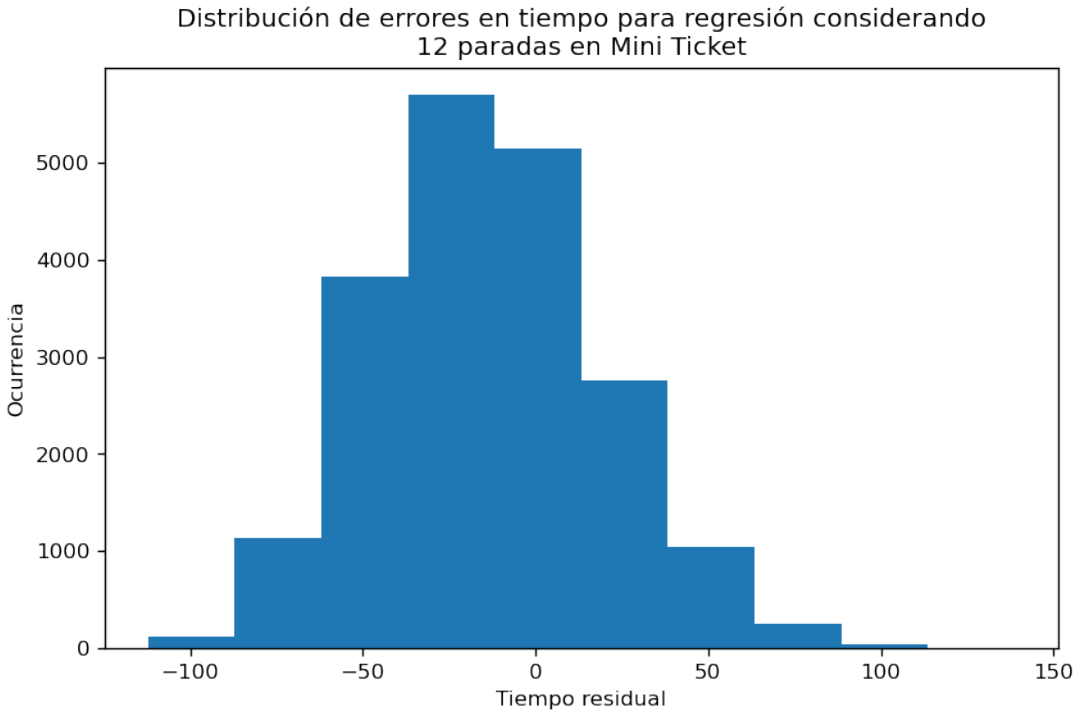

Figura 4.6: Distribución de errores acumulados para simulación de rutas con 12 paradas de *Mini Ticket*.

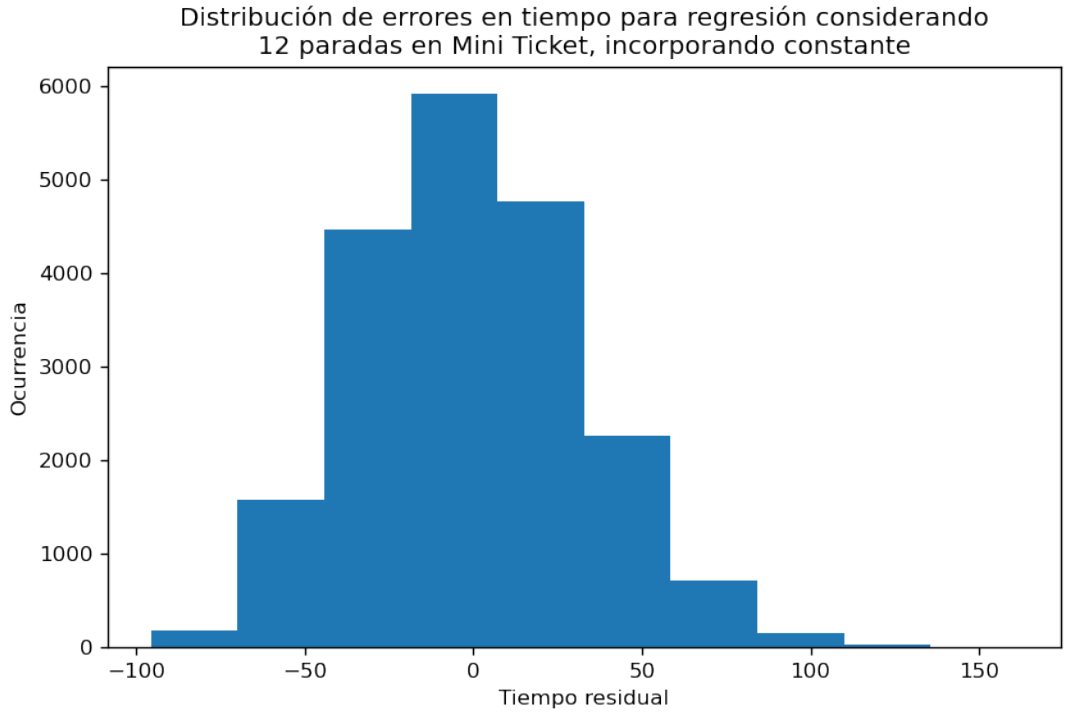

Figura 4.7: Distribución de errores acumulados para simulación de rutas con 12 paradas de *Mini Ticket*, ajustado por un sesgo que la centra en cero.

