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

MODELO DE CLUSTERIZACIÓN DE LAS INSTANCIAS DE OPTIMIZACIÓN PARA LA SELECCIÓN AUTOMÁTICA DE PARÁMETROS DE LA HEURÍSTICA DE OPTIMIZACIÓN USADA POR LA EMPRESA SIMPLIROUTE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INDUSTRIAL

IGNACIO SEBASTIÁN CONTRERAS CÁCERES

PROFESOR GUÍA:
RODOLFO URRUTIA URIBE

MIEMBROS DE LA COMISIÓN:
PATRICIO CONCA KEHL
ÁLVARO ECHEVERRÍA SOLÍS

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL
POR: IGNACIO SEBASTIÁN CONTRERAS CÁCERES
FECHA: 2020
PROF. GUÍA: RODOLFO URRUTIA URIBE

MODELO DE CLUSTERIZACIÓN DE LAS INSTANCIAS DE OPTIMIZACIÓN PARA LA SELECCIÓN AUTOMÁTICA DE PARÁMETROS DE LA HEURÍSTICA DE OPTIMIZACIÓN USADA POR LA EMPRESA SIMPLIROUTE

SimpliRoute es una StartUp con cinco años de experiencia en el mercado, la cual busca democratizar la resolución de los problemas de planificación de despacho que puede tener cualquier tipo de empresa. El producto principal que ha desarrollado, es una heurística de optimización que permite resolver el problema conocido como VRP (Vehicle Routing Problem). Este problema permite encontrar la ruta óptima que debe realizar una flota de vehículos para atender a sus clientes. El VRP es un problema con complejidad *NP-Hard*, lo que se traduce en que encontrar soluciones óptimas sea una tarea extremadamente difícil.

En el contexto de la resolución del problema de ruteo de vehículos (VRP), el área de Data Science ha identificado que la heurística de optimización que actualmente es usada por SimpliRoute está siendo utilizado para resolver una gran variedad de problemas de planificación. Ahora bien, dado que la heurística de optimización no encuentra soluciones exactas sino que en base a reglas y criterios predefinidos encuentra soluciones que buscan acercarse al óptimo, es difícil pensar que la heurística pueda tener un buen desempeño al resolver para toda la variedad de instancias.

Lo anterior se traduce en una problemática para la empresa, ya que la heurística actual no es capaz de encontrar las mejores soluciones para todos los casos, por lo que en este trabajo se plantea una solución que permita a la plataforma caracterizar el problema a resolver y a partir de dicha caracterización determinar aquella configuración de la heurística de optimización que genera que esta tenga el mejor desempeño para ese problema de planificación.

Mencionado lo anterior, el objetivo general de este trabajo es *generar un clasificador de instancias que entregue la mejor configuración de la heurística de optimización utilizado por SimpliRoute de modo de mejorar el desempeño del solver*. Para abordar este objetivo, fue necesario definir una caracterización de las instancias y en base a esta se entrenaron diferentes modelos para realizar la predicción de la mejor configuración. De forma empírica, comparando dos configuraciones distintas, se encontraron casos en que la solución encontrada puede llegar a mejorar hasta en un 80 %. En promedio, se puede llegar a mejorar la solución entre un 5-25 %. El modelo que escoge siempre la configuración con mejor desempeño logra una mejoría promedio de un 2.26 %.

Los resultados de este trabajo no fueron favorables, ya que no se obtuvo un modelo de predicción que resolviese la problemática. Sin embargo, se pudo reafirmar que la problemática planteada en este trabajo de título es relevante y merece la pena buscar otro enfoque para resolverlo, así como también se obtuvieron ideas importantes a considerar tanto para el funcionamiento de la heurística de optimización como para volver a abordar esta problemática, principalmente relacionado con el manejo de los datos.

A mis padres y hermana por su amor e incondicionalidad, también por enseñarme a ser crítico y cauto de la realidad que nos rodea.

A la Pochoclita por apañarme en todo momento pero principalmente por siempre estar motivándome a sacar lo mejor de mí.

A Ferdi por ser un tremendo amigo y por llevarme siempre a pensar más allá.

Al cariño de los amigos de la U.

Al equipo de SimpliRoute por apoyarme y confiar en este proceso.

A los profesores Rodolfo Urrutia y Patricio Conca por el tiempo y la guía brindada.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

El presente trabajo de memoria se desarrolla en el contexto de la resolución del problema del ruteo de vehículos (*VRP*[1], por sus siglas en inglés) realizado por la empresa SimpliRoute. El problema de ruteo de vehículos ha sido fuertemente estudiado en el mundo académico, en especial por el área de la Investigación Operativa (Laporte 2009). Lo que ha motivado que este problema sea tan estudiado es que la resolución de este problema puede traer grandes beneficios a las organizaciones, sin embargo, dicho problema tiene una alta complejidad de resolución, lo que hace que sea muy difícil encontrar soluciones y a su vez soluciones óptimas.

Este problema fue presentado por primera vez en el trabajo realizado por Dantzig & Ramser en el año 1959. Lo que los motivó a plantear esta problemática era encontrar la ruta óptima que debe realizar una flota de camiones desde su origen y diversas estaciones de servicio, de modo de minimizar la distancia recorrida. En dicho trabajo no se realizaron aplicaciones prácticas del problema resuelto, sin embargo, se puede inferir directamente que si disminuye la distancia recorrida por los camiones se disminuye el costo por concepto de combustible.

El principal beneficio que obtienen las empresas al aplicar y resolver el VRP en sus operaciones es que logran disminuir sus costos logísticos, ya que en primera instancia se puede lograr disminuir el tamaño de la flota y disminuir la cantidad de kilómetros recorridos, ahorrándose costos asociados al mantenimiento de los vehículos y costos de combustible, respectivamente.

Para que las empresas puedan capturar ese beneficio deben ser capaces de resolver el problema, el cual en la práctica no es sencillo de resolver (Laporte 2009). Por lo cual, usualmente recurren a *softwares* dedicados a resolver este problema. Estos *softwares* solían tener un costo monetario alto, lo que hacía difícil adquirirlos, sin embargo, en el último tiempo han ido apareciendo nuevos actores, quienes aprovechándose del avance de la tecnología, han podido ofrecer el servicio de ruteo a un costo menor y con soluciones de calidad, es así como nace SimpliRoute, empresa donde se desarrolla el presente trabajo.

SimpliRoute es una *StartUp* chilena fundada en el año 2015, dedicada principalmente a ofrecer un *software* de ruteo de vehículos. Desde sus inicios hasta la fecha, SimpliRoute ha mostrado tener relevancia en el mercado, alcanzando a tener clientes en 22 países del mundo.

Cuando el problema de ruteo modela situaciones reales, encontrar la planificación óptima puede llegar a ser muy complejo, por lo que SimpliRoute diseñó su propia metodología (*i.e.*,

[1]VRP: Vehicle Routing Problem.

heurística) de resolución en base a diferentes *heurísticas*. Con esta heurística de optimización han podido ofrecer sus servicios, llegándole a generar ahorros en los costos logísticos a sus clientes de en promedio un 30 %. El *software* de ruteo de SimpliRoute es ofrecido como un servicio a través de internet, es decir, un *SaaS*[2].

Sin embargo, con el paso del tiempo, la empresa ha ido adquiriendo nuevos clientes en diferentes rubros y partes del mundo, a diciembre de 2019 SimpliRoute tenía clientes en 22 países distintos. El hecho de contar con esta variedad de clientes produce que SimpliRoute esté resolviendo problemas de optimización muy variados entre sí y toda esta variedad está siendo resuelta por una única heurística de optimización.

Dado que la heurística de optimización de SimpliRoute está construido en base a heurísticas, lo descrito anteriormente se traduce en un problema. La razón de esto es que las heurísticas son una serie de reglas que permiten encontrar buenas soluciones de acuerdo al conocimiento que se tenga del problema (Burke et al. 2019). Entonces, al ser una serie de reglas pre-definidas, resulta difícil pensar que estas reglas funcionarán bien para una gran variedad de instancias.

Esta problemática en particular es la que motiva el desarrollo de este trabajo de memoria, en el cual se busca lograr que la heurística de optimización tenga diferentes comportamientos ajustado a las características propias de las instancias de optimización que desean resolver los clientes. La razón de buscar que la heurística tenga diferentes comportamientos es lograr que este puede encontrar mejores soluciones, es decir, tenga un mejor desempeño.

Para llevar a cabo lo anterior, se busca generar un modelo que permita *clusterizar* los problemas de optimización y para cada uno de estos clusters ajustar la heurística de optimización de manera tal que este logre su mejor desempeño. El ajuste de la heurística de optimización se realiza mediante la modificación de ciertos parámetros que controlan el comportamiento del mismo, por lo que el objetivo principal de este trabajo es, mediante un modelo, *clusterizar* los problemas de optimización y para cada cluster entregar la mejor configuración de parámetros de la heurística de optimización, para así lograr el mejor desempeño de esta.

La estructura de este trabajo de memoria es la siguiente: en el capítulo 1, se presentan los antecedentes generales de la empresa. En el capítulo 2, se define formalmente la problemática a abordar. En los capítulos 3 y 4, se presentan los objetivos y metodología de trabajo respectivamente. En el capítulo 5, se presenta el marco conceptual con el que se desarrolla este trabajo. En el capítulo 6, se justifica la selección de los parámetros a configurar por el modelo propuesto en este trabajo y se explica brevemente el funcionamiento de la heurística de optimización. En el capítulo 7, se lleva a cabo el procesamiento de los datos, tras el cual se conforma la base de instancias con las que se entrenará el modelo. En el capítulo 8, se plantea la caracterización y se realiza un análisis exploratorio de las instancias presentes en la base de datos. Finalmente en los capítulos 9 y 10, se presentan y discuten los resultados obtenidos, respectivamente. Para finalmente en el capítulo 11, concluir.

[2] *SaaS: Software as a Service*. En estos casos, lo que se ofrece es el acceso a un *software* como si este fuese un servicio. La particularidad de esta forma de ofrecer el acceso al *software* es que el dueño del servicio se encarga de mantener los recursos computacionales, lo que se traduce en un beneficio para el cliente, ya que este sólo se encarga de hacer uso del servicio, sin preocuparse de los aspectos técnicos.

Capítulo 1

Antecedentes Generales

1.1. Historia de SimpliRoute

SimpliRoute es una *StartUp* dedicada a ofrecer un *software* que permite resolver el problema de ruteo, conocido como *Vehicle Routing Problem* (VRP). Haciendo uso de este *software*, las empresas pueden lograr disminuir sus costos de operación, específicamente, los relacionados a los despachos que deben realizar.

La empresa se origina después que los fundadores se dieran cuenta que las organizaciones y/o las empresas debían realizar grandes esfuerzos para generar rutas inteligentes al momento de efectuar sus despachos, es decir, encontrar una solución al problema de ruteo. Las empresas que deben realizar despachos como parte de sus operaciones, desean que estos sigan *rutas inteligentes*, ya que de esta manera pueden maximizar el nivel de servicio, es decir, atender a la mayor cantidad de clientes cumpliendo con su promesa de servicio, minimizando los costos implicados en el despacho.

Cuando se dice *rutas inteligentes* se hace referencia a que con los mismos recursos que antes se usaban para realizar X visitas, mediante una *ruta inteligente* se pueden hacer X+Y visitas.

Para poder determinar cuáles son las rutas inteligentes, las empresas tienen dos opciones. La primera opción es desarrollar y resolver sus propios modelos de optimización, lo que puede llegar a ser muy caro y extenuante en términos del uso de los recursos, debido a la complejidad del VRP (Savelsbergh 2008). Una segunda opción que tienen, es resolver su problema haciendo uso del *softwares* comerciales elaborados por terceros, quienes entregan la planificación de las rutas de forma sencilla.

Esta última opción solía tener un alto costo monetario, debido a que las empresas desarrolladoras requerían de complejas estructuras para poder ofrecer sus servicios. Usualmente las empresas vendían licencias para hacer uso de sus *softwares*, estas licencias tenían un costo aproximado de entre USD$20.000 a USD$40.000, con un límite de planificación de hasta 50 rutas por ejecución (Partyka & Hall 2012).

Sin embargo, gracias a los avances que ha tenido la tecnología, ha permitido que las empresas de *softwares* puedan ofrecer su producto como un servicio. Es así como SimpliRoute ofrece un *SaaS*[3] de ruteo de vehículos. Esta forma de ofrecer el servicio, permite que el negocio sea flexible, en términos de costos, y sea altamente escalable, ya que una característica de esta modalidad es que le permite al negocio tener una estructura de costo donde principalmente se tienen costos variables.

Otra característica relevante del servicio de SimpliRoute es que es ofrecido completamente a través de internet por lo que los clientes pueden acceder al servicio de forma sencilla y sin necesitar de sofisticados recursos computacionales.

El *software* que ofrece SimpliRoute busca que sus clientes puedan obtener las rutas óptimas de una forma sencilla y a un bajo costo. En base a esto, SimpliRoute definió su misión como **entregar a las empresas de logística y de servicios, la mejor herramienta para mejorar su operación logística diaria** y su visión es **convertirse en líder mundial de herramientas de inteligencia para la logística, innovando y creando nuevos servicios para todas las empresas, independiente de su tamaño y complejidad**.

A lo largo de los años han ido mostrando relevancia en el mercado. En enero del año 2015 ganan el premio *StartUp Chile*, lo que les permite financiar sus primeros desarrollos. En agosto del mismo año, SimpliRoute es reconocida en el evento *eCommerce Startup Competition* con una mención especial por ser una de las *StartUps* chilenas con más impacto en el mercado digital en Chile. En octubre del mismo año comienza su proceso de internacionalización al ser aceptados en la incubadora *500 Startup*.

De 2015 a 2018, SimpliRoute aumentó sus clientes de 14 a 150, para luego en Junio de 2019 alcanzar 205 clientes en 22 países, entre los que se destacan Argentina, Brasil, Costa Rica, Colombia, España, Australia y Gran Bretaña. También en el año 2019 abren oficinas propias en Perú y México, mientras que en los otros países de latinoamérica cuenta con *partners*, es decir, socios que ofrecen los servicios en representación de SimpliRoute. En los países fuera del continente americano, se ha llegado a los clientes completamente a través de la web.

Con respecto al tamaño de la empresa, en Septiembre del 2018 facturó aproximadamente USD$55.000, con un crecimiento de un 13 % mensual, lo que se traduce en USD$820.000 anuales, según lo mostrado en una de las rondas de financiamiento organizadas por TechCrunch y FBStart (fig. 1.1).

1.2. Servicio ofrecido por SimpliRoute

SimpliRoute se encuentra inmersa en la industria de la venta de servicios y del desarrollo de *softwares*. Estos desarrollos buscan apoyar y mejorar los procesos logísticos de las empresas, principalmente enfocado en los procesos de despacho. SimpliRoute se cataloga co-

[3] *SaaS: Software as a Service*. En estos casos, lo que se ofrece es el acceso a un *software* como si este fuese un servicio. La particularidad de esta forma de ofrecer el acceso al *software* es que el dueño del servicio se encarga de mantener los recursos computacionales, lo que se traduce en un beneficio para el cliente, ya que este sólo se encarga de hacer uso del servicio, sin preocuparse de los aspectos técnicos.

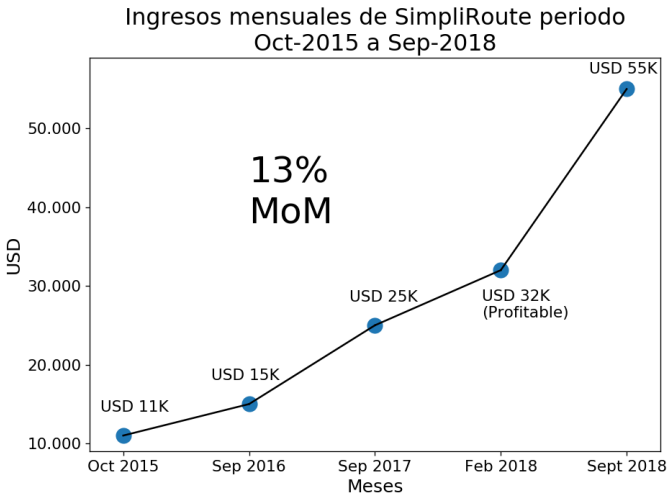


Figura 1.1: Ingresos mensuales de SimpliRoute en el tiempo (Echeverria & Shats 2018). Adaptación realizada a partir de lo mostrado en la presentación.

mo una empresa B2B, ya que la oferta de sus servicios se encuentra orientada a empresas, independientemente de su tamaño (pequeñas, medianas y grandes).

Cabe destacar, que el *core* del negocio de SimpliRoute, es ofrecer un servicio de planificación de las rutas de despacho, mediante la optimización de estas rutas. En esta línea, la propuesta de valor de SimpliRoute es: *generar ahorros de un* 30 % *en los costos de logística de nuestros clientes*.

De manera adicional, se ofrecen servicios complementarios, los cuales permiten que quien tenga la labor de planificar las rutas pueda tener un mayor control de sus procesos de despacho. Entre estos servicios adicionales, se encuentra el monitoreo de los vehículos en tiempo real, despliegue de reportes para la gestión, entre otros.

Es importante recalcar que estos servicios son ofrecidos a sus clientes mediante la modalidad de *SaaS*, es decir, para acceder a los servicios de SimpliRoute, sólo se debe tener acceso a internet. Esta modalidad facilita que los clientes puedan contratar este tipo de servicios de logística, ya que disminuye los requerimientos técnicos con los que debe contar para llevar a cabo la optimización.

1.3. Servicio de optimización

Hasta ahora, se ha mencionado que SimpliRoute ofrece un servicio de optimización, el cual permite que las empresas puedan planificar sus rutas de manera *inteligente* y así reducir sus costos logísticos. Sin embargo, no se ha mencionado cuál es el problema que tienen las empresas que hace que requieran de un servicio que les determine las rutas. En esta sección se describirá a grandes rasgos el problema de la planificación.

En toda organización o empresa que debe realizar despachos, existe una persona que se

encarga de determinar los despachos que se deben realizar durante el día o un periodo de tiempo determinado, esta tarea puede llegar a ser muy compleja para ser resuelta a mano.

La complejidad de la tarea anterior está dada por la gran cantidad de variables que debe tener en consideración quien planifica las rutas, entre las variables más comunes a considerar están: el nivel de servicio que desea brindar la empresa, la jornada laboral de los transportistas, la capacidad de los vehículos, el tiempo de viaje entre los puntos, la prioridad de atención de los clientes, etc. En este escenario, tomar una decisión que le permita atender la mayor cantidad de puntos de forma de mantener los costos de este proceso lo más bajos posible, respetando las restricciones del negocio y del problema, resulta bastante difícil. Además, a medida que crece el número de puntos a atender y las reglas de negocio que se deben considerar, es exponencialmente más difícil tomar una buena decisión.

En este escenario, donde la tarea del planificador se vuelve extremadamente difícil aparecen los *softwares* de optimización, que hacen uso de los recursos computacionales para evaluar la mayor cantidad de escenarios posibles y así determinar cuál de estos es el mejor.

A continuación se mostrará un ejemplo práctico resuelto por SimpliRoute. En este ejemplo de planificación, se desea realizar 16 despachos a distintos puntos de la ciudad de Santiago y se dispone de una flota de 5 vehículos. Cada punto tiene una determinada demanda y cada vehículo tiene una determinada capacidad. La demanda está expresada en términos de unidades de capacidad, es decir, cuántas unidades de capacidad demanda cierto cliente.

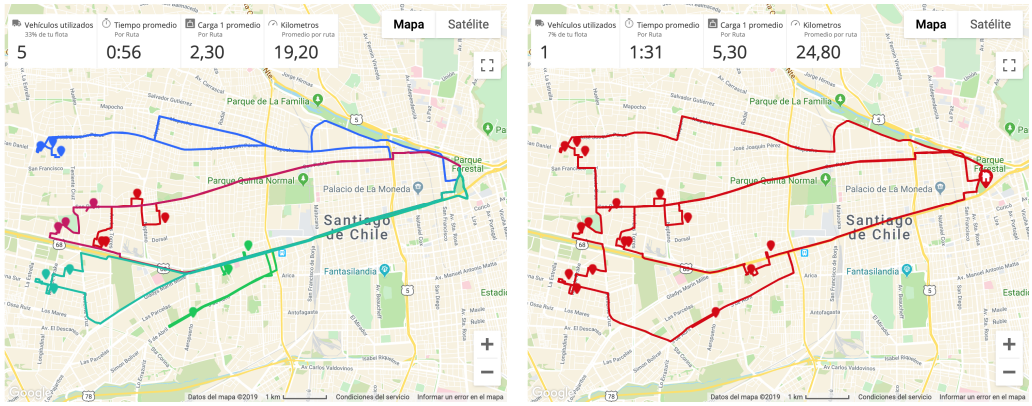
Ahora bien, a pesar de ser pocos puntos y muchos vehículos, la solución no resulta sencilla de calcular. En este caso en particular, la demanda de cada cliente es de 1 unidad de capacidad, 4 de 5 vehículos tienen una capacidad de 1 y el quinto vehículo tiene una capacidad de 6.

Entonces, dado el problema de optimización a resolver, en la figura 1.2 se plantean dos posibles soluciones, las cuales serán analizadas a continuación.

Para construir la primera solución (fig. 1.2a), se consideró utilizar todos los vehículos, la razón de tomar esta decisión puede haber sido atender a todos los clientes temprano o simplemente utilizar todos los recursos disponibles, es decir, cargar todos los camiones al comienzo de la jornada y despachar. Esta solución requiere que se realicen varios viajes, ya que la capacidad acumulada de los vehículos es de 10 y la demanda total es de 16. Entonces, en la solución mostrada en la figura 1.2a se requiere que se realicen 7 viajes (rutas) en total, donde en promedio cada viaje se tarda 56 minutos y en total se requieren aproximadamente 7 horas. También, en promedio, en cada viaje se recorren 19.2 kilómetros, lo que en total se traduce en 134 kilómetros recorridos.

Por otro lado, en la figura 1.2b se hizo uso del servicio de optimización de SimpliRoute, logrando atender la misma cantidad de puntos, usando menos camiones, demorándose menos tiempo y recorriendo menos kilómetros. Esta solución requiere que sólo un vehículo realice tres viajes (rutas), en cada viaje en promedio se requieren 1:31 hrs., lo que en total se traduce en 4:30 hrs. y en términos de kilómetros se recorren en total 74.

Al comparar estas dos soluciones, resulta directo darse cuenta que para este caso, utilizar



(a) Solución al realizar una planificación tratando de usar todos los recursos disponibles.

(b) Solución entregada por el servicio de optimización de SimpliRoute.

Figura 1.2: Comparación de dos soluciones al problema de planificación.

el servicio de optimización de SimpliRoute le puede traer beneficios a este cliente, ya que para este tipo de problemas de planificación se requiere usar menos vehículos, requiere que los conductores pasen menos tiempo en la calle y además puede ahorrar combustible, porque se recorren menos kilómetros.

Finalmente, queda mostrado que el problema al que se enfrenta el planificador no es un problema sencillo de resolver y hacer uso de un servicio de optimización le puede permitir agilizar el proceso de planificación de rutas y además, obtener soluciones que permitan maximizar el número de despachos y/o minimizar sus costos de transporte.

Capítulo 2

Definición y justificación del problema

En este capítulo se definirá y describirá la problemática abordada en este trabajo, para luego poder mostrar el impacto de la misma.

Para llevar a cabo lo anterior, el capítulo comienza describiendo el contexto en el que está inmerso el problema, para luego describir y justificar la problemática. Finalmente, se mostrará el impacto que tiene el problema abordado.

2.1. Contexto

El servicio de optimización de SimpliRoute está compuesto por los tres macroprocesos mostrados la figura 2.1, los cuales son descritos a continuación.

El primer macroproceso, denominado *definición del problema a optimizar*, ocurre en la plataforma web de SimpliRoute. En este proceso, el cliente entrega la información de los despachos que desea realizar, definiendo los puntos a visitar con sus respectivas características (ubicación, demanda, ventana horaria, etc.), la flota con la que cuenta y sus características (lugar de partida, capacidad, jornada de trabajo, etc.) y otros parámetros a considerar en la optimización, como por ejemplo si desea minimizar los vehículos o no. A esta información se le denomina **instancia de optimización** o simplemente **instancia**. Cuando finaliza este proceso, la instancia es enviada al optimizador.

El segundo macroproceso, denominado *optimización*, ocurre directamente en el servicio de optimización de SimpliRoute. En este proceso se recibe el problema a optimizar, es decir, la instancia y se resuelve, entregando una solución. Cabe recordar que el problema que se resuelve es el de la planificación de las rutas de despacho, a este problema se le conoce académicamente como *Vehicle Routing Problem* (VRP). Cuando finaliza este proceso, la solución es enviada al siguiente proceso.

Finalmente, el último macroproceso se denomina *editado y guardado del plan*. En este proceso se recibe la solución desde el optimizador y se muestran gráficamente, en la plataforma

web, las rutas que deben hacer los vehículos. Además, en caso que el usuario lo desee, puede editar las rutas manualmente. Este proceso finaliza con el guardado de las rutas, de esta forma la planificación de las rutas queda a disposición del cliente para que haga uso de ellas.

Ya descritos los macroprocesos del servicio de optimización es posible indicar dónde está inmersa la problemática abordada en este trabajo de memoria. La problemática abordada ocurre en el proceso de *optimización*. En las siguientes sub-secciones se explicará el problema de optimización que resuelve SimpliRoute.

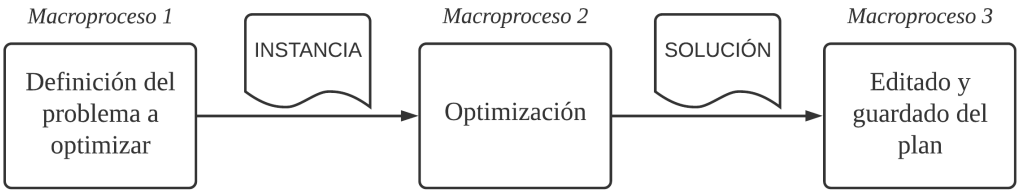


Figura 2.1: Diagrama de los macroprocesos involucrados en el servicio de optimización de SimpliRoute.

2.1.1. Vehicle Routing Problem

Como se ha mencionado en las secciones anteriores (1.2 y 1.3), el *core* del negocio de la empresa es realizar la planificación de las rutas de despacho de sus clientes, problema conocido académicamente como *Vehicle Routing Problem* (VRP).

El VRP es un problema de optimización, cuyo objetivo es determinar las rutas óptimas que debe realizar una flota de vehículos para atender un determinado conjunto de destinos.

Este problema fue presentado por primera vez en el trabajo realizado por Dantzig & Ramser (1959). En este trabajo se plantea una de las variantes más sencillas del VRP, ya que considera un conjunto de n puntos a atender, conexos entre sí, y una flota de un solo vehículo, y se debe cumplir con la única restricción de no superar la capacidad del vehículo. Con el paso de los años, se fueron planteando nuevas variantes del VRP, lo que se traduce en la incorporación de nuevas restricciones al modelo, lo que permitió ir modelando la realidad de una mejor manera.

Una de las particularidades del VRP es lo complejo que es de resolver, habiéndose demostrado que es un problema con una complejidad *NP-Hard* -se explica en detalle en la sección 5.1.1- (Lenstra & Rinnooy 1979). Esta característica del problema hace que sea extremadamente complejo resolverlo, los que se traduce en que se requiere de mucha capacidad computacional y/o tiempo para obtener las soluciones exactas. Además, el hecho de que se resuelva el VRP en alguna de sus distintas variantes, es decir, con más restricciones, hace que sea cada vez más difícil de encontrar una solución.

Para ejemplificar lo complejo que puede llegar a ser resolver el VRP, se analizará una de las versiones más sencillas, la formulación realizada por Dantzig & Ramser (1959). En este

trabajo, se identifica que el número total de soluciones posibles crece de forma factorial según el número de puntos a visitar, es decir, dado un conjunto de n puntos conexos entre sí y una flota de un vehículo, existen $\frac{1}{2}n!$ soluciones posibles.

Dada la complejidad de resolución del VRP, se han diseñado distintos tipos de algoritmos para poder encontrar una solución al problema. Estos algoritmos se pueden dividir en tres tipos, en los *algoritmos exactos*, las *heurísticas* y las *metaheurísticas*, descritos a continuación.

- **Algoritmos exactos:** Este tipo de algoritmos logran encontrar la solución óptima al problema, pero dada la complejidad, resulta poco factible usarlos en instancias reales.
- **Heurísticas:** Este tipo de algoritmos consisten en una serie de reglas que permiten encontrar buenas soluciones, para las situaciones que fueron diseñadas (Burke et al. 2019). Las heurísticas usualmente encuentran óptimos locales.
- **Metaheurísticas:** Este tipo de algoritmos consisten en relajaciones a la búsqueda exacta de la solución, lo que permite realizar una mayor exploración del espacio de soluciones. Usualmente este tipo de algoritmos encuentra buenas soluciones pero requieren de una gran capacidad computacional (Bonet 2018).

En el capítulo 5 se explica con mayor detalle el problema de ruteo de vehículos, su complejidad, sus variantes y los métodos de resolución.

2.1.2. Heurística de optimización

La heurística que ha elaborado SimpliRoute le permite ofrecer a sus clientes tres opciones de funciones objetivo distintas. La primera función objetivo busca minimizar el **tiempo de viaje** de los vehículos, la segunda función objetivo busca minimizar la **distancia** recorrida por los vehículos y la tercera función objetivo busca minimizar el **número de vehículos** que se utiliza.

Por otro lado, la heurística de optimización de SimpliRoute es capaz de resolver para diferentes variantes del problema de ruteo, entre las cuales se destacan las siguientes:

- Capacidad de carga, los vehículos tienen una capacidad limitada para cargar.
- Flota heterogénea, la flota está compuesta por distintos tipos de vehículos. Por ejemplo, una flota puede estar compuesta por motocicletas, autos y camiones.
- Rutas abiertas, los vehículos pueden o no terminar donde empezaron.
- Zonas geográficas, con esta variante cada vehículo se encuentra obligado a circular en una determinada zona geográfica, definida por los clientes.
- Habilidades especiales, en este caso hay visitas que sólo pueden ser atendidas por cierto tipo de vehículos, como por ejemplo clientes que reciben productos congelados, sólo pueden ser atendidos por vehículos que puedan llevar productos congelados.

- Prioridades, se le agrega una prioridad a las visitas. Esto permite definir clientes preferentes, de esta forma al diseñar las rutas, se preferirá atender a los clientes con mayor prioridad por sobre los que carecen de esta.

- Ventanas de tiempo, los clientes finales estipulan el rango horario en que pueden recibir el despacho, como por ejemplo, un cliente determina que puede recibir el despacho entre las 13:00 y las 15:00.

Las distintas variantes del VRP clásico se traducen en la incorporación de nuevas restricciones al modelo, de manera que el modelo refleje de mejor manera la realidad o se pueda mejorar el nivel de servicio de los despachos.

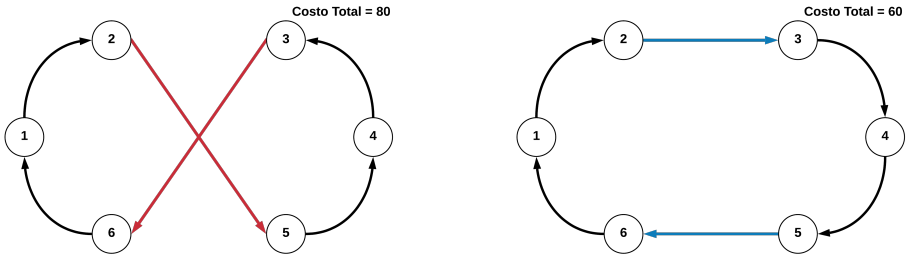
Con el paso del tiempo y la experiencia que ha ganado SimpliRoute en el mercado, la empresa ha logrado determinar que una de las características más valoradas por sus clientes al momento de optimizar y planificar sus rutas, es la de obtener una solución en el menor tiempo posible. Por lo anterior, la empresa se compromete a entregar una respuesta en menos de quince minutos. Para lograr buenos resultados en este tiempo, el algoritmo de optimización se ha desarrollado en base a heurísticas.