Al igual que el caso anterior, se agrega un sesgo basado en la distribución resultante de la figura 4.6, por lo que se obtiene nuevamente una distribución normal centrada en cero en la figura 4.7. Cabe destacar que se toma como supuesto que un camión podría tener rutas de hasta 12 paradas para este tipo de viaje. Este estimado puede ser un poco extremo dado que la referencia manual de paradas por camión se encuentra alrededor de las 5 paradas por camión. De todas maneras, al igual que en el caso anterior, el 90 % de las rutas simuladas de 12 paradas no sufrirían adelantos de más de 28 minutos, ni atrasos de 30 minutos.

4.1.3. Apreciaciones Finales

La adición de los sesgos anteriores tiene como consecuencia que las regresiones inicialmente generadas por el modelo, quedan ligeramente desfasadas al atraso, en las rutas de *Big Ticket* y, ligeramente adelantadas en rutas de *Mini Ticket*. En la figura 4.8, se aprecia justamente ese comportamiento en las curvas verde y roja, respectivamente.

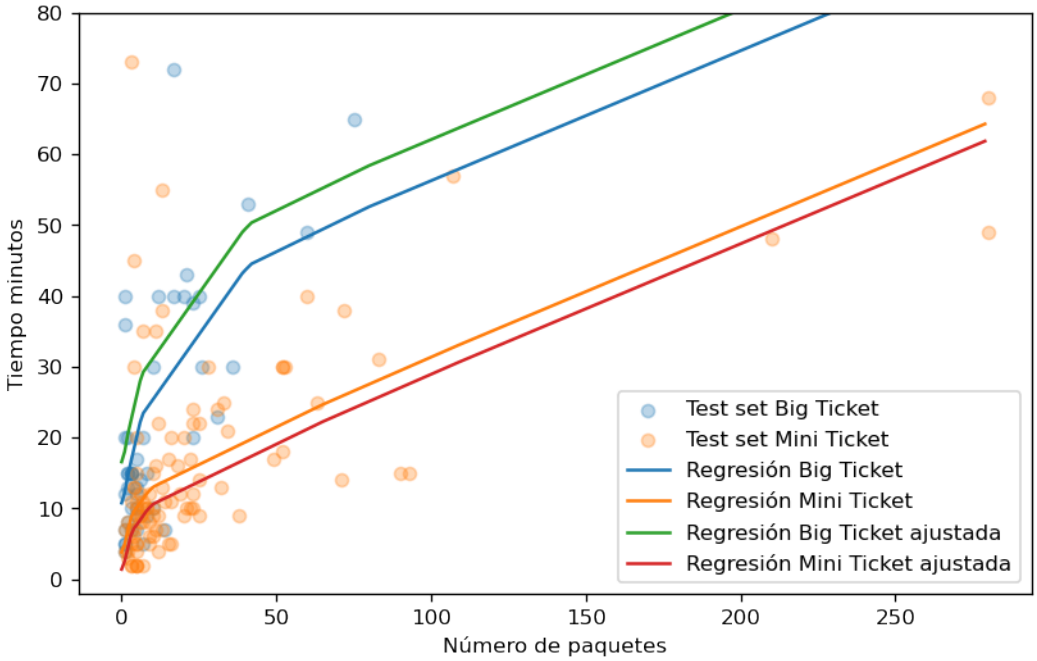

Figura 4.8: Regresión de tiempo de servicio para *Big Ticket* y *Mini Ticket* basado en el número de paquetes a cargar, considerando los sesgos calculados para centrar el error de tiempo estimado de las rutas generadas en cero.

Con lo expuesto anteriormente, se puede argumentar que este desarrollo cumple con desarrollar una metodología que permita estimar el tiempo de servicio, teniendo como mayor tiempo de atraso de las rutas simuladas para el percentil 90, de 32 minutos. Este atraso se toma en adelante en consideración como media hora menos posible de ruta. Con esto, si las rutas deben terminar a las 14:00, al tener en consideración este posible error, se adelanta el horario de término a las 13:30. A futuro probablemente se pueda desarrollar algo más robusto y que tenga aún menos error para las rutas, como lo desarrollado en [1], sin embargo, se requerirá información más detallada de los proveedores y, adicionalmente, de información relacionada a los GPS de los camiones para poder lograrlo.

4.2. Rutas Generadas para Flota Big Ticket

Para *Big Ticket* se configura el piloto con los siguientes parámetros:

- Factor de tráfico: 1.5
- Distancia máxima: 50 km
- Porcentaje de ocupación máximo: 50 %
- Número de puntos esperados: 5
- Penalización por desbalance de puntos: 5
- Porcentaje de ocupación penalizado: 35 %
- Penalización por volumen adicional: 5
- Ventana horaria CD: 9:00 a 13:00