En la heurística de SimpliRoute se encuentran implementadas varias heurísticas. Entre estas heurísticas, las que más relevancia tienen son las heurísticas de Local Search, esta familia de metodologías permiten ir evaluando de forma local diferentes mejoras a la solución del problema de optimización. La más utilizada es la metodología denominada *2-exchange*, la cual consiste en lo siguiente: en primer lugar, se comienza con una solución factible y de ahí en adelante se van alternando los arcos de la ruta, luego de alternar los arcos, se evalúa si el nuevo escenario es mejor y se continúa probando, hasta que se acabe el tiempo o se cumpla el criterio de iteraciones, este criterio establece que si la solución no tiene una mejora significativa en las siguientes n iteraciones, entonces finaliza (González & Uribe 2018*a*).

Un ejemplo del funcionamiento de *2-exchange* se encuentra en la figura 2.2, donde los nodos están representados por círculos numerodas del 1 al 6. En la figura 2.2a se muestra una solución factible, luego mediante *2-exchange* se alternan los arcos (2, 5) y (3, 6) por los arcos (2, 3) y (5, 6) respectivamente. Como se puede apreciar, en este ejemplo, la solución 2.2b es mejor que la solución 2.2a debido a que el costo total de la nueva ruta es menor. La metodología de *2-exchange* sigue alternando los arcos hasta que se cumple algún criterio de terminación. Más detalle de esta metodología se encuentra en trabajo de Bräysy & Gendreau (2005).

También para el caso de las restricciones, en particular de la restricción de la ventana horaria, se utiliza una heurística de inserción, la cual consiste en que dada una ruta, se intenta insertar un nuevo punto de atención a la ruta. Para realizar esta inserción se debe verificar que para los nodos presentes en la ruta, ninguna de las restricciones del problema se rompan. En particular la restricción de la ventana horaria es muy costosa de calcular por lo que se utiliza una metodología basada en lo propuesto en el trabajo de Lu & Dessouky (2006), en donde para cada tramo de la ruta, se determina cuánto es posible atrasar dicho tramo de forma de no afectar a las demás entregas (González & Uribe 2018*b*).

En la figura 2.3 se muestra un ejemplo de esta metodología de inserción. Como se puede



(a) Solución factible con la que se comienza. Los arcos a evaluar son los (2, 5) y (3, 6).

(b) Los arcos (2, 5) y (3, 6) fueron alternados por los (2, 3) y (5, 6) respectivamente.

Figura 2.2: Ejemplo de la aplicación de la metodología de local Search *2-exchange*.

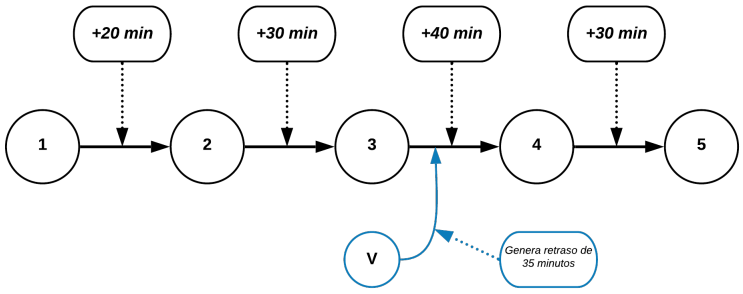


Figura 2.3: Ejemplo de la metodología de inserción utilizada por la heurística de optimización de SimpliRoute. Extraído de González & Uribe (2018*b*).

apreciar en dicha figura, para la ruta (1, 2, 3, 4, 5) se tiene calculado el tiempo máximo que se puede atrasar la llegada a cada punto. Por ejemplo, el tramo (2, 3) de la ruta indica que la llegada al punto 3 se puede atrasar hasta 30 minutos y aún así se va a cumplir con la ventana horaria de ese cliente. Entonces, si se quiere insertar una nueva visita v entre el nodo 3 y el 4, se debe calcular cuánto retrasa la atención de 4 el hecho de atender a v primero. Si este retraso es mayor a la holgura de alguno de los puntos sucesivos a 4, entonces no se puede insertar v en dicho tramo. En el ejemplo de la figura 2.3, insertar el nodo v entre 3 y 4 genera un retraso de 35 minutos, lo que rompe la ventana horaria del nodo 5, por lo que no es factible de insertar. Para más detalle del funcionamiento de esta metodología remitirse al trabajo de Lu & Dessouky (2006).

Es importante notar que el algoritmo que implementa estas heurísticas es determinista, lo que significa que dada una instancia y un contexto de resolución, siempre se producirá la misma solución.

Ya habiendo descrito el funcionamiento general de la heurística se procederá a comentar a grandes rasgos la arquitectura que tiene el hardware en el que está montada la heurística.

La arquitectura usada por SimpliRoute para ejecutar la heurística de optimización es bastante sencilla, esta cuenta de tres CPU con las mismas características. Cada una de estas CPU cuenta con un procesador *dual-proccesor dual core Intel(R) Xeon(R) Platinium* de 3.00

GHz, 4GB de RAM y un sistema operativo de 64-bit. Las CPU son coordinadas por un balanceador de carga, el cual está encargado de que cada una de las CPU tenga una carga de trabajo similar.

Cabe destacar que estas CPU las provee un tercero, el cual brinda un servicio de *cloud computing*. El beneficio que tiene esta arquitectura es que es flexible, en el sentido de que al aumentar la carga de trabajo se puede añadir nuevas CPUs sin complejidad y de forma rápida.

2.2. Definición del problema

Hasta ahora se han generado nociones básicas de en que consiste el problema de ruteo, su complejidad y formas de resolverlo. Además, en la sección 1.1 se mencionó que SimpliRoute atiende a clientes en distintas partes del mundo, generando que la heurística de optimización que utiliza SimpliRoute tenga resolver para una variedad de instancias. Con estos elementos es posible definir la problemática a abordar.

2.2.1. Problema a abordar

Actualmente, como muestra la figura 2.1, cada vez que un cliente requiere realizar la planificación en la plataforma de SimpliRoute, es decir, desea optimizar sus rutas de despachos, debe entregar una serie de detalles a ser considerados por la heurística, tales como vehículos disponibles, visitas a realizar y otras opciones de optimización, a este conjunto de información entregada por el cliente se le denomina **instancia** del problema a resolver.

Como ya se ha mencionado en secciones anteriores, la propuesta de valor de SimpliRoute está en **generar un 30 % de ahorro en los costos logísticos de los clientes**, además esta solución debe ser entregada en **menos de quince minutos**, debido a la alta valoración que tienen los clientes de esta característica.

Una de las particularidades de SimpliRoute, es que no restringe a sus clientes en el tamaño de las instancias a resolver, es decir, no existe un límite de vehículos que se pueden ingresar ni un límite de visitas a realizar, es más, la empresa constantemente está buscando generalizar y extender las características, de modo que se modele de mejor manera la realidad y el cliente sea quien define el tamaño de su problema.

Por lo anterior, uno de los mayores desafíos a los que se enfrenta SimpliRoute es el de entregar las mejores soluciones de planificación. Por lo que la gran cantidad de características y restricciones consideradas al momento de resolver el problema, hacen que acercarse al óptimo sea exponencialmente más difícil.

Es por esta razón que el desafío o problemática a la que se enfrenta SimpliRoute y que será abordada en este trabajo de título, es hacer que su heurística de optimización tenga el mejor desempeño para cada una de las instancias a resolver, sin la necesidad de desarrollar

nuevas y distintas heurísticas de optimización. En otras palabras,

> **Para la heurística de optimización que hoy utiliza SimpliRoute, es imposible encontrar las mejores soluciones para todos los tipos de instancias que debe resolver.**

Lo anterior quiere decir que dada la configuración actual que tiene la heurística, no es capaz de recoger toda la variedad de instancias y resolverlas de la mejor manera.

Para mostrar cómo afecta esta problemática a las soluciones que entrega la heurística de optimización, en la sección 2.3 se muestra que con una variación en la configuración de ciertos parámetros se puede lograr que un 43 % de las instancias estudiadas mejore su solución, logrando una mejoría promedio de un 4.8 %.

Las principales causas identificadas que produce esta problemática se encuentran resumidas en la figura 2.4 y son descritas a continuación.

La primera causa identificada corresponde a que la empresa cuenta con una **capacidad limitada de recursos computacionales**, en este sentido, el recurso computacional es un elemento clave al momento de buscar soluciones al problema de optimización, ya que al disponer de una capacidad de procesamiento mayor posible crear heurísticas que sean capaces de evaluar un mayor número de escenarios.

Una segunda causa identificada corresponde a la promesa de SimpliRoute de resolver el problema de planificación en un **tiempo limitado**. Recordar que SimpliRoute se compromete con sus clientes a entregar un resultado en menos de quince minutos, ya que el tiempo de resolución es una característica fundamental con la que los clientes evalúan a los optimizadores. Además, el tiempo de resolución también es un elemento clave, ya que si se dispone de más tiempo, lógicamente es posible evaluar más escenarios.

La importancia que tiene la capacidad de los recursos computacionales y el tiempo de resolución con los que se disponga, se debe al nivel de complejidad que tiene resolver el problema de ruteo (mencionada en la sección 2.1.1).

Finalmente, la tercera causa identificada corresponde a la **amplia variedad de instancias** que debe resolver la heurística. Como ya se ha mencionado, SimpliRoute no restringe a los clientes los problemas que pueden enviar a resolver, es decir, pueden ingresar una cantidad virtualmente ilimitada de puntos a visitar y de vehículos. Esto se traduce en que a la heurística de optimización llegue una amplia variedad de instancias.

En las secciones siguientes (2.2.2 y 2.3) se mostrará la presencia de variabilidad de las instancias y el efecto que tiene esta variabilidad en los resultados que entrega la heurística de optimización.

Es importante destacar que estas causas corresponden más bien a hechos con los que debe lidiar SimpliRoute y en general la industria y la competencia, debido a la gran dificultad que se tiene al intentar resolver el VRP. A los diferentes actores puede afectarle en mayor o menor medida, sin embargo, deben lidiar con las mismas causas.

Por otro lado se identificaron los siguientes efectos de la problemática mencionada. El primer efecto lo sufren los clientes directos de SimpliRoute, el cual tiene relación con que al no entregar las mejores soluciones, los clientes ven disminuida su capacidad de generar ahorros en costos y además, dificulta que ellos a su vez puedan cumplir con el nivel de servicio prometido a sus clientes.

El segundo efecto lo sufre directamente SimpliRoute, ya que el no entregar buenas soluciones, el nivel de satisfacción que tienen con las planificaciones disminuye, por lo que pueden perder clientes o su mercado objetivo se reducirá, debido a que la heurística no es capaz de satisfacer a todos los segmentos.

El tercer y último efecto identificado tiene estrecha relación con el segundo, donde al verse mermada la capacidad de entregar buenas soluciones les dificulta cumplir con la propuesta de valor de la empresa, lo que en el mediano plazo puede generar que la empresa vea afectado y/o reducido su negocio.

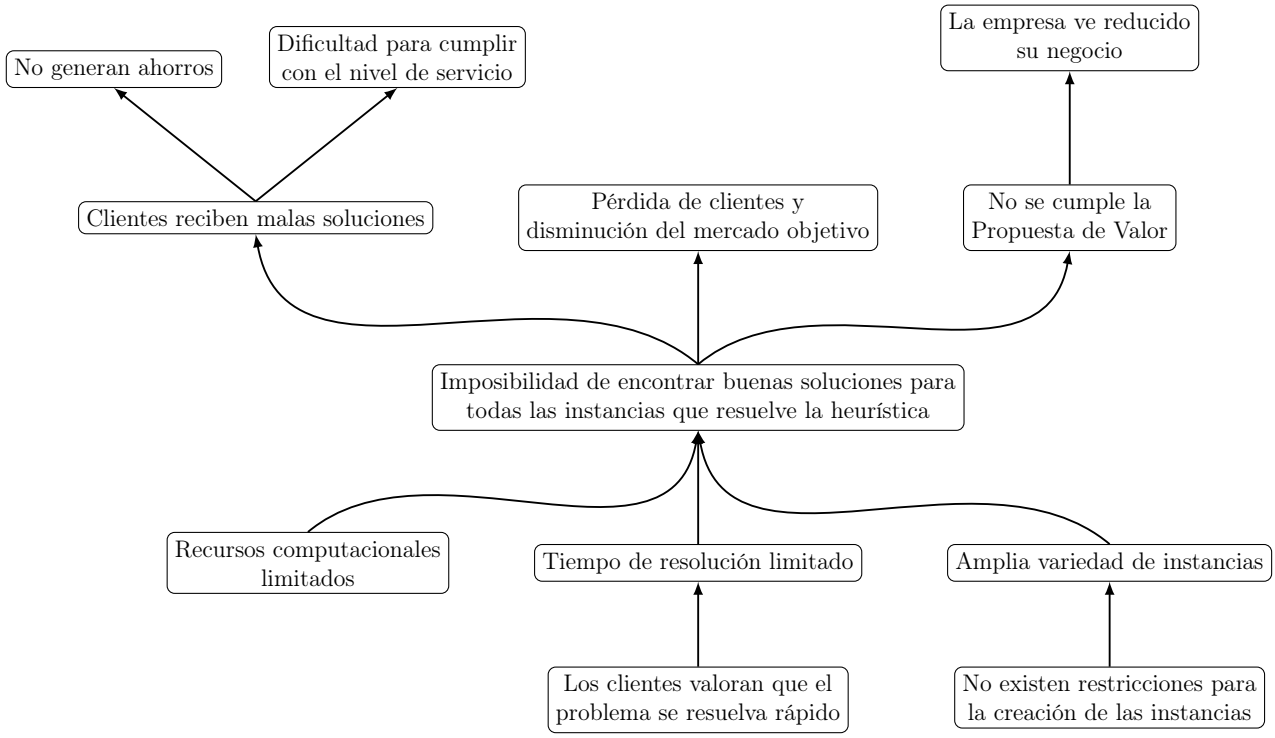


Figura 2.4: Árbol de Problemas.

2.2.2. Variabilidad de las instancias

Como se mencionó en la sección anterior, una de las causas de la problemática planteada es la variabilidad que existe en las instancias que debe resolver SimpliRoute, en particular, la heurística de optimización.

En esta sección se mostrará la presencia de variabilidad entre las instancias que resuelve SimpliRoute. Para eso es necesario tener en mente las siguientes definiciones.

- **Instancia o instancia de optimización:** Corresponde al conjunto de información entregada por los clientes, la cual resume el problema que desean optimizar.

- **Nodos**: Corresponde a los lugares que deben ser visitados para hacer entrega del producto/servicio.

- **Depots:** Corresponde a los lugares donde el vehículo comienza su recorrido.

Para realizar el siguiente análisis descriptivo, se cuenta con una base de datos con información de las instancias generadas por todos los clientes entre los meses de Mayo de 2016 y Enero de 2017. Esta base tiene un tamaño de 3.374 instancias y será utilizada solamente de forma preliminar. De cada una de estas instancias se tiene información referente a la flota de vehículos a rutear y sus características y de los clientes a visitar. Como primera aproximación para cuantificar el problema, se comenzará analizando y describiendo la información contenida en esta base.

Destacar que esta base de datos corresponde a la base que la empresa utiliza como base de prueba para realizar modificaciones y estudios a la plataforma, por lo que a pesar de estar desactualizada, se considera relevante para el análisis.

En la figura 2.5 se puede apreciar la gran variedad de instancias que tiene que resolver la heurística, característica que se mencionó como causa del problema anteriormente. De forma preliminar se analizarán características generales, tales como la cantidad de vehículos y puntos a visitar. También se considerarán dos características generales mencionadas en el trabajo de Rasku et al. (2017), las cuales corresponden a la posición del centro (*centroide*) de los puntos a rutear y la distancia promedio entre los clientes y el depot.

A partir del gráfico 2.5a se puede apreciar que la gran mayoría de las instancias que se resuelven, utilizan flotas relativamente pequeñas, en el 75 % de los casos se utilizan menos de 12 vehículos. Cabe recordar que a pesar de que sean flotas pequeñas, el problema sigue siendo complejo de resolver, ya que la mayor complejidad está dada por la cantidad y características de los nodos, como se mostró en la sección 2.1.1.

Observando el gráfico 2.5b se aprecia que la mayor concentración de los datos indica que mayoritariamente las instancias tienen entre $[0, 100]$ puntos a visitar. En particular, el promedio de puntos a visitar es de 72 y la mediana es de 55.

En el gráfico 2.5c se puede apreciar claramente que hay una gran cantidad de instancias pequeñas, cerca del 42 % de las instancias corresponde a problemas con menos de 40 nodos y menos de 2 vehículos. Un hecho importante que se desprende de este gráfico es que no es clara la tendencia que a mayor número de vehículos, mayor cantidad de puntos a visitar. Esto nuevamente muestra que existe una variabilidad entre las instancias.

Se calcularon dos características mencionadas por Rasku et al. (2017), las cuales permiten describir a las instancias. Estas fueron, la ubicación del *centroide* de los puntos a visitar y la distancia promedio entre el depot y los clientes, estas características se pueden apreciar en los gráficos 2.5e y 2.5d respectivamente.

En la figura 2.5e, el eje X representa la longitud y el eje Y representa la latitud del

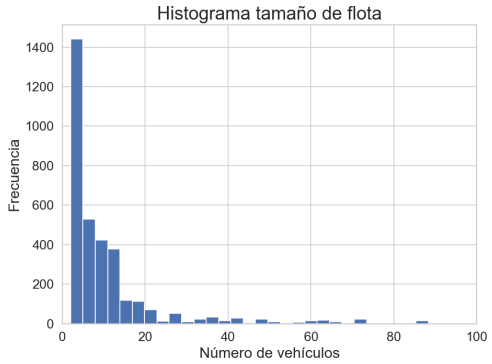
centroide. A partir de este gráfico, se puede apreciar que las instancias que llegan se originan en distintas partes del mundo, lo que hace que las instancias tengan una variabilidad generada según el lugar del mundo donde fueron originadas.

Para obtener la distancia entre dos puntos a partir de la latitud y longitud, se utilizó la fórmula de Haversine[4]. Se realizó un histograma (fig. 2.5d) de las distancia promedio entre los clientes y el *depot* para cada una de las instancias. En este gráfico se aprecia que no existe una clara densidad de agregación, lo que muestra que el set de instancias puede llegar a ser muy variable entre sí. Debido a que los valores de la distancia del histograma de la figura 2.5d se encuentran en el intervalo $[0, 6172]$, para evitar distorsionar el gráfico, se graficaron sólo los valores pertenecientes al intervalo $[0, 30]$, correspondiente al $91.5\,\%$ de los datos.

A partir de la información mostrada, se puede concluir que la heurística de optimización de SimpliRoute resuelve para una amplia variedad de instancias, las cuales tienen diferentes características entre sí.

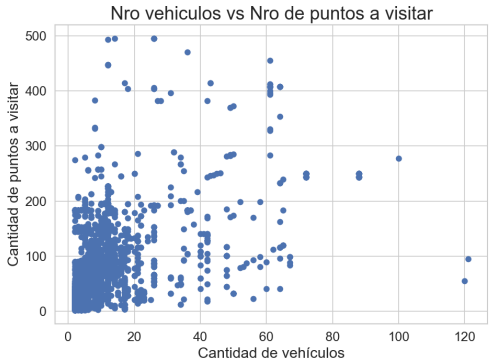
Ahora bien, de acuerdo a lo planteado por el teorema *No Free Lunch Theorem*, no se puede esperar que una heurística de optimización tenga buen desempeño en todos los tipos de instancias (Wolpert & Macready 1997). Aplicando este teorema a la heurística de SimpliRoute, se tiene que si las instancias que se resuelven son variadas, entonces la heurística de optimización no tiene el mejor desempeño en cada una de estas.

[4]Esta fórmula permite calcular la distancia que hay entre dos puntos en una esfera. Para este caso se utiliz'o el valor del radio de la tierra calculado en el Ecuador (R=6378Km)

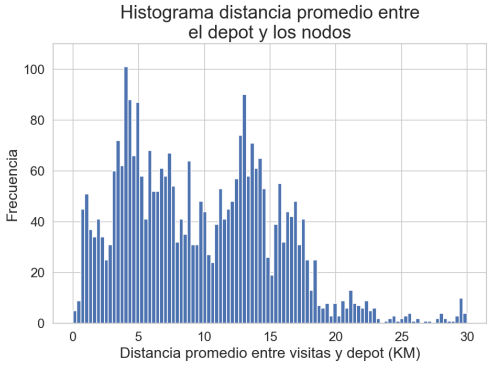


(a) Histograma del número de vehículos.

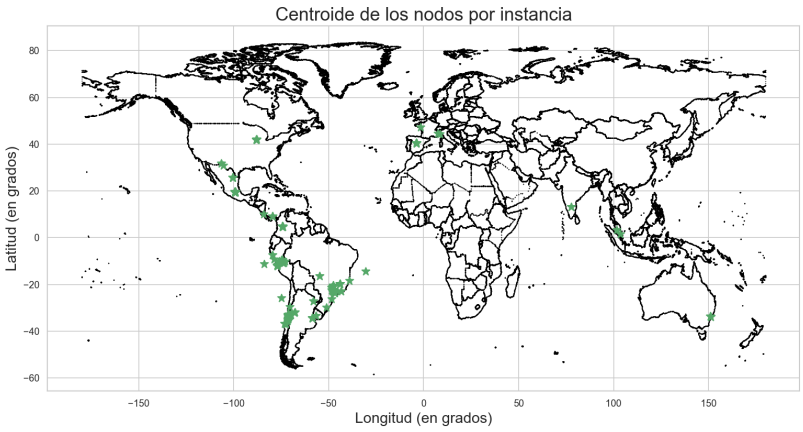
(b) Histograma del número de puntos a visitar.



(c) Gráfico de dispersión, número de vehículos versus número de clientes.



(d) Histograma distancia promedio entre clientes y depots.



(e) Posición del *centroide* de los nodos en el mapa.

Figura 2.5: Análisis descriptivo instancias de pruebas.

2.3. Efecto de la configuración de parámetros

Existe evidencia bibliográfica en donde se muestra que la modificación de los parámetros de ejecución de la heurística permite encontrar diferentes soluciones.

Como primera evidencia, Malitsky (2014) muestra que con tan solo escoger el *solver* correcto para la instancia que se está resolviendo, permite mejorar el desempeño en al menos un 20 %.

Luego, como segunda evidencia, Rasku et al. (2017) identifica que mediante la automatización de los parámetros se logra disminuir el GAP promedio que existe entre la mejor solución conocida para una instancia y el resultado entregado por un algoritmo. En algunos experimentos logra reducir el GAP promedio en un 60 % en comparación a la configuración *default* del algoritmo.

También, Demkes (2014) encuentra que mediante la automatización de parámetros se logra disminuir el número de vehículos utilizados en un 6 %(ver Demkes 2014, Cap 6, pp 55). Importante destacar que en dicho experimento, tanto la configuración *default* como la configuración determinada por la configuración automática de los parámetros, en las soluciones encontradas se logran asignar todas las tareas, sin embargo, la nueva configuración utiliza menos vehículos.

Por otro lado, para validar que la heurística de SimpliRoute tiene un comportamiento similar frente a diferentes configuraciones y que resulta relevante configurarla de manera correcta, se procedió a experimentar con configuraciones diferentes de la heurística de optimización. Las instancias utilizadas corresponden a todas las descritas en la figura 2.5, haciendo la distinción entre aquellas instancias con menos de 40 nodos a visitar, de aquellas que tienen 40 o más nodos.

En esta sección no se explicará en detalle el funcionamiento de la heurística de optimización de SimpliRoute, ya que la información necesaria será descrita en el capítulo 6. De todos modos, se describirá a grandes rasgos el rol que tienen los parámetros incluidos en este experimento.

Los parámetros que serán modificados en este experimento son tres, llamados *alpha*, *beta* y *gamma*. Estos están involucrados en la definición del punto de partida de la heurística.

Cuando comienza la ejecución de la heurística, el problema es dividido en subproblemas. Un subproblema se construye creando subconjuntos de nodos y vehículos a partir de los conjuntos originales contenidos en la instancia. Los nodos son escogidos siguiendo diferentes metodologías, las cuales serán explicadas en el capítulo 6. Luego, para asignar los vehículos a los subproblemas se determina el nivel de afinidad que existe entre los vehículos y los nodos, el objetivo de la asignación es maximizar las afinidades entre los vehículos y los nodos del subproblema. El nivel de afinidad corresponde a un puntaje ponderado de las siguientes características: (1) estimación del número de clientes que no se insertarán en el subproblema, (2) costo por uso del vehículo y (3) distancia del *depot* del vehículo a los nodos. Los pesos de dicha ponderación corresponden a los valores de *alpha*, *beta* y *gamma* respectivamente.

En primer lugar se ejecutó la configuración 0.2/0.2/0.6 (*alpha*/*beta*/*gamma* respectivamente). Esta configuración es la que utiliza la heurística por *default*. Luego se ejecutó la configuración 0.4/0.4/0.2 (*alpha*/*beta*/*gamma*) sin variar ningún otro parámetro de la heurística de optimización.

Para comparar ambas configuraciones se utilizó la función objetivo que busca minimizar la heurística de optimización, a esta función objetivo se le conoce como **SimplifyScore** (SS). Se evalúo la mejora porcentual que tuvo cada instancia i con la segunda configuración, es decir,

$$variacion\,\% = \frac{SS_{conf2} - SS_{conf1}}{SS_{conf1}} \tag{2.1}$$

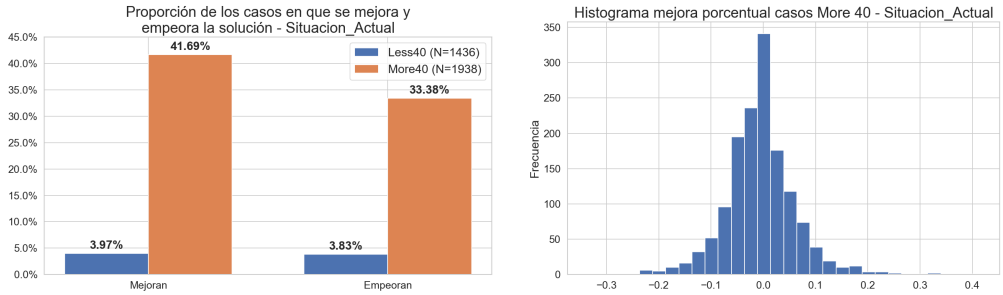
Notar que valores negativos de la ecuación 2.1 significa que la solución mejoró.

Observando el gráfico de la figura 2.6a se aprecia que tanto en las instancias con menos de 40 nodos como en las instancias con más de 40 nodos, existen situaciones en que la solución es mejorada y empeorada con la nueva solución.

También se destaca que en las instancias con más de 40 nodos, la cantidad de situaciones en que la solución varía aumenta radicalmente. En estos casos, un 41.7 % de las instancias mejora su solución, mientras que en un 33.4 % de las instancias, la solución empeora. Este aumento se puede explicar debido a que mientras más variables de decisión haya, mayor es el espacio de soluciones, por lo que cabe la posibilidad de encontrar mejorías. En cambio, en las instancias con menos de 40 puntos, el espacio de solución es menor, por lo que es más probable acercarse al óptimo en las etapas de mejoramiento de la solución inicial.

Por otro lado, al estudiar la distribución de variación porcentual de una configuración respecto a la otra (fig. 2.6b), se puede observar que existe una distribución muy amplia, es decir, se tiene que en promedio la variación porcentual es de $-0.53\,\%$, mientras que se tienen casos extremos en que la solución puede llegar a empeorar en un $40\,\%$ y al contrario mejorar en $33\,\%$. Sin embargo, si solamente se observan aquellas instancias en que la solución mejora, se tiene que estas en promedio mejoran su solución en un $4.8 \pm 5.2\,\%$.

De lo realizado anteriormente, se puede concluir que diferentes configuraciones efectivamente producen diferentes soluciones. Además, existen casos en que una determinada configuración produce mejorías para un grupo de instancias, pero a su vez desfavorece a otro grupo de instancias, por lo que **no es posible** determinar una única configuración para la heurística que sea dominante en desempeño.



(a) Proporción de casos en que se mejora y empeora la situación actual.

(b) Histograma de la mejora porcentual en las instancias con más de 40 nodos.

Figura 2.6: Resultados de la configuración de parámetros en la situación actual.

2.4. Enfoque de solución

2.4.1. Posibles soluciones

En primer lugar, hay que recordar cuales son las causas del problema (resumidas en la figura 2.4), para poder plantear una solución que se ajuste a este.

Como primera aproximación para solucionar el problema descrito, lo ideal, sería eliminar o mitigar las causas que lo generan, por lo que a continuación se evaluará cómo se pueden de eliminar o mitigar dichas causas y cuál sería su impacto en la empresa.

En primer lugar, se podría restringir a los clientes el **tamaño de las instancias** que pueden ingresar, de esta manera, la empresa podría diseñar y configurar una heurística *ad-hoc* a este tipo particular de problemas. Sin embargo, al restringir el tamaño de las instancias, se estaría yendo en contra de uno de los principios de la empresa, particularmente con el de *llegar a todas las empresas, independientemente de su tamaño*, por lo que no interesa diseñar una solución en este sentido.

En segundo lugar, la empresa no está dispuesta a aumentar el **tiempo de resolución** del problema, ya que como se ha mencionado en reiteradas ocasiones, el tiempo que tarda el optimizador en encontrar una solución debe ser acotado y breve, debido a que los clientes no están dispuestos a esperar un largo tiempo por la solución.

Como se mostrará en la sección 5.1, la resolución del VRP tiene una complejidad tipo *NP-Hard*, lo que implica que a medida que crece el problema, la dificultad de encontrar una solución crece de una forma desconocida, razón por la cual, los **recursos computacionales** siempre serán limitados. En este escenario, la mejor respuesta es definir un algoritmo en base a metaheurísticas y/o heurísticas. Estos algoritmos al ser una serie de reglas definidas siempre encontrarán una solución en un tiempo acotado, el cual depende de la programación del algoritmo.

Ahora bien, si se aumentasen los recursos computacionales de los que dispone la heurística de optimización, lo único que se lograría sería disminuir el tiempo de resolución, dejando invariante la calidad de la solución entregada. Por lo anterior, la única forma de mejorar la calidad de la solución mediante el aumento de los recursos computacionales, sería crear una nueva heurística que lo permita. En este nuevo escenario, nuevamente los recursos computacionales se volverían una limitante en términos de la calidad de la solución, por lo tanto, aumentar las capacidades computaciones no resuelve completamente el problema descrito en la figura 2.4.

Resulta lógico pensar que existen situaciones en donde el desempeño de la heurística se vea mermado debido a la cantidad de trabajo que esté realizando la *máquina* donde está alojada la heurística. Sin embargo, cabe recordar que las CPU se encuentran alojadas en un servicio de *cloud computing*, lo que le permite a la empresa *incorporar* nuevos computadores en situaciones donde la carga de trabajo sea alta y exista latencia en el proceso de resolución.

Por las razones mencionadas y tal como se mencionó en la sección 2.2.1, las causas iden-

tificadas en la figura 2.4 son parte del contexto de la industria en la que está inmersa la empresa, es más, estos elementos son aquellos que busca solucionar la empresa a sus clientes, ya que son dificultades propias del problema de ruteo.

No obstante, las heurísticas y metaheurísticas suelen tener parámetros configurables que determinan la ejecución de estas y en su defecto, podrían afectar la solución entregada. Por lo tanto, dado una tipo de problema, se puede definir un set de parámetros que le permitan a las metodologías de resolución obtener un mejor desempeño (Malitsky 2014). Una forma de realizar este ajuste, es efectuarlo de manera manual, lo que se traduce en una tarea que demanda tiempo y de personal experto para que la realice (Hutter et al. 2009).

Dado que puede ser costoso determinar la mejor configuración para la heurística, la solución propuesta en este trabajo de título es diseñar un método que permita configurar de manera automática la heurística de SimpliRoute. A este tipo de metodologías se les conoce como **Configuración Automática del Algoritmo** (AAC[5], por sus siglas en inglés).

Para ejemplificar esta idea, considere el experimento realizado en la sección anterior. Lo que haría la configuración automática de parámetros es seleccionar la configuración que mejor desempeño entrega para cada instancia, en el ejemplo, se traduce en que el configurador automático escogería la configuración *default* para ejecutar las instancias que *empeoran* con la segunda configuración. Y en el caso contrario, para los casos que *mejoran* se escogería la segunda configuración, obteniendo así los beneficios de ambas configuraciones.