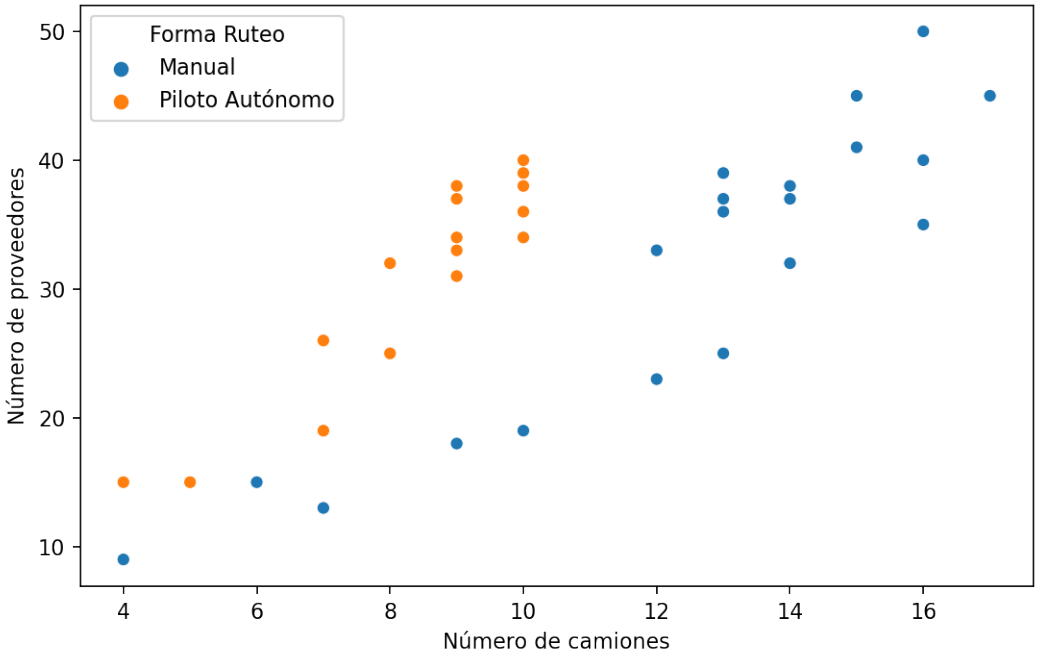

Figura 4.9: Número de puntos vs cantidad de camiones para rutas *Big Ticket*

Como se aprecia en la figura 4.9, las rutas que utilizaron el algoritmo de ruteo tuvieron, para cierta cantidad de puntos a retirar, una menor cantidad de camiones utilizados. Por lo que sugiere que los camiones realizan más paradas por ruta. Sin embargo, esta no muestra si este efecto es debido completamente a el algoritmo utilizado o más bien por cambios de otras variables. Es por eso, que se midieron 3 variables de control y 3 experimentales, con el objetivo de medir varias dimensiones adicionales y, determinar el verdadero impacto del algoritmo.

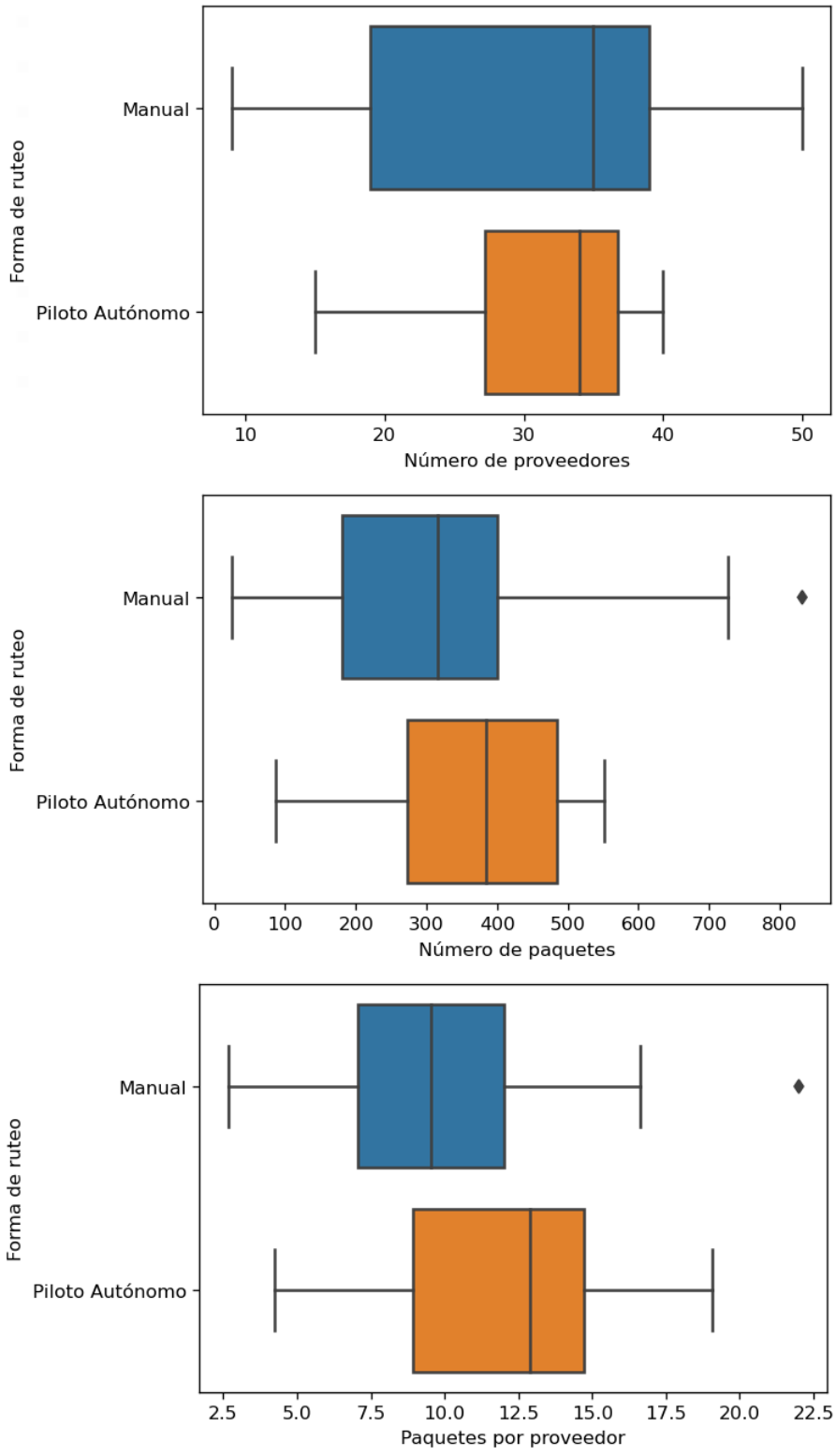

Figura 4.10: Variables de control *Big Ticket*

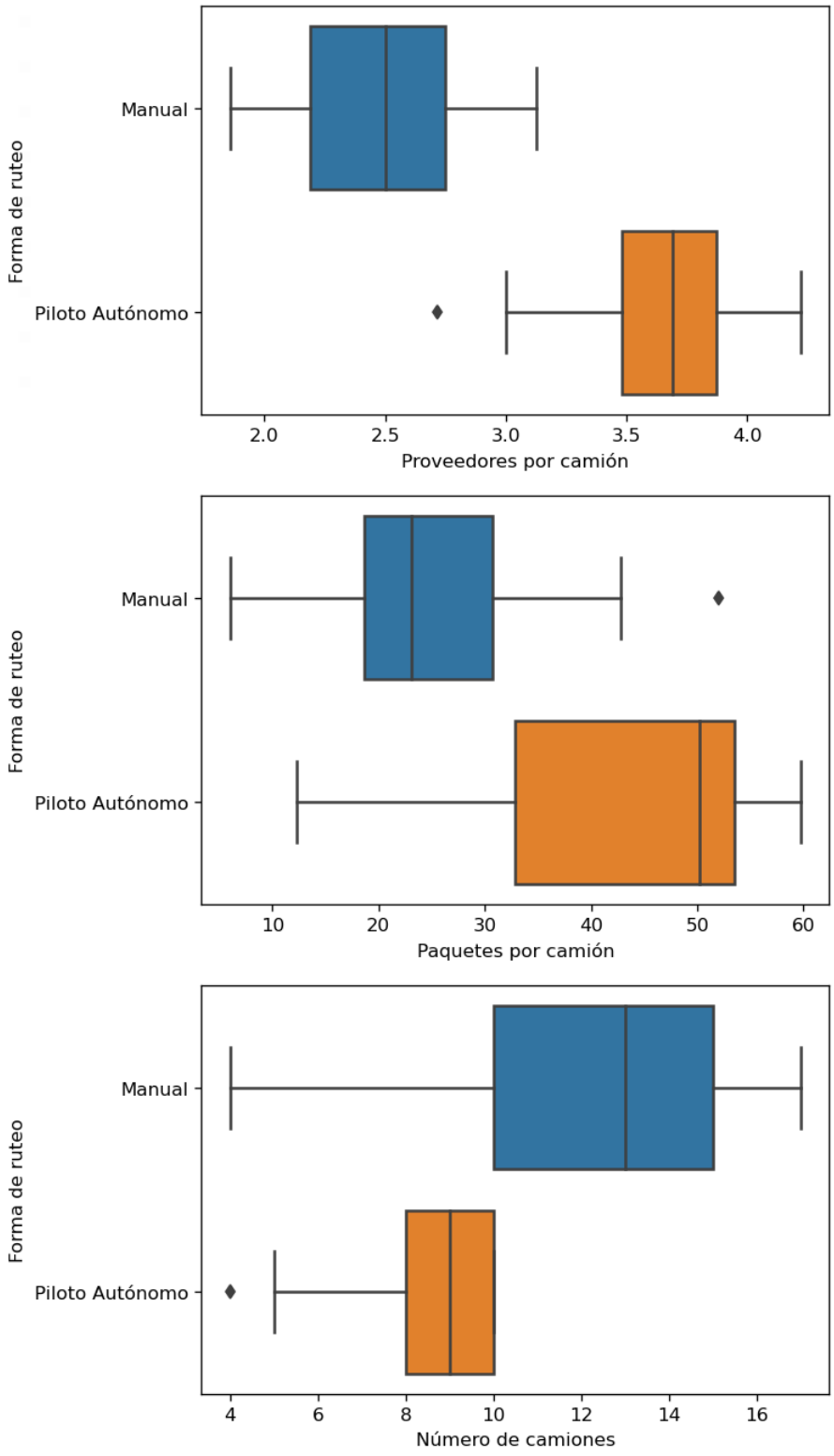

Figura 4.11: Variables manipuladas por algoritmo en rutas Big Ticket

En la figura 4.10, se tiene la distribución de 3 variables para ambas muestras, mostrando el número de proveedores a visitar, el número de paquetes a retirar, para un día en particular y, el ratio entre ellos. Estas variables, como fue comentado en la sección de *Evaluación de impacto de la herramienta de ruteo*, son variables que se busca se mantengan constantes en

ambas muestras. Por lo anterior y, dado que las distribuciones no se distribuyen normalmente, se realiza para cada par de muestras el *Mann-Whitney U-test* y se busca que no exista una diferencia estadísticamente significativa para las medias.