2.4.2. Descripción de la solución escogida

Para describir la solución propuesta, es necesario mencionar los elementos importantes dentro del macroproceso llamado *optimización* (fig. 2.1).

El primer elemento relevante es la *instancia*, recordar que corresponde al conjunto de información que el usuario desea optimizar. El segundo elemento relevante es la *heurística de optimización*, que es donde se resuelve la instancia. Finalmente, el tercer elemento corresponde a la *solución*, la cual es el conjunto de información que contiene la planificación de las rutas.

Cada uno de estos elementos se relacionan entre sí como muestra la ecuación 2.2. En este flujo, la instancia I es enviada a la heurística A, donde A toma I como el conjunto de datos de entrada y entrega una solución.

$$I \mapsto A(I) \mapsto Solucion \tag{2.2}$$

En este escenario, la solución propuesta consiste en generar una metodología que permita configurar de manera automática la heurística de optimización. Para llevar a cabo esto, se propone crear una nueva capa de resolución al problema. El objetivo de esta nueva capa de resolución es estudiar, caracterizar y segmentar a las instancias que llegan a la heurística de optimización. Una vez segmentadas las instancias, se les asignarán los parámetros que mejor

[5]AAC: Automatic Algorithm Configuration.

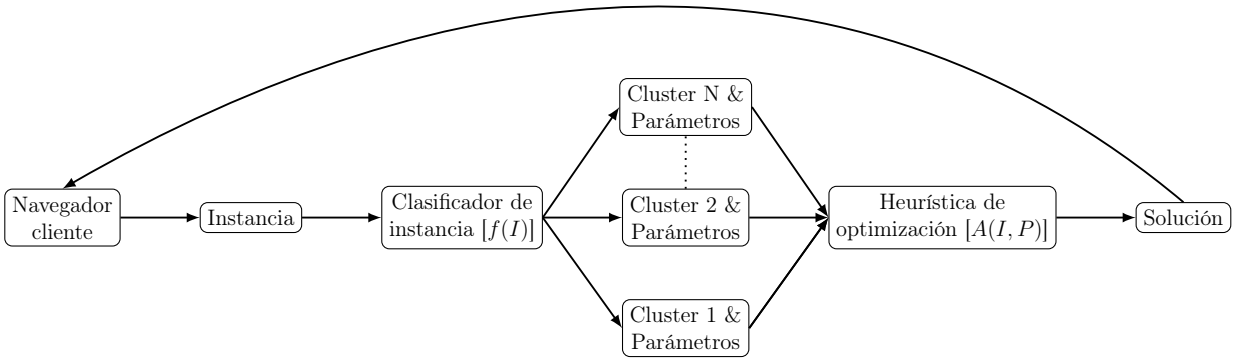


Figura 2.7: Representación flujo de resolución con modelo de clasificación.

rendimiento producen en la heurística de optimización para dicho segmento.

Esta nueva capa puede ser representada por una función clasificadora de instancias que les asocia un conjunto de parámetros. Entonces, con la incorporación de esta solución, el flujo de optimización se puede representar como sigue. Sea $f(I)$ una función que clasifica a la instancia I, entregando como resultado a la instancia I y un *set* de parámetros P a ser utilizados por la heurística, entonces se tiene que,

$$I \mapsto f(I) = \{I, P\} \mapsto A(I, P) \mapsto Solucion \tag{2.3}$$

Para construir esta nueva capa (*i.e.*, la función $f(I)$) se utilizarán herramientas de Machine Learning, ya que estas herramientas permiten aprender sobre las distintas ejecuciones de la heurística. En base a esta información, se construirá un modelo que *clasifique a las instancias y que permita decidir los parámetros de la heurística para cada instancia a resolver*.

Dado que el tiempo de ejecución es un elemento clave en el proceso de optimización, es necesario utilizar una metodología que no agregue o que agregue una fracción de tiempo marginal al proceso de ejecución. Para esto se utilizará una metodología de configuración de parámetros *offline*. Esta metodología genera su *aprendizaje* fuera del flujo de optimización y luego *utiliza lo aprendido* para determinar a qué segmento pertenece la instancia y así devolver el set de parámetros que mejor ajuste. De esta manera, la función $f(I)$ agregará un tiempo de ejecución marginal. En la sección 5.2 se profundizará esta metodología.

Finalmente, a continuación se ilustrará cómo sería el nuevo flujo en la resolución de los problemas, el cual se encuentra representado en la figura 2.7.

En primera instancia, el cliente define su problema en el **navegador**. El problema es enviado al software, el cual lo transforma y lo convierte en la **instancia**. Esta instancia ingresa al **clasificador**, modelo que determina a que **segmento** pertenece la instancia. Cada segmento tiene un set de parámetros definidos. Luego, la instancia y los parámetros son ingresados a la **heurística**. Por último, la heurística entrega una **solución**, la cual es enviada al navegador del cliente, quien continua con los otros procesos dentro de la plataforma.

Capítulo 3

Objetivos y alcances

3.1. Objetivos

Tras haber descrito, justificado el problema y planteado el enfoque de solución, se han definido los siguientes objetivos.

El **objetivo general** de este trabajo de título es el siguiente:

> **Generar un clasificador de instancias que entregue la mejor configuración de la heurística de optimización utilizada por SimpliRoute de modo de mejorar las soluciones que entrega el solver respecto a la función objetivo denominada SimplifyScore dada la restricción de 15 minutos de ejecución**

Para alcanzar el objetivo general, se definieron los siguientes **objetivos específicos**:

- Determinar los parámetros de la heurística que serán modificados
- Determinar una metodología que permita mitigar el sesgo generado por instancias similares
- Usando la metodología anterior, construir un set de instancias relevantes a partir de la base de datos de instancias de SimpliRoute
- Determinar las características más relevantes de las instancias que producen que la heurística entregue mejores resultados
- Clusterizar las instancias utilizando las características identificadas por el punto anterior
- Determinar cuáles son las mejores configuraciones para cada uno de estos clusters

Es importante destacar que para medir si la solución entregada por una determinada

configuración, es mejor que otra solución, se compararán las funciones objetivo del solver y el tiempo de ejecución. A la función objetivo se le conoce como **SimplifyScore**

3.2. Alcances

El alcance de este trabajo es encontrar una configuración de la heurística de optimización que permita que, para una determinada instancia, se pueda encontrar una mejor solución.

Es importante destacar que la solución propuesta y alcance de este trabajo solamente busca mejorar la solución en términos objetivos, es decir, busca la mejoría de la solución en términos de métricas objetivas, como por ejemplo el número de clientes que se atenderán, el número de vehículos que se utilizan, la distancia recorrida, entre otras métricas.

En base a lo anterior, para este trabajo se definió que para determinar que si una solución es mejor a otras soluciones, para cierta instancia, se tomarán en consideración dos elementos. El primer elemento corresponde a la función objetivo de la heurística de optimización, conocida como **SimplifyScore**. El segundo elemento corresponde al **tiempo de ejecución**. Ahora bien, una solución es mejor que otra cuando el **SimplifyScore** es menor. En caso que ambas soluciones tengan el mismo **SimplifyScore**, se escoge aquella solución que tiene un menor **tiempo de ejecución**. A este tipo de regla para determinar que configuración es mejor, se le conoce como *regla lexicográfica*.

También para evaluar si el nuevo flujo propuesto (eq. 2.3 y fig 2.7) mejora la situación del flujo de la heurística de optimización actual (eq. 2.2), se comparará contra la mejor versión de la heurística que se tuvo a comienzos del mes de agosto del año 2019.

Como se verá en el capitulo 6, se definirá un portafolio de heurísticas que cuenta con 24 distintas configuraciones de este.

Para la conformación de la base con la que se realizará el estudio, se utilizarán aquellas instancias generadas en el periodo Ene-2019 y Mar-2019. En la sección 2.3 se descubrió que en las instancias con mayor número de puntos, las soluciones variaban más al variar las configuraciones de la heurística, por lo que sólo se tomarán en consideración instancias que tengan al menos 40 nodos y una flota de más de un vehículo.

Capítulo 4

Metodología de Trabajo

4.1. Metodología de Trabajo

Dada la solución propuesta, se propone utilizar la metodología conocida como KDD. Esta metodología es ampliamente utilizada en la ciencia de los datos, ya que tal como su nombre lo dice, ayuda al descubrimiento de conocimiento en bases de datos (la sigla en inglés representa Knowledge Discovery in Databases). Más detalle sobre esta metodología se puede encontrar en el artículo *From Data Mining to Knowledge Discovery in Databases* de Fayyad et al. (1996).

Esta metodología se puede resumir en cinco etapas.

1. La **primera etapa** consiste en la selección de los datos relevantes para el estudio.

2. En la **segunda etapa** se debe pre-procesar los datos, de manera de obtener una base íntegra para realizar modelos.

3. La **tercera etapa** consiste en generar transformaciones de las variables existentes, con el objetivo de poder observar comportamientos en los datos.

4. En la **cuarta etapa** se aplican los modelos y herramientas propuestas con el fin de encontrar comportamientos/patrones ocultos.

5. Finalmente, en la **quinta etapa** se deben interpretar y evaluar los modelos en base a métricas relevantes del negocio u organización.

Se propone utilizar esta metodología debido a que se ajusta a los pasos necesarios para construir la nueva capa propuesta en la sección 2.4.2. Para poder crear esta nueva herramienta, primero es necesario construir conocimiento, en el sentido de que es necesario identificar varios elementos antes de construir la misma. Entre estos elementos que se deben identificar, en primer lugar se debe decidir cuáles serán los datos relevantes para los estudios y cómo estos se deben considerar. En segundo lugar se debe estudiar y decidir cuáles son las características importantes a considerar y cuáles podrían producir ruido. Luego en función de eso,

se debe decidir que modelo aplicar a dichas características y/o aspectos relevantes.

4.2. Desarrollo de la solución

La primera etapa de la metodología corresponde a la selección de los datos relevantes. En este caso, SimpliRoute cuenta con una gran base de datos asociados a cada uno de los procesos de optimización, razón por la cuál se determinará que tipos de registros se escogerán para entrenar los modelos.

En la segunda etapa de la metodología se pre-procesan los datos. En este caso, a partir de la selección de los datos realizada anteriormente, es necesario limpiar la base de datos de aquellas instancias que pueden generar ruido en el modelo o que simplemente no agregan información adicional, como lo son las instancias duplicadas.

Los primeros dos pasos son llevados a cabo en el capítulo 7. La tercera etapa consiste en realizar transformaciones relevantes a los datos existentes. En este caso en particular esta transformación tiene dos aristas. En primer lugar es necesario transformar cada instancia en un conjunto de características que la describan y la segunda arista de esta etapa es determinar si alguna variable requiere de alguna transformación relevante. Este paso el llevado a cabo en el capítulo 8.

La cuarta y quinta etapa son llevadas a cabo en los capítulos 9 y 10. En estas etapas se plantean los modelos de clusterización y predicción que luego son evaluados con las métricas estipuladas en la sección de alcances (sec. 3.2).

Capítulo 5

Marco Conceptual

5.1. Vehicle Routing Problem

El problema que se abordará en este trabajo tiene como base el problema de ruteo de vehículos. Ya que como se mencionó en la 1.2, determinar las rutas óptimas es el servicio principal que ofrece SimpliRoute. Por esta razón, es necesario describir de forma genérica el problema y las metodologías que se han desarrollado para solucionarlo.

El problema de ruteo de vehículos lo que persigue resolver es identificar la o las rutas óptimas que debe realizar una flota de vehículos para atender un determinado conjunto de destinos, sujeto a ciertas restricciones. La primera formulación de este problema es realizada por Dantzig & Ramser (1959), en este trabajo se logró plantear el VRP como una extensión del TSP. Esta formulación representa una de las variantes más sencillas del problema de ruteo de vehículos, ya que considera solamente una flota compuesta de un vehículo y un conjunto de n puntos conexos entre sí. También es importante destacar que el VRP es un problema con complejidad *NP-Hard*. Esta característica del problema hace que sea extremadamente complejo resolverlos, es decir, se requiere de mucha de capacidad computacional y/o tiempo para obtener las soluciones exactas.

5.1.1. Complejidad de resolución

Resolver el VRP puede llegar a ser muy complejo, ya que en su forma más sencilla, el número de soluciones que existen crece de forma factorial según el número de puntos a visitar, es decir, dada una flota compuesta por un solo vehículo y un conjunto con n puntos conexos entre sí, existen $\frac{1}{2}n!$ soluciones posibles (Dantzig & Ramser 1959).

En su presentación original, el VRP solamente consideraba la capacidad del vehículo como restricción al problema. Con el paso de los años, se fueron incorporando distintas restricciones al modelo que lo complejizan cada vez más.

La complejidad del problema del VRP es *NP-Hard* (Savelsbergh 2008). La sigla NP hace referencia a aquellos problemas que pueden ser resueltos mediante un algoritmo en un tiempo polinomial no-determinista. Esto quiere decir que a medida que crece el problema, la dificultad y el tiempo de resolución crece de una forma polinomial que se puede calcular. Otra forma de entender la complejidad *NP-Hard*, es que esta indica que no existe una forma polinomial conocida que permita determinar la dificultad ni el tiempo de resolución, por lo que los problemas de este tipo son muy difíciles de resolver. Para resolver este tipo de problemas se deben buscar otras alternativas para llegar a una solución.

5.1.2. VRP genérico y variantes

A continuación se presentará la formulación *clásica* del VRP.

Sea $G = (V, A)$ un gráfo dirigido, donde $V = \{0, \ldots, n\}$ es el conjunto de vértices y $A = \{(\mathrm{i}, j) : \mathrm{i}, j \in V, \mathrm{i} \neq j\}$ es el conjunto de arcos. El vértice 0 representa al *depot*, donde están ubicados los M vehículos con capacidad Q, mientras que el resto de los puntos representan los nodos a visitar, *i.e.*, los clientes. Los clientes tienen una demanda no-negativa menor a la capacidad de los vehículos, *i.e.*, $0 < \mathrm{d_i} \leq Q$. Luego, a partir de A, se define una matriz de costo $c_{\mathrm{i}j}$. En la formulación clásica, se considera el tiempo de viaje, las distancias y los costos como equivalentes. Finalmente, el VRP consiste en designarle rutas a los M vehículos de modo que las rutas comiencen y terminen en el *depot*, que cada cliente sea visitado solo y solo una vez y que la demanda de la ruta sea menor a la capacidad Q del vehículo. La designación de rutas óptimas está dada por la minimización de los costos de realizarlas.

A partir de la formulación anterior, con el paso del tiempo hay ido surgiendo distintas variantes que se hacen cargo de realidades no modeladas por la formulación clásica. Entre estas variaciones se tienen la inclusión de las ventanas horarias, relaciones de precedencia entre los puntos a visitar, prioridades de visitas, entre otras variantes que el modelo original no capturaba.

En el trabajo realizado por Drexl (2012) se identifican las variaciones del VRP más comunes y más importantes realizadas. Se nombrarán las que tienen relación con el servicio ofrecido por SimpliRoute.

- **VRP con Capacidades (CVRP):** Esta variante ha sido bastante estudiada, y lo que considera es que cada punto a visitar tiene una demanda no-negativa y los M vehículos que conforman la flota pueden tener capacidades distintas mayores a 0.

- **VRP con Ventanas Horarias (TWVRP):** Esta variante lo que considera es que cada cliente puede o no tener una ventana horaria para ser atendido.

- ***Pick-up-and-deliery* (PDP):** En esta variante la tarea que deben realizar los vehículos es transportar carga desde un punto a otro y no solo desde el *depot*, sino que también deben recoger y entregar carga en su ruta.

- **VRP con flota heterogénea (HRVP):** Esta variante considera que los vehículos pueden ser distintos entre sí, en términos de costos fijos y variables, de capacidad y de tipo de vehículo.

- **VRP con Rutas Abiertas (LRP):** En esta variante se permite que los vehículos no necesariamente comiencen y terminen en el *depot*.