Para el caso de la figura 4.11, se obtienen las 3 variables directamente afectadas por el uso del algoritmo de ruteo. Estas son, el número de camiones utilizados en cada día y los ratios asociados con la cantidad de paradas y paquetes que se hacen en promedio para cada retiro. En este caso en particular, se busca que la diferencia entre medias sea estadísticamente significativa y se repite la misma prueba estadística.

Estos resultados quedan presentes en la tabla 4.12, donde se muestra: el valor promedio para cada variable y muestra, el tipo de variable y si existen diferencias estadísticamente significativas tanto en varianza como en media.

Variable	Manual	Piloto Autónomo	Tipo de variable	Varianza estadísticamente significativa	Diferencia estadísticamente significativa
Puntos	30,4	31,3	Control	No	No
Paquetes	317	364	Control	No	No
Paquetes por proveedor	9,95	11,77	Control	No	No
Número de camiones	12,05	8,56	Experimental	Sí	Sí
Paquetes por camión	24,7	42,7	Experimental	No	Sí
Proveedores por camión	2,47	3,63	Experimental	No	Sí

Figura 4.12: Tabla resumen de indicadores de ruta para ruta en Big Ticket

En una primera instancia, al observar los resultados de la tabla 4.12, es posible afirmar que la cantidad de puntos a retirar se mantuvo constante entre los experimentos. Por ende, en la métrica de proveedores retirados por camión es posible afirmar que el incremento de puntos visitados, que aumentó en un 46.9 %, es producido plenamente por el algoritmo de ruteo.

En tanto, pese a que no exista una diferencia estadísticamente significativa, para la distribución de la cantidad de paquetes por proveedor, hubo un aumento de un 18 %. Lo anterior, tiene como consecuencia que posiblemente el número de paquetes cargados por camión quede inflado en ese porcentaje, por lo que para medir el posible impacto atribuible al algoritmo se puede calcular el número de paquetes por camión dividido en el aumento de la cantidad de paquetes por proveedor. Esto da como resultado que la cantidad de paquetes retirados por camión ajustado, es de un 46.5 %, similar al incremento de puntos visitados por camión.

Es por el efecto de ambos indicadores, que se puede atribuir una disminución en la cantidad de camiones necesarias para para funcionar, bajo las mismas condiciones, en un 31.9 %. Este valor es similar a la disminución del 28.9 % obtenido sacando la división entre los resultados del piloto con el proceso manual respecto a la cantidad de camiones usados.

4.3. Rutas Generadas para Flota Mini Ticket

Para *Mini Ticket* se configura el piloto con los siguientes parámetros:

- Factor de tráfico: 1.5
- Distancia máxima: 50km
- Porcentaje de ocupación máximo: 50 %
- Número de puntos esperados: 8
- Penalización por desbalance de puntos: 10
- Porcentaje de ocupación penalizado: 50 %
- Penalización por volumen adicional: 0
- Ventana horaria CD: 9:00 a 13:00

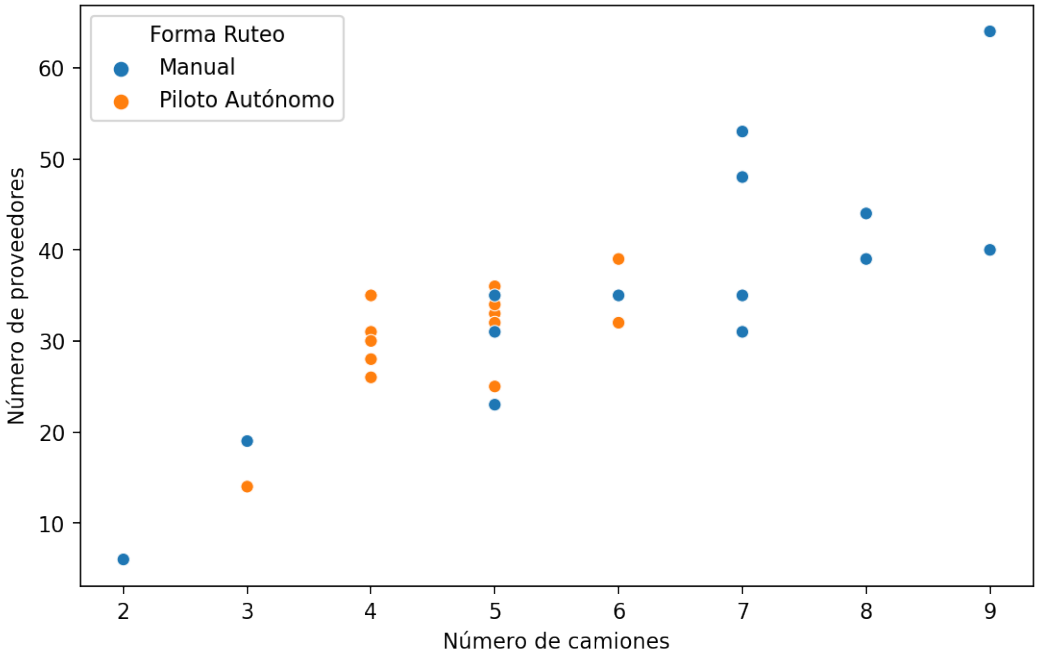

Figura 4.13: Número de puntos vs cantidad de camiones para rutas *Mini Ticket*

En la figura 4.13, se aprecia cómo las rutas para el caso del piloto estaban inmersas en un escenario completamente diferente. A simple vista, se aprecia que el número de proveedores para la muestra realizada con el escenario manual tiene una varianza mucho más grande que el caso con el piloto. Esto puede dar ciertas luces de que ambos escenarios no sean del todo comparables. De todas maneras, se aprecia a simple vista que, para la misma cantidad de puntos, en algunos casos se obtiene una cantidad de camiones menor para el piloto.

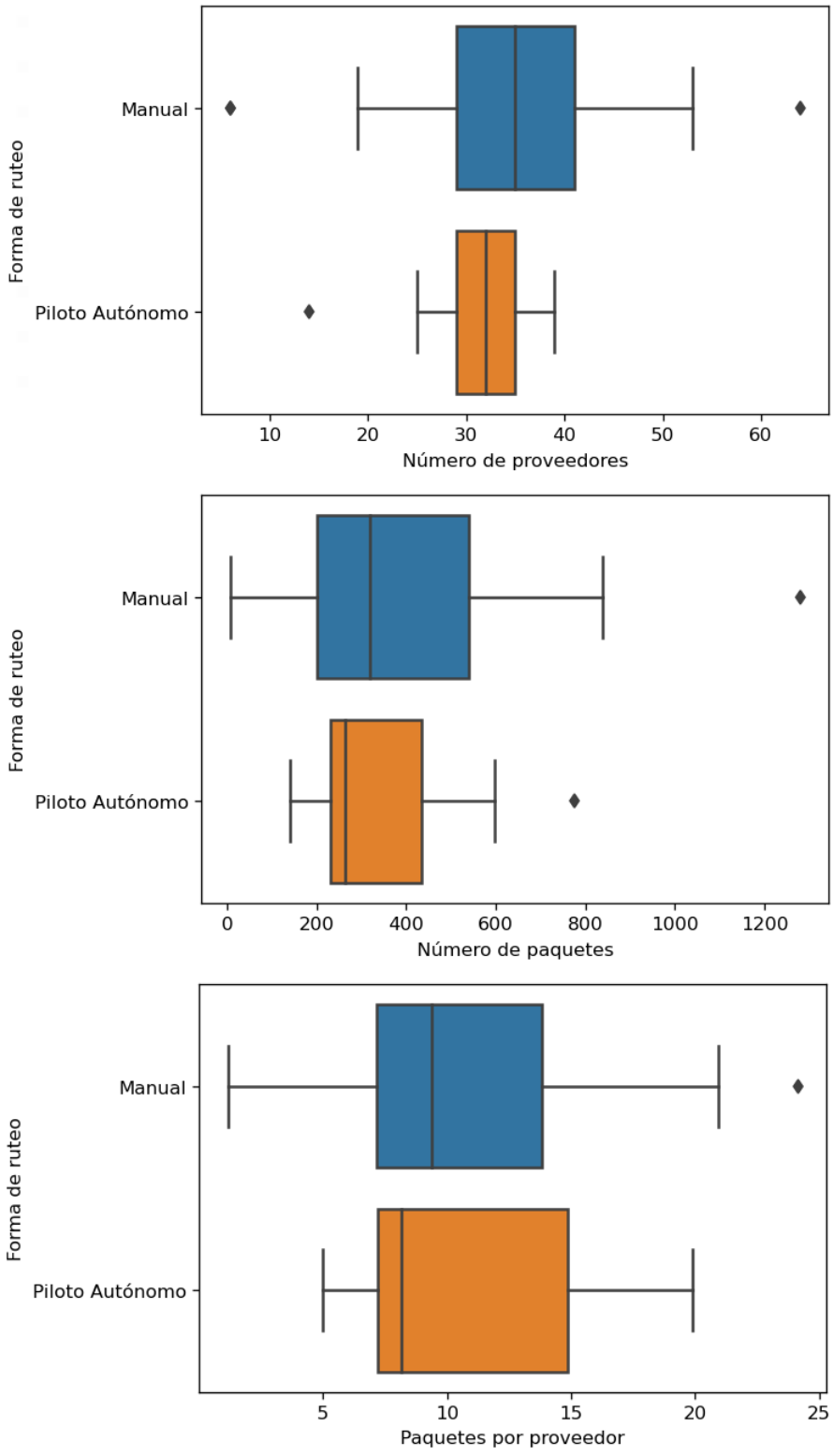

Figura 4.14: Variables de control *Mini Ticket*

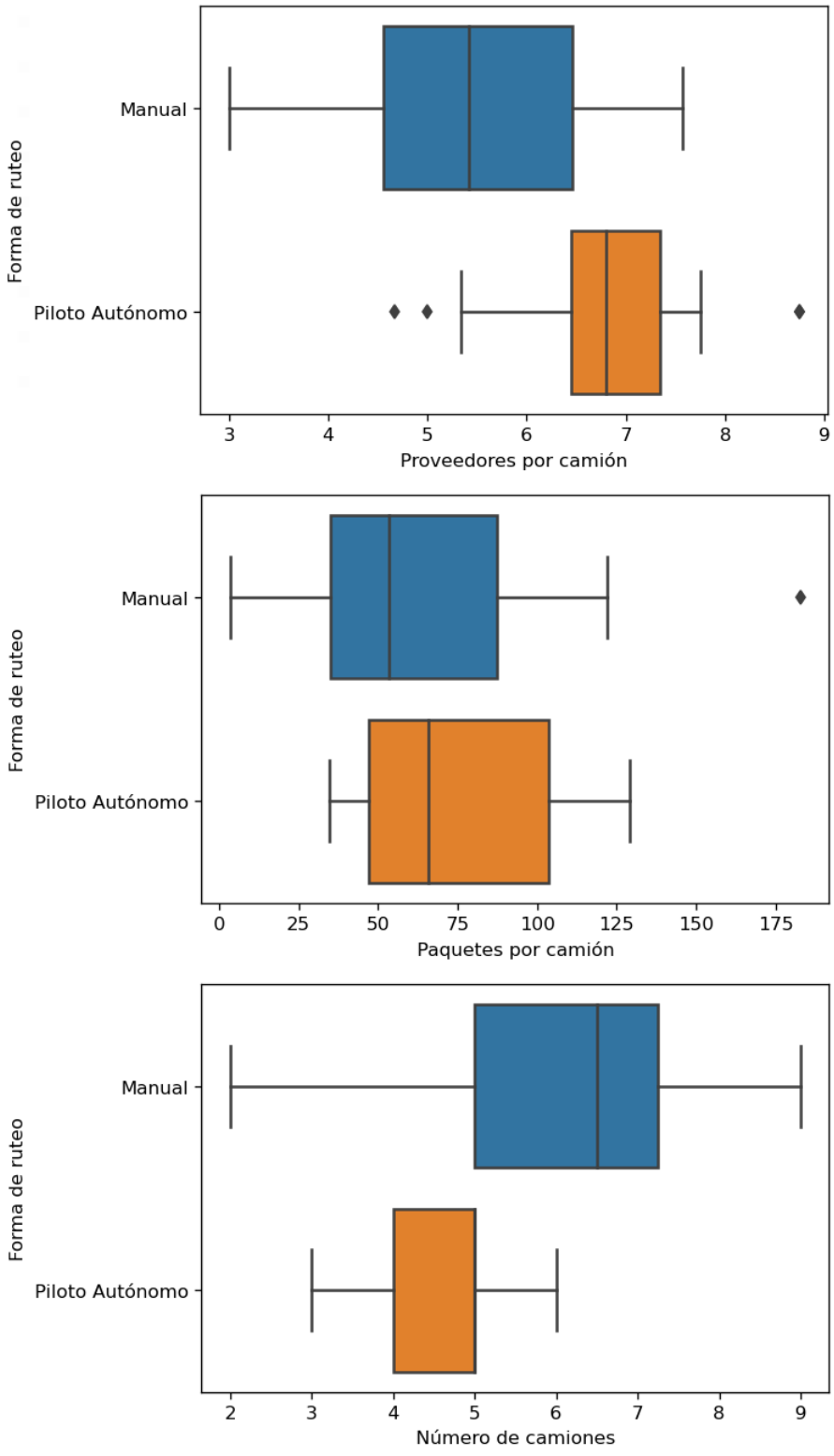

Figura 4.15: Variables manipuladas por algoritmo en rutas *Mini Ticket*

Variable	Manual	Piloto Autónomo	Tipo de variable	Varianza estadístiamente significativa	Media estadisticamente significativa
Proveedores	33,8	31,1	Control	Sí	No
Paquetes	411	343	Control	No	No
Paquetes por proveedor	10,9	11,1	Control	No	No
Camiones	6,0	4,6	Experimental	Sí	No
Paquetes por camión	63,8	74,6	Experimental	No	No
Proveedores por camión	5,4	6,8	Experimental	No	Sí

Figura 4.16: Tabla resumen de indicadores de ruta para ruta en *Mini Ticket*

De la tabla 4.16, se aprecia que, tanto el número de camiones como el número de proveedores poseen una diferencia de varianza estadísticamente significativa. Esto significa que, las condiciones basales para el *Mann-Whitney U-test* no están dadas y, por tanto, al igual que el caso anterior, estas distribuciones no son comparables. Por ende, es necesario tomar otras métricas para poder medir el impacto del algoritmo de ruteo.

Si se observan las otras variables de control, presentes en la figura 4.14, todas poseen distribuciones que no poseen diferencias estadísticamente significativas tanto en media como en varianza. Con esto, se descarta la posibilidad de utilizar la cantidad de proveedores retirados en promedio por camión y solo queda abierta la posibilidad de comparar la cantidad de paquetes por camión cargados.

Por un lado, el número de paquetes tiene una distribución que no muestra diferencias significativas según el *Mann-Whitney U-test*, pero posee una media que disminuye en un 16 % durante la aplicación del piloto. Esto probablemente se deba al *outlier* presente en la muestra de ruta manual, en donde se cargaron en un día más de 1200 paquetes como se aprecia en la figura 4.15. Por el otro lado, la diferencia de paquetes cargados por camión es de un 16 %, pero no es estadísticamente significativa dado el tamaño de muestra de datos.

Es por ello que, de la forma en la que está diseñado el experimento, no es posible otorgarle al algoritmo de ruteo mejoras en la eficiencia de las rutas de *Mini Ticket*. Esto se debe, principalmente, a que se tienen 18 muestras manuales y 17 del piloto para este tipo de ruta. El incrementar el tamaño de muestra puede aclarar el posible impacto, estimado bajo las diferencias de medias en un 13 %,.