- **Vehículos con zonas y habilidades:** Esta variante considera que los vehículos están confinados a trabajar en ciertas zonas definidas. También considera que algunos clientes requieren de ciertas habilidades para poder ser atendidos, por lo que para que un vehículo pueda visitar a dicho cliente, tiene que tener esa habilidad (por ejemplo transporte de productos congelados)

Todas las variantes van agregando o modificando restricciones a la formulación clásica, de este modo se puede ir ajustando el modelo a la realidad.

5.1.3. Metodologías de resolución

Dado que el VRP es un problema del tipo *NP-Hard*, llegar a encontrar el óptimo del problema puede llegar a tardar días o meses. Para sortear esta problemática se han construido diversos tipos de soluciones. Estas metodologías se pueden clasificar en tres tipos, los *algoritmos exactos*, las *heurísticas* y las *metaheurísticas*. Para más detalle de las metodologías de resolución, remitirse a Laporte (2009) y Drexl (2012).

Algoritmos exactos

Lo que buscan los algoritmos exactos es encontrar el óptimo del problema de forma exacta. No obstante, dado el nivel de complejidad que tiene el problema, existen pocos algoritmos que puedan resolver para instancias que ocurran en la vida cotidiana. Desde lo académico se ha logrado encontrar soluciones para instancias con aproximadamente 100 puntos (Laporte 2009)

Entre los algoritmos exactos que más destacan se encuentran los pertenecientes a la familia de los *direct tree search* y los de *programación dinámica*.

Los algoritmos de **Direct Tree Search**, en español árboles de búsqueda directa, resuelven el VRP utilizando como base el algoritmo de *branch-and-bound*. Este algoritmo va creando ramas de soluciones posibles y va detectando en cuales de estas no se va a encontrar una solución óptima, por lo que se descarta dicha rama. Las extensiones de este algoritmo que han resultado más exitosas son las propuestas por Christofides & Eilon (1969), Christofides (1976) y Christofides et al. (1981).

Los algoritmos de **Programación dinámica** consisten en algoritmos que crean subproblemas y los van resolviendo. A medida que va encontrando el óptimo para estos subproblemas, los va haciendo parte de la solución final.

Heurísticas Clásicas

Debido a que el tiempo y costo de encontrar una solución exacta para problemas grandes es alto, se han tenido que desarrollar alternativas. Las heurísticas son una respuesta a esta problemática. Estas se pueden describir como una serie de *reglas de oro* que guían la búsqueda de la solución en el espacio de soluciones Burke et al. (2019). Esta búsqueda probablemente lleve a encontrar una solución sub-óptima, pero de buena calidad. Estas *reglas de oro* son construidas en base a la experiencia de los investigadores y del conjunto de instancias que se esté estudiando.

Las heurísticas pueden ser divididas en dos tipos, las heurísticas de *construcción* y las heurísticas de *mejora*. La principal diferencia está en que las del primer tipo no comienzan con una solución factible, sino que van construyendo la solución desde cero. Mientras que las heurísticas de *mejora* comienzan con una solución factible y la van mejorando.

Dentro de las **heurísticas de construcción**, uno de los algoritmos más reconocidos es el *algoritmo de ahorro* propuesto por Clarke & Wright (1964). Este algoritmo consiste en ir incorporando los puntos a la ruta, en la media que esta incorporación se traduzca en una disminución de la función objetivo. La solución comienza con una ruta básica, que une el *depot* con un nodo y devuelta al *depot*. Luego se comienza a iterar sobre los nodos.

Las **heurísticas de mejora** comienzan con una solución factible. A partir de este punto se trata de generar mejoras en la función objetivo haciendo cambios en las rutas. Estos cambios pueden ser (i) *intra-rutas*, es decir, cambiar el orden de visita dentro de una misma ruta o pueden ser (ii) *inter-ruta*, es decir, se intercambian los puntos a visitar entre distintas rutas.

Metaheurísticas

Con respecto a las metaheurísticas estas consisten en relajaciones a la búsqueda exacta de la solución, mediante mayores exploraciones al espacio de soluciones, para esto se definen reglas de búsqueda y se realizan múltiples recombinaciones de las soluciones. Estas metodologías suelen encontrar buenos resultados, pero aún siguen requiriendo de una gran capacidad computacional (Bonet 2018).

Las metaheurísticas se pueden dividir en tres campos de estudio, (i) los Local Search, (ii) Population Search y (iii) redes neuronales.

Los algoritmos de **local search** principalmente se encargan de explorar el espacio de solución en el vecindario de la solución que se lleve calculada en ese momento. Un tipo de Local Search reconocido es el Tabu Search presentado por Glover (1986).

Los algoritmos de **population search** buscan replicar el proceso de selección natural. Para esto se utilizan algoritmos genéticos (Holland 1975). Estos algoritmos lo que hacen es seleccionar a un *padre* del espacio de soluciones, perturbarlo y ver si dicha perturbación entrega una mejor solución, comparada con la peor solución de la población, en dicho caso se reemplaza la peor solución por la solución encontrada.

Las **redes neuronales** buscan replicar el funcionamiento de las neuronas en el cerebro humano, con el objetivo de imitar la inteligencia. Al utilizar estas metodologías lo que se logra es que mediante un proceso de aprendizaje se cree un modelo que permita recorrer el espacio de soluciones de la mejor manera. Entre las soluciones destacadas, aplicadas al VRP se encuentran las realizadas por Ghaziri (1991) y Schumann & Retzko (1995).

5.2. Configuración del algoritmo

Existe una serie de algoritmos que son utilizados para encontrar soluciones a problemas altamente complejos, entre los cuales se encuentran los algoritmos que resuelven problemas de optimización combinatorial, como lo es el VRP. El comportamiento de estos algoritmos está determinado por una serie de reglas y criterios definidos, los cuales pueden o no estar parametrizados.

Ahora bien, es lógico pensar que dado que el comportamiento del algoritmo varía al cambiar la configuración, es posible encontrar diferentes soluciones. Este hecho motiva la necesidad de encontrar una configuración que permita que un algoritmo de optimización tenga el mejor desempeño, es decir, encuentre las mejores soluciones.

En consecuencia, el proceso para encontrar la mejor configuración del algoritmo, puede llegar a ser bastante compleja, tedioso y requerir de una enorme cantidad de esfuerzo, como lo muestra Adenso-Diaz & Laguna (2006). Debido a la complejidad y a que es una tarea que demanda tiempo, motiva la necesidad de automatizar dicho proceso.

A continuación se describirán cuatro aproximaciones que son usualmente utilizadas para resolver el problema de la automatización de la configuración de los parámetros. Estas distinciones son identificadas por Malitsky (2014), las cuales corresponden a las siguientes: construcción del algoritmo, configuración independiente de la instancia, configuración dependiente de la instancia y métodos adaptativos. Para mayor detalle de esta revisión, remitirse al capítulo 2 del libro *Instance-Specific Algorithm Configuration* de Malitsky (2014).

Algorithm Construction [6]

Esta metodología consiste en algoritmos que van **construyendo la heurística** a partir de bloques fundamentales más pequeños. En estos casos, se utilizan técnicas de machine learning que permiten evaluar distintas configuraciones para luego armar una configuración que tenga un buen desempeño en un conjunto de instancias de entrenamiento. Dos ejemplos de esta metodología son el sistema MULTI-TAC (Minton 1996) y el sistema CLASS (Fukunaga 2008).

La primera metodología lo que busca es construir una secuencia de tareas a realizar, las cuales se van agregando paso a paso. Para ir seleccionando cada una de las reglas que se aplicarán, en cada iteración se escoge la configuración que más éxito tuvo en la base de prueba, es decir, la configuración donde más instancias son ejecutadas correctamente. Esta

[6]Se utilizará el nombre en inglés debido a que resulta complejo traducir este concepto al español sin que pierda su sentido.

metodología tienen la ventaja de que puede representar a todos los posibles solvers existentes y puede identificar las variaciones que hacen que estos solvers tengan un mejor desempeño (Malitsky 2014). Una desventaja de esta metodología es que a medida que aumenta el número de reglas posibles a utilizar, el número de configuraciones posibles también aumenta de forma exponencial, por lo que se hace muy difícil calcular la configuración óptima. Lo que la metodología CLASS persigue obtener la automatización de configuración de los algoritmos, orientada al uso de los Local Search. Lo que busca es crear un lenguaje común entre todos los solvers de Local Search.

Instance-Oblivious Tuning [7]

Esta metodología lo que busca es encontrar una configuración de parámetros que tenga el **mejor desempeño promedio** en el set de instancias de entrenamiento. Para los solvers que se desarrollan en esta clasificación existen parámetros categóricos, es decir, parámetros que determinan una decisión sobre que camino tomar, como por ejemplo con que heurística comenzar. También hay parámetros ordinales, quienes usualmente controlan criterios de decisión, como puede ser el criterio de parada de un proceso de clusterización. Y finalmente hay parámetros continuos, que pueden definir la probabilidad de tomar una decisión o los pesos relativos de una función objetivo. Es importante distinguir la naturaleza de los parámetros a configurar, debido a que no resulta trivial comparar directamente en ciertos casos, por ejemplo, los valores categóricos no tienen distancia ni orden, por lo que es imposible compararlos entre sí sin haber previamente definido un criterio.

Existen múltiples casos en que se utiliza esta metodología para configurar los algoritmos, entre los identificados se encuentran CALIBRA (Adenso-Diaz & Laguna 2006), MADS (Audet & Orban 2006) y SATenstein (Khudabukhsh et al. 2009). También está la metodología de F-Race (Birattari et al. 2002), la cual consiste en que distintas configuraciones compiten entre sí en un set de entrenamiento y aquellas configuraciones que se vayan quedando atrás en ciertos indicadores estadísticas se van eliminando, hasta tener a un ganador.

Por último, se destaca la metodología ParamsILS (Hutter et al. 2009) debido a que logra ser una metodología genérica para la configuración de los parámetros y capaz de configurar para una gran cantidad de parámetros. Esta metodología lo que propone es comenzar desde una configuración aleatoria, para luego desde ahí ir estudiando el espacio de solución mediante Local Search en un determinado vecindario hasta encontrar un óptimo local. Una vez alcanzado el óptimo local, se escoge un nuevo punto del espacio de solución y se realiza el mismo procedimiento. Como competencia de esta metodología, aparecen algoritmos genéticos basados en género, esto permite ir explorando el espacio de solución de una forma en que en una iteración, se prueban las diferentes configuraciones, luego a la siguiente generación se le provoca una *mutación* mediante Local Search y se evalúa esta generación, de modo que si la mutación es exitosa, esta tendería a heredarse a las siguientes generaciones.

[7] Se utilizará el nombre en inglés debido a que resulta complejo traducir este concepto al español sin que pierda su sentido.

Instance-Specific Regression [8]

En este caso, lo que se busca es encontrar la **mejor configuración** para una **instancia determinada**, a diferencia lo descrito anteriormente donde lo que se busca es encontrar una configuración suficientemente buena para el promedio.

Como se mostró en la sección 2.3, distintas configuraciones conducen a distintas soluciones, debido a que todas las instancias son distintas entre sí, es decir, resuelven un problema con características distintas, tal como lo demuestra Wolpert & Macready (1997), en donde se plantea que es infactible crear un único algoritmo que tenga un óptimo desempeño con todas las instancias.

Entre las metodologías más utilizadas para encontrar configuraciones que se ajusten de forma específica a las instancias, están los portafolios de algoritmos. Estas metodología lo que persiguen es estimar un modelo que permita predecir cuál de todos los algoritmos pertenecientes al portafolio tendrá el mejor desempeño. Dentro de esta metodología se identifica a SATzilla Xu et al. (2008) como uno de los más galardonados para resolver el problema de SAT.

Otra metodología destacada corresponde a la denominada CPHydra O'Mahony et al. (2008), la cual lo que busca es maximizar la probabilidad condicionada a la instancia que tiene un determinado algoritmo de encontrar una solución en un determinado tiempo, es decir, dada a instancia se busca maximizar la probabilidad de encontrar una solución. Para estimar la probabilidad de resolución que tiene el algoritmo, se recurre a un set instancias de entrenamiento, de las cuales se tiene conocimiento del rendimiento de cada uno de los solvers para cada una de las instancias, entonces cuando llega una nueva instancia, se determinan los k vecinos más cercanos a dicha instancia y se estima la probabilidad de resolución que tienen los algoritmos.

Adaptative Methods [9]

Estas metodologías lo que hacen es ir **ajustando el algoritmo** a medida que se va resolviendo el problema, de esta manera, el algoritmo va aprendiendo de la estructura del problema y utiliza esta información para decidiendo que camino es mejor tomar dado el caso.

Un ejemplo de esta metodología es la denominada STAGE (Boyan et al. 2000), esta metodología mientras va buscando la solución, STAGE va aprendiendo a predecir el desempeño que tendrán los Local Search, para así poder decidir que Local Search utilizar en cada iteración. Esta metodología ha demostrado ser útil al momento de mejorar el desempeño de los algoritmos de Local Search.

Para este tipo de metodologías, se destaca también la denominada *impact-based search strategies* Refalo (2004), esta metodología lo que busca es reducir el espacio de búsqueda de las configuraciones, para lo cual va recordando cómo se ha ido acotando el dominio de las

[8] Se utilizará el nombre en inglés debido a que resulta complejo traducir este concepto al español sin que pierda su sentido.

[9] Se utilizará el nombre en inglés debido a que resulta complejo traducir este concepto al español sin que pierda su sentido.

variables y el impacto que tuvo, para así ir conduciendo a la selección de la mejor heurística.

Por último, para cerrar esta sección, se mostrará una distinción relativa al momento en que los distintos modelos produce el aprendizaje para la automatización.

Momento del aprendizaje

En Burke et al. (2019) se identifica una distinción en relación al momento en que el proceso de automatización aprende para poder determinar la mejor configuración de los parámetros. En esta distinción se identifican tres grupos: el primer grupo corresponde a aquellos en que el aprendizaje se realiza *online*, el segundo grupo corresponde a aquellos cuando el proceso de aprendizaje se realiza de forma *offline* y tercer grupo cuando no se realiza ningún tipo de aprendizaje.

Con respecto a la configuración de parámetros *online*, en estos casos lo que se busca es que el algoritmo vaya aprendiendo a medida que va solucionando el problema (Burke et al. 2019). El beneficio de utilizar esta aproximación está en que el algoritmo se centra en buscar configuraciones que suelen tener un buen desempeño, de esta manera, no se perdería tiempo en evaluar malas configuraciones (Demkes 2014). Dado que el proceso de aprendizaje se realiza durante la obtención de la solución, se considera que este procedimiento añade tiempo a la ejecución del algoritmo.

Por otro lado, con respecto a la configuración de parámetros *offline*, esta consiste en que el algoritmo aprende en base a un set de instancias de prueba, de manera de predecir la mejor configuración para instancias aún no vistas por el algoritmo (Burke et al. 2019). El beneficio de utilizar esta aproximación, está en que dado que el tiempo no es tan relevante al momento de evaluar configuraciones, el algoritmo se puede dar el lujo de evaluar configuraciones que aún no han sido probadas (Demkes 2014). A diferencia de la configuración *online*, este proceso de aprendizaje no añade tiempo de ejecución al algoritmo, debido a que se aprende en un instante distinto al instante en que se intenta resolver el problema.

Finalmente, en el caso en que no hay aprendizaje, en ningún momento el algoritmo obtiene una retroalimentación de las configuraciones realizadas anteriormente, por lo que la selección de la configuración se realiza básicamente en base a ciertas reglas (Demkes 2014).

Para el caso de este trabajo, el enfoque de aprendizaje que se utilizará será el aprendizaje *offline* debido a lo mencionado en la sección 2.2.1, donde se considera el tiempo de resolución como una característica relevante y no se desea que la resolución del problema tarde más tiempo de lo que se demora actualmente.

Conclusión

Tras la breve revisión realiza en la sección anterior, se determinó que el mejor curso de acción que se tomará para llevar a cabo la solución propuesta es abordarlo con la metodología de *Instace-Specific Regression*. La razón es que actualmente la empresa no cuenta con variadas formas de heurísticas que se pueden utilizar, por lo que no resulta factible implementar un *método constructivo*. Sin embargo, la heurística que han ido desarrollando si posee algunos parámetros que son modificables y que hoy día solo se usa ha configuración *default*.

Hasta la fecha, cada vez que se realizan modificaciones o actualizaciones al algoritmo la empresa se preocupa que estas modificaciones sigan teniendo un óptimo desempeño en promedio, por lo que se podría decir que se está utilizando una metodología de *Instance-Oblivious Tuning*. Sin embargo, como se ha discutido, esta metodología es incapaz de capturar la variabilidad de las instancias y de hacer que para todas las intactas el algoritmo tenga un buen desempeño.

Por último, los métodos adaptativos no interesa usarlos como metodología de tuneo de parámetros, ya que esta metodología al funcionar de forma *online* agrega tiempo de ejecución al momento de resolver para una instancia. Por la misma razón, para las otras metodologías se escogerá un enfoque de configuración *offline*.

5.3. Modelos de aprendizaje

Como ya se ha mencionado, para llevar a cabo la solución propuesta (sec. 2.4.2) se hará uso de técnicas de *Machine Learning* para aprender sobre las características de las instancias y así poder agruparlas.

Una clasificación clásica de los modelos de *Machine Learning* corresponde al nivel de supervisión humana que tienen los modelos en su proceso de entrenamiento. Bajo esta clasificación se identifican cuatro categorías (Géron 2017). El aprendizaje *supervisado*, *no-supervisado*, *semisupervisado* y *mediante refuerzo*.

El aprendizaje **supervisado** consiste en modelos que además de las características de los registros del problema que se quiere modelar, se incluye el *output* deseado, es decir, incluye una etiqueta (Géron 2017). Esta etiqueta es definida por un humano. Dentro de los modelos supervisados, estos se diferencian según la tarea que llevan a cabo, están aquellos modelos que realizan una *clasificación* y otros que realizan una *predicción*. Un ejemplo del primer caso son los modelos de regresión logística y un ejemplo del segundo caso son los modelos de regresión lineal.

El aprendizaje **no-supervisado** consiste en modelos que no requieren conocer la 'etiqueta' del registro, ya que estos modelos buscan identificar cuál es la relación que existe entre los registros y agruparlos, es decir, generar sus propias etiquetas (Géron 2017).

El aprendizaje **semisupervisado** consiste en modelos que realizan una agrupación de forma independiente, tal como lo hace modelos no-supervisados, pero luego requiere que algún humano valide dicha agrupación o le coloque una cierta etiqueta, para que en etapas sucesivas el modelo tenga incorporada dicha información (Géron 2017).

Por último, el aprendizaje **mediante refuerzo** son modelos que son capaces de observar su *ambiente* y tomar decisiones y esas decisiones pueden ser premiadas o castigadas. De esta manera, lo que se busca es que el modelo aprenda a tomar decisiones que lo favorezcan (Géron 2017). Ejemplos de este tipo de modelos son aquellos que aprenden a jugar un determinado juego.

A continuación se revisarán los aspectos teóricos de tres métodos de agrupación de datos. Dos de ellos son corresponden a métodos no supervisados y uno corresponde a un método supervisado. Destacar que de estos métodos se revisarán los aspectos más relevantes y su funcionamiento de forma general, sin llegar a detallar en profundidad a los mismos. Los métodos propuestos son *K-means*, *Gaussian Mixture* y *k-NN*.

5.3.1. *K-Means*

Este modelo de agrupación de datos permite agrupar la información en un número k de grupos. Este método de agrupación corresponde a un método que realiza su aprendizaje de forma *no-supervisada*, ya que no requiere conocer las 'etiquetas' de estos.

Para realizar la clasificación, lo que hace este método es dividir la información en los k grupos y asignar cada punto de información a alguno de estos grupos. A estos grupos también se les conoce como *clusters*. La asignación de la información a los *clusters* la realiza mediante la minimización del criterio conocido como *within-cluster sum-of-squares* (WCSS por sus siglas en inglés), expuesta en la ecuación 5.1 (Pedregosa et al. 2011).

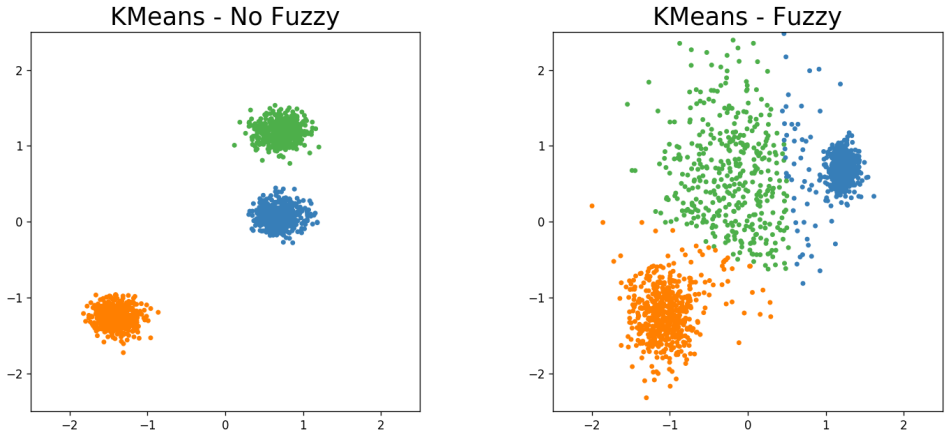
El planteamiento del problema propuesto por este modelo (*k-means*) es el siguiente. Sea un conjunto de datos N, donde la información de cada dato se encuentra representado por el vector de características X. Lo que hace *k-means* es dividir la data en k clusters disjuntos entre sí. Cada cluster C_j queda descrito por μ_j, el cual es la media de las características X del conjunto de datos en el cluster j. A esta métrica se le conoce como *centroide*. Luego, el objetivo que persigue esta metodología es escoger los *centroides* de manera tal que se minimice el WCSS, tal como lo muestra la ecuación 5.1 (Pedregosa et al. 2011).

$$WCSS = \sum_{\mathrm{i}=0}^{n} \min_{\mu_j \in C} (||x_{\mathrm{i}} - \mu_j||^2) \tag{5.1}$$

El problema anterior es resuelto mediante el algoritmo de Lloyd, el cual consiste en tres simples pasos.

En la **Paso 1** se escogen de forma aleatoria k centros. En el **Paso 2** se asigna cada dato de la base al *centroide* más cercano. Finalmente, en el **Paso 3** se recalculan los nuevos centros del cluster, los cuales se encuentran definidos por μ_j. Este algoritmo repite los pasos 2 y 3 hasta que la diferencia entre los viejos y nuevos *centroides* ya no sea significativa, en ese momento se detiene y entrega el resultado (Pedregosa et al. 2011).

Este método suele funcionar bien cuando *clusters* se encuentran bien definidos en el espacio, es decir, se encuentran claramente separados entre sí. Cuando los puntos en el espacio es difusa, es decir, se encuentran todos cercanos entre sí, este método no resulta muy bueno para definir los clusters. Un ejemplo de lo anterior son las figuras 5.1a y 5.1b. En la figura 5.1a, se aprecia que la data está claramente separada en el espacio, mientras que en la figura 5.1b está mas difusa, por lo que es difícil definir los *clusters*.



(a) Ejemplo clusterización cuando la data está separada entre sí.

(b) Ejemplo clusterización cuando la data no está bien separada entre sí.

Figura 5.1: Ejemplos de la clusterización realizada por el método *k-means* en casos en que la data no es difusa y que si es difusa. Ejemplos extraídos de (Pedregosa et al. 2011).

5.3.2. *Gaussian Mixture*

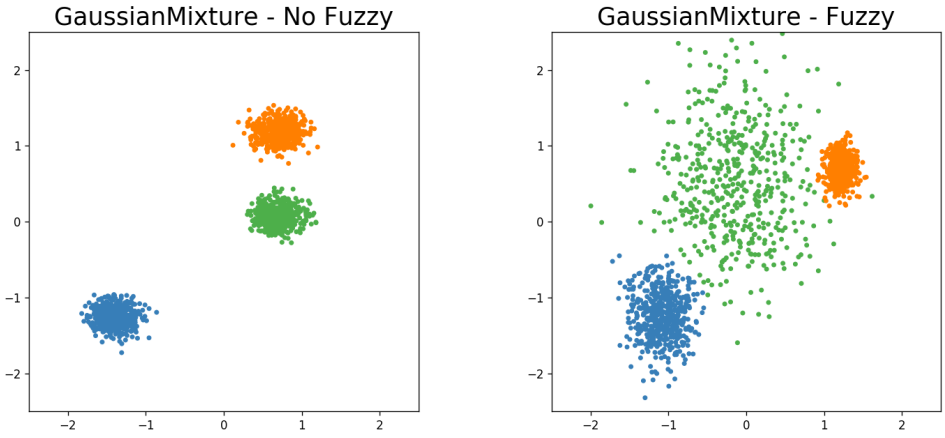
Al igual que el método *k-means*, este método también permite agrupar la data en *clusters*. Este método también realiza su proceso de aprendizaje de forma *no-supervisada*, ya que realiza la agrupación exclusivamente a partir de las características, sin conocer las 'etiquetas' de los registros.

Una de las ventajas que tiene este método de *clusterización* por sobre el método de *k-means*, es que permite tener en consideración la varianza de los datos, y no solo la distancia.

Este método de clusterización es un modelo probabilístico, ya que se basa en que los datos de la muestra son producidos a partir de una cierta distribución Gaussiana. Sin embargo, este modelo plantea que los datos no provienen de una única distribución, sino que existen clusters de datos donde cada cluster tiene su propia distribución, por lo que la generación de los datos en la base proviene de una superposición de estas distribuciones. A partir de esta idea, el objetivo de esta metodología es determinar la distribución de cada uno de los clusters de datos (Pedregosa et al. 2011).

Para determinar las distribuciones de los clusters, este método busca maximizar la esperanza de que los datos provengan de una cierta distribución θ, a esto se le conoce como *expectation-maximization*. En este proceso lo que se busca es determinar los parámetros de las distribuciones de cada uno de los clusters. Para eso se maximiza la *log-verosimilitud* de que los datos provengan de cierta distribución.

En este caso, la *log-verosimilitud* se describe como la probabilidad de obtener la data X para los Z grupos dados los parámetros θ. Entonces básicamente al maximizar lo que se hace es obtener los parámetros θ que estiman de mejor forma los datos X.



(a) Ejemplo clusterización cuando la data está separada entre sí.

(b) Ejemplo clusterización cuando la data no está bien separada entre sí.

Figura 5.2: Ejemplos de la clusterización realizada por el método *Gaussian Mixture* en casos en que la data no es difusa y que si es difusa. Ejemplos extraídos de (Pedregosa et al. 2011).

Como ya se mencionó, este método de clusterización permite definir de mejor manera los clusters, incluso cuando los datos no se encuentran tan alejados entre sí. Esto se puede apreciar en las figuras 5.2b y 5.2a, donde cada color representa a un cluster.

5.3.3. *k-Nearest Neighbors*

A diferencia de los modelos anteriores, este método realiza su aprendizaje de forma *supervisada*, ya que requiere que se le indique cuales son las 'etiquetas' de los registros. Para más detalle remitirse a Pedregosa et al. (2011).

De este modelo existen dos versiones, una que permite clasificar la información, al igual que los modelos anteriores, y una segunda versión que permite directamente realizar predicciones. En este caso se hará uso de la versión que permite identificar clasificaciones.

Este es un modelo con un funcionamiento bastante sencillo, se basa en la idea de clasificar a partir de la base de entrenamiento sin realizar interpolaciones ni extrapolaciones de la data, sino que dice que supone que el nuevo dato de entrada se parecerá a sus vecinos cercanos.

Este método lo que hace es tomar el dato de entrada X y compararlo con los k vecinos más cercanos en la base de entrenamiento y a partir de esa vecindad realizar una predicción.

El ejemplo más sencillo ocurre cuando k es igual a 1. En ese caso lo que realiza el modelo es tomar el dato de entrada X, compararlo con el primer vecino más cercano y predecir que ambos van a tener el mismo comportamiento, por ende la misma clasificación. Para valores de $k > 1$, el modelo determina que la nueva instancia tendrá el mismo comportamiento que la mayoría, es decir, cada uno de los vecinos cercanos *vota* sobre que categoría debiese tener el nuevo dato de entrada y la que más se repita es la escogida.

Capítulo 6

Selección parámetros a configurar

Se le recuerda al lector que la solución propuesta al comienzo de este trabajo (sec. 2.4.2, cap. 2) consiste en determinar cuál es la mejor configuración de la heurística de optimización de SimpliRoute para una determinada instancia. Para llevar a cabo este objetivo se debe comenzar escogiendo cuáles serán los parámetros que se configurarán. Es por esto, que el objetivo de este capítulo es mostrar cuáles serán los parámetros de la heurística de optimización cuya elección será automatizada.

Para llevar a cabo dicho objetivo, en primera instancia se explicará de forma general el funcionamiento de algunas partes de la heurística de optimización de SimpliRoute, en particular las asociadas a los parámetros escogidos. Luego, se validará el impacto que tiene la variación de dichos parámetros en las soluciones entregadas por la heurística de optimización.

6.1. Funcionamiento general de la heurística de optimización

En términos generales, la heurística de optimización utilizada por SimpliRoute se puede dividir en dos fases. La primera fase denominada *creación de subproblemas*, donde el problema general, llamado instancia, es dividido en pequeños subproblemas. Luego, cada uno de estos subproblemas es enviado a la siguiente fase, denominada *optimización*. En esta fase cada uno de los subproblemas es resuelto mediante las diferentes heurísticas implementadas. Finalmente, cuando a cada uno de estos subproblemas se le encuentra solución, estas soluciones son unidas para entregar la solución general.

Dado que los parámetros que serán descritos en este capítulo y que estarán disponibles para su modificación sólo tienen influencia en las decisiones tomadas en la primera fase, no se profundizará en el funcionamiento ni en los detalles de la fase de *optimización*.

Tal como el nombre de la primera fase lo indica, el objetivo de esta fase es crear subproblemas a partir del problema original. Como es lógico, cada subproblema se construye a

partir de los nodos y los vehículos del problema original. En otras palabras, cada subproblema posee un subconjunto $n_{\text{i}} \subseteq N$ de nodos y un subconjunto $v_{\text{i}} \subseteq V$ de vehículos, donde N y V corresponden al conjunto de nodos y vehículos del problema general, *i.e.* la instancia. Cabe destacar que los subproblemas creados corresponden a subconjuntos disjuntos de la instancia, es decir, la unión de todos los subproblemas permite recrear a la instancia.

La idea general para seleccionar cuáles son los nodos y vehículos que conformarán a cada subproblema se basa en lo siguiente. En primer lugar los nodos del conjunto de nodos N son agrupados en n grupos. Luego, se determina que habrán tantos subproblemas como grupos de nodos se hayan creado y a cada uno de esos subproblemas se les asignará un grupo (conjunto) de nodos. Para crear estos grupos, se disponen de diferentes metodologías que separan a los nodos. Además, cada una de estas metodologías que dividen a los nodos toman en consideración dos criterios. El primer criterio indica que si para conformar los grupos sólo se mirará la posición de estos o si también se considerará la demanda de los nodos. El segundo criterio indica si los grupos que se crearán serán o no proporcionales a las capacidades de los vehículos.

A esta altura de la primera fase, se han creado n subproblemas (uno por cada grupo) y para estos subproblemas solamente se han definido cuáles serán sus nodos. Ahora solo resta asignar a los vehículos a los subproblemas. Para realizar dicha asignación se toman en consideración tres criterios, el primer criterio corresponde a la estimación de clientes que no podrán ser atendidos si el vehículo v_{i} es asignado al conjunto de nodos n_j. El segundo criterio corresponde al costo en el que se incurre por usar el vehículo v_{i}. Y el tercer criterio corresponde a la distancia que existe entre el *depot* del vehículo v_{i} y el centro del conjunto de nodos n_{i}.

Una vez creados los subproblemas, cada uno de estos es resuelto en la siguiente fase.

Resulta importante destacar que cuando la heurística de optimización alcanza los 15 minutos de ejecución, esta deja de optimizar y no entrega ninguna solución. Razón por la cual es muy importante mantener el tiempo de ejecución bajo. De todas formas, este problema actualmente está siendo abordado por la empresa.

En la sección siguiente se explicará con un poco más de detalle los parámetros y los posibles modificaciones que se pueden realizar.