Capítulo 5

Conclusiones

Durante el presente trabajo se generó una plataforma que: toma los archivos generados por el equipo de planificación, los procesa de tal forma que, geocodifica las direcciones, calcula distintos estimados de tiempo en ruta, servicio y distancia, *obteniendo los datos necesarios para que el algoritmo de ruteo pueda cumplir con las restricciones reales de tiempo y capacidad de los vehículos disponibles.*

Es más, durante el desarrollo de este proyecto se logra concebir e implementar una *metodología que permita estimar el tiempo de servicio que le tomará a un camión retirar los paquetes de un punto determinado*, de tal forma que el error de subestimación para el 90 % de los casos, para distintas simulaciones de ruta, es de 32 minutos. Con estos datos, se generan rutas factibles tanto en los flujos de *Big Ticket* como *Mini Ticket*, con disminuciones asociadas al primer flujo de los costos operacionales en un 31 %.

Es por lo descrito anteriormente que el objetivo principal, asociado a *optimizar la planificación del transporte de primera milla al desarrollar una herramienta que permita resolver el problema de ruteo de vehículos*, es parcialmente cumplido puesto que no se generan mejoras estadísticamente significativas en la eficiencia de las rutas de *Mini Ticket*.

De todas maneras, el impacto generado no deja de ser relevante, el proyecto desarrollado constituye ser un habilitador en la estrategia de la empresa para entregar los productos con menores costos logísticos, en el menor tiempo posible. Se logra automatizar una tarea manual, permitiendo a los operarios del algoritmo ahorrar tiempo, y mejorar los indicadores de servicio al incorporar variables dinámicas frente a las estáticas anteriores, las adición de las cuales, adicionalmente tienen como resultado rutas más densas, y por ende, con menores costos por paquete transportado.

Asimismo, el proyecto queda abierto como posibles derivadas de desarrollo, como lo es seguir configurando los hiperparámetros de tal forma que se pueda seguir mejorando los indicadores de eficiencia en las rutas, como también generar un modelo más robusto para estimar el *Service Time* con menores errores, o incluso integrar el origen de los camiones a las rutas.

Bibliografía

[1] Song, J., Wen, R., Xu, C., y Tay, J. W. E., "Service time prediction for last-yard delivery," en 2019 IEEE International Conference on Big Data (Big Data), pp. 3933–3938, 2019, doi:10.1109/BigData47090.2019.9005585.

[2] Toth, P. y Vigo, D., The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, 2002, doi:10.1137/1.9780898718515.

[3] Brucato, C., The Traveling Salesman Problem. 2013, http://d-scholarship.pitt.edu/18770/.

[4] Laporte, G., "The traveling salesman problem: An overview of exact and approximate algorithms," European Journal of Operational Research, vol. 59, no. 2, pp. 231–247, 1992, doi:https://doi.org/10.1016/0377-2217(92)90138-Y.

[5] Flood, M. M., "The traveling-salesman problem," Operations Research, vol. 4, no. 1, pp. 61–75, 1956, http://www.jstor.org/stable/167517 (visitado el 2022-11-28).

[6] Blazinskas, A., "Combining 2-opt , 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem," 2011.

[7] Blocho, M., "Chapter 4 - heuristics, metaheuristics, and hyperheuristics for rich vehicle routing problems," en Smart Delivery Systems (Nalepa, J., ed.), Intelligent Data-Centric Systems, pp. 101–156, Elsevier, 2020, doi:https://doi.org/10.1016/B978-0-12-815715-2.00009-9.

[8] Voudouris, C., Tsang, E., y Alsheddy, A., Guided Local Search, pp. 321–361. 2010, doi:10.1007/978-1-4419-1665-5_11.

[9] Blum, C. y Roli, A., "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," vol. 35, no. 3, 2003, doi:10.1145/937503.937505.

[10] Tsang, E. y Voudouris, C., "Fast local search and guided local search and their application to british telecom's workforce scheduling problem," Operations Research Letters, vol. 20, no. 3, pp. 119–127, 1997, doi:https://doi.org/10.1016/S0167-6377(96)00042-9.

[11] Glover, F., "Tabu search—part I," ORSA j. comput., vol. 1, no. 3, pp. 190–206, 1989.

[12] Kirkpatrick, S., Gelatt, C. D., y Vecchi, M. P., "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671–680, 1983, doi:10.1126/science.220.4598.671.

[13] Jozefowiez, N., Semet, F., y Talbi, E.-G., "An evolutionary algorithm for the vehicle routing problem with route balancing," European Journal of Operational Research, vol. 195, no. 3, pp. 761–769, 2009, doi:https://doi.org/10.1016/j.ejor.2007.06.065.

[14] Cheng, D., Gkountouna, O., Züfle, A., Pfoser, D., y Wenk, C., "Shortest-path diversification through network penalization: A washington dc area case study," en Proceedings

of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS'19, (New York, NY, USA), Association for Computing Machinery, 2019, doi:10.1145/3357000.3366137.

[15] Perron, L. y Furnon, V., "Or-tools.", https://developers.google.com/optimization/.

[16] Dijkstra, E. W., "A note on two problems in connexion with graphs," Numerische mathematik, vol. 1, no. 1, pp. 269–271, 1959.

[17] Belka, R. y Godlewski, M., "Distance matrix generation for dynamic vehicle routing optimization in transport fleets management," en Artificial Intelligence and Sustainable Computing (Pandit, M., Gaur, M. K., Rana, P. S., y Tiwari, A., eds.), (Singapore), pp. 77–89, Springer Nature Singapore, 2022.

[18] Mann, H. B. y Whitney, D. R., "On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other," The Annals of Mathematical Statistics, vol. 18, no. 1, pp. 50 – 60, 1947, doi:10.1214/aoms/1177730491.