6.2. Descripción de los parámetros a configurar

6.2.1. Parámetros a configurar

En base a lo anterior, los parámetros pueden ser categorizados en cuatro niveles, según el orden de ejecución en que son requeridos. Al primer nivel se le denominará **Generación de grupos de clientes (nivel 1)**, al segundo nivel se le denominará **Criterio de agrupación (nivel 2)**, al tercer y cuarto nivel se les denominará **Criterio de balanceo (nivel 3)** e **Importancia relativa de atributos (nivel 4)**.

Ahora bien, como se mencionó en la sección anterior, el primer paso es dividir el conjunto de nodos en n grupos. Para esto, en el **nivel 1** se dispone con tres metodologías distintas, denominadas *ByArray*, *ByNewMethod* y *ByPaper*. Básicamente lo que hacen estas metodologías es agrupar a los clientes en los n grupos siguiendo diferentes reglas de agrupación.

Luego, para cada una de estas metodologías se deben definir los criterios en base a los que conforman los grupos. Para esto se hace uso de los parámetros del **nivel 2**. En este nivel existen dos criterios posibles, el primer criterio es conformar los grupos mirando exclusivamente la distribución en el espacio de los nodos, mientras que el segundo criterio consiste si se mirará la distribución de la demanda, *i.e.*, si se construirán grupos con demandas similares.

Un segundo criterio que se puede tomar en consideración al construir las agrupaciones de clientes, es considerar las capacidades de los vehículos, esto significa que los grupos de nodos, será proporcionales a las capacidades de los vehículos. Este criterio es recogido por el **nivel 3**.

Una vez que ya se han construido los grupos de clientes, es necesario asignarles una flota de vehículos a cada uno de estos grupos, para que luego el problema sea resuelto. Esta asignación está reglada por las parámetros del **nivel 4**. Para asignar los vehículos a los grupos de clientes se toman en consideración tres atributos, los cuales son: (i) *estimación de clientes que no podrán ser atendidos*, (ii) *costo por uso de los vehículos* y (iii) *distancia del depot del vehículo al centro del grupo de nodos*. Estos tres criterios son recogidos en una función objetivo, donde cada uno de estos atributos son ponderados por un valor entre 0 y 1, por lo que el coeficiente que acompaña a estos valores en la función representa la importancia relativa de dicho atributo, como se puede apreciar en la ecuación 6.1, con A_1, A_2, A_3 equivalente a los atributos descritos respectivamente.

$$F = \alpha \cdot A_1 + \beta \cdot A_2 + \gamma \cdot A_3 \quad con\ \alpha + \beta + \gamma = 1 \tag{6.1}$$

Con la finalidad de acotar las combinaciones posibles, se propone escoger sólo dos combinaciones, las cuales según el equipo de SimpliRoute han mostrado ser buenas. Estas configuraciones son las siguientes:

$$\boxed{\textit{conf. 1:}} \quad \alpha = 0.2 \wedge \beta = 0.2 \wedge \gamma = 0.6$$
$$\boxed{\textit{conf. 2:}} \quad \alpha = 0.4 \wedge \beta = 0.4 \wedge \gamma = 0.2$$

Ya descritos los distintos niveles que serán configurados, es sencillo verificar que existen 24 posibles configuraciones, ya que por cada metodología existente (3) en el **nivel 1**, se puede ejecutar de ocho formas distintas. Las diferentes combinaciones se encuentran resumidas en la tabla A.1 adjunta en el Anexo A.

6.2.2. Configuración actual de la heurística

Ya descritos los parámetros que son configurables de la heurística de optimización, es posible describir el funcionamiento de la heurística en la situación actual.

Actualmente, SimpliRoute para abordar la problemática expuesta en este trabajo propuso la siguiente solución, intentar ejecutar la heurística de optimización con cuatro configuraciones distintas y de los resultados de estas cuatro ejecuciones entregar la solución que menor *SimplifyScore* tenga.

Se dice que el proceso de optimización *intenta* ejecutar cuatro versiones porque a medida que va ejecutando las diferentes configuraciones va calculando el tiempo que le queda restante, por lo que si estima que con la siguiente ejecución se superarán los 15 minutos de ejecución, entonces no continua. Las configuraciones que se intentan ejecutar son las heurísticas 1, 9, 17 y 7 de la tabla A.1, en ese orden.

Los problemas que tiene lo anterior es que la estimación del tiempo de ejecución de la heurística casi nunca es precisa, lo que puede llevar a tomar una decisión equivocada. Además, al realizar diversas ejecuciones de la heurística de optimización, el cliente tiene que esperar a lo menos cuatro veces más de lo que esperaría con una sola ejecución.

Esta es la situación actual contra la que se comparará la solución propuesta en la sección 2.4.2 (cap. 2).

6.3. Validación importancia de la variación de los parámetros seleccionados

En esta sección se mostrará que los parámetros seleccionados son relevantes al momento de optimizar y que diferentes configuraciones conducen a diferentes resultados. Para llevar a cabo esta validación se realizará un análisis similar al realizado en la sección 2.3. Notar que se utilizará la base de instancias descritas en la sección 2.2.2, con la salvedad que en los casos en que las instancias tienen más de 40 puntos, se escogieron al azar 1001 instancias de un total de 1938. La razón de escoger una muestra se debe a que obtener las soluciones de las 1938 instancias toma aproximadamente 7 horas por configuración probada, por lo que realizar los experimentos con la totalidad de las instancias tomaría demasiado tiempo.

Para validar la selección de los parámetros se conducirán cuatro experimentos, uno por cada nivel presentado en la sección anterior. Para cada uno de los experimentos se escogió una configuración base llamada *configuración 1* y una configuración de prueba llamada *configuración 2*. Destacar que las configuraciones 1 y 2 son iguales en todos los parámetros excepto en uno, de esta forma determinar si dicha modificación genera un impacto. Las configuraciones testeadas en cada uno de los experimentos se encuentran resumidas en la tabla 6.1.

Luego de haber calculado las soluciones que entrega la heurística en ambos casos, se comparan y se determina la cantidad de casos en que se mejora y empeora la solución y

también se evalúa la mejora porcentual de la solución (ecuación 6.2). Cabe recordar que para medir y evaluar que tan buena es una solución, se determinó que la métrica a utilizar sería la denominada **SimplifyScore**. Esta métrica corresponde a la función objetivo de la heurística de optimización de SimpliRoute.

$$variacion\,\% = \frac{SS_{conf2} - SS_{conf1}}{SS_{conf1}} \tag{6.2}$$

Nro. Exper.	Nivel testeado	Configuración 1	Configuración 2
1	Nivel 1	*ByArray*	*ByNewMethod*
2	Nivel 2	*arrayOfDemands*	*arrayOfOnes*
3	Nivel 3	*False*	*True*
4	Nivel 4	$\alpha = 0.2, \beta = 0.2, \gamma = 0.4$	$\alpha = 0.4, \beta = 0.4, \gamma = 0.2$

Tabla 6.1: Resumen de las variaciones en las configuraciones realizadas en los experimentos.

En la figura 6.1 se encuentran los resultados de los experimentos. A partir de estas gráficas, se puede concluir que para los diferentes niveles de parámetros propuestos, escoger una u otra configuración provoca que se generen diferentes soluciones, lo que se traduce en casos en que se mejora la solución y casos en que se empeora la solución.

Al igual que en lo mostrado en la sección 2.3, nuevamente las instancias *más grandes* son las que usualmente se ven más afectadas por la variación en la configuración. Esto ocurre ya que al ser el problema más grande, el espacio de soluciones también es más grande, por lo que es más probable que existan más óptimos locales[10], por lo que al variar el punto de partida, la heurística encuentra óptimos locales distintos. Una de las dificultades que tienen las heurísticas es que una vez que encuentran un óptimo local, les es muy difícil salir de ahí.

Por otro lado, estudiando los histogramas de las mejoras porcentuales de la métrica **SimplifyScore** (función objetivo de la heurística de optimización), en particular en los casos en que las instancias tienen más de 40 puntos, se puede observar claramente que existen casos en que la variación porcentual de la solución puede llegar a alcanzar hasta un 80 % de mejora, mientras que existen casos en que la solución puede llegar a empeorar un 50 %. Dicha variación porcentual se mide entre el valor obtenido en la configuración 1 y la configuración 2.

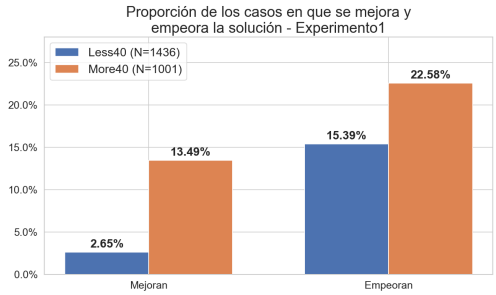
Resulta interesante observar estos histogramas, ya que los valores extremos pueden distorsionar métricas como el promedio. Por ejemplo, si se observa la figura 6.1g, se tiene que una parte importante de las instancias mejoradas llegan a mejorar su solución entre un 5-25 %, pero las instancias que empeoran se encuentran concentradas en el intervalo 5-15 %, por lo que los casos extremos hacen que el promedio se mueva más hacia el centro. Algo similar ocurre al observar el gráfico de la figura 6.1h pero en el sentido contrario.

El análisis anterior se concentró en las instancias con más de 40 puntos debido a que como ya se había mostrado en capítulos anteriores, son los casos en que las instancias presentan

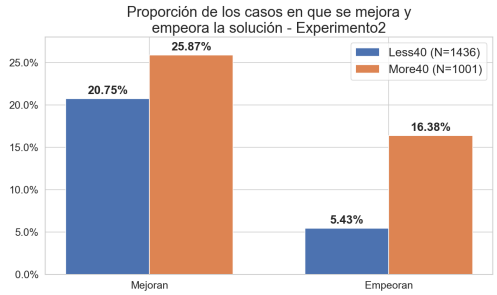
[10]Un punto x^* se dice óptimo local de una función $f(x)$ si para una determinada vecindad de x^* se cumple que $f(x^*)$ es mínimo o máximo en dicha vecindad, dependiendo del sentido de la optimización.

una mayor variación. Además, en estos casos se enfocará el trabajo de memoria.

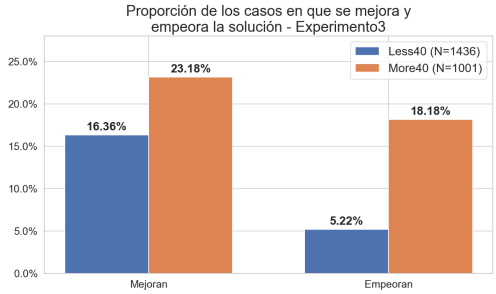
Lo importante a concluir de este análisis es que al variar todas las configuraciones propuestas en la sección 6.2 se obtienen efectos significativos en las soluciones encontradas. Por lo tanto, serán tomados en consideración los cuatro niveles de configuraciones para el proceso ajuste de los parámetros.



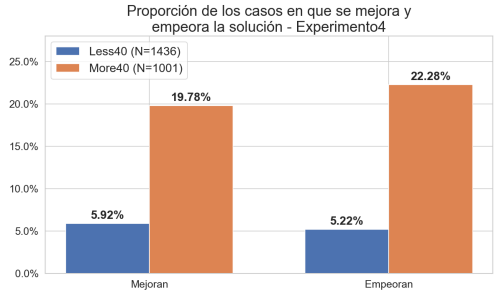
(a) Proporción de casos en que se mejora y empeora variando la configuración del nivel 1.



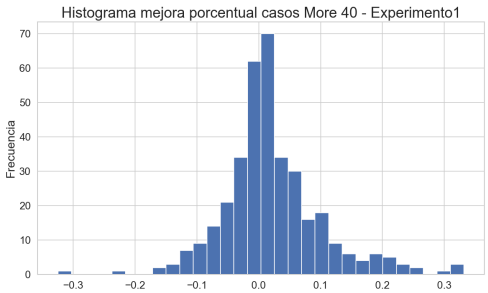
(b) Proporción de casos en que se mejora y empeora variando la configuración del nivel 2.



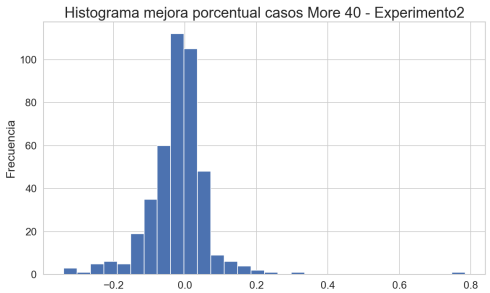
(c) Proporción de casos en que se mejora y empeora variando la configuración del nivel 3.



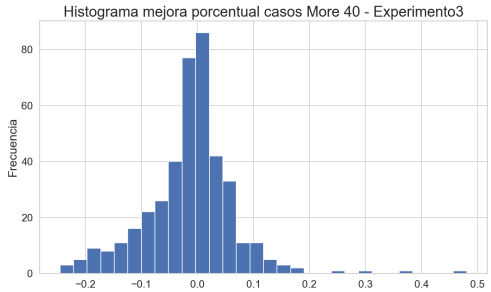
(d) Proporción de casos en que se mejora y empeora variando la configuración del nivel 4.



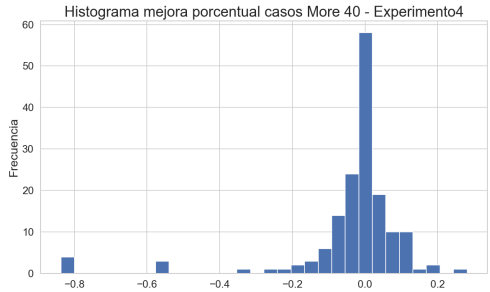
(e) Histograma de la mejora porcentual variando la configuración del nivel 1.



(f) Histograma de la mejora porcentual variando la configuración del nivel 2.



(g) Histograma de la mejora porcentual variando la configuración del nivel 3.



(h) Histograma de la mejora porcentual variando la configuración del nivel 4.

Figura 6.1: Resultados de la variación de los parámetros. Figuras (a - d) muestran la porción de la base de datos que mejora y empeora. Figuras (e - h) muestran los histograma de la mejora porcentual en casos con más de 40 nodos.

Capítulo 7

Selección y procesamiento de datos

De acuerdo a la metodología expuesta en el capítulo 4, corresponde realizar la selección y procesamiento de los datos que se utilizarán en este trabajo.

En primer lugar se realizará la selección de los datos a utilizar, esto corresponde a escoger cuáles son los datos, de la base de datos, que serán candidatos a ser utilizados en este trabajo.

Luego de haber seleccionado este set de datos, se procederá a pre-procesarlos, para así obtener una base íntegra con la cual se trabajará en los pasos siguientes. El objetivo de este procesamiento es quitar de la base de datos aquellas que son consideradas *similares*. El concepto de *similitud* de instancias será detallado más adelante.

Tras el proceso de selección y pre-procesamiento de los datos, se genera la base de instancias con las que se trabajará de aquí en adelante en este informe. Esta nueva base de datos posee 2896 instancias.

7.1. Selección de los datos

Para seleccionar los datos que son relevantes para el estudio, es necesario identificar el origen y el uso que se le da a los datos en el proceso productivo.

En el proceso productivo de la empresa, es decir, en el proceso de optimización de la heurística se genera y almacena diversa información. Es posible categorizar esta información en dos grandes familias. La primera familia corresponde a las *peticiones* que se le realizan a la heurística de optimización y la segunda familia corresponde a la *respuesta* que entrega la heurística de optimización.

Con respecto a estas dos familias, en el presente trabajo interesa fundamentalmente estudiar las peticiones que se le realizan a la heurística de optimización, ya que se optó por construir un portafolio de algoritmos, como se mencionó en la sección 3.2 es necesario volver a calcular las soluciones para cada una de las configuraciones, por lo que no interesa la respues-

ta que entregó en su momento la heurística de optimización, por esta razón, la información de las respuestas será dejada de lado en este estudio.

Es importante destacar que en el flujo de optimización, la petición que se realiza a la heurística se divide en dos pasos. El primer paso consiste en el envío de la petición realizada por el cliente en su navegador web, esta información es almacenada y se le denominará *PeticiónWeb*. El segundo paso consiste en la transformación de la *PeticiónWeb* en un segundo tipo de registro, que se denominará *PeticiónAlgoritmo*.

La gran diferencia entre los tipos de registros mencionados está en que el primero corresponde exactamente a la información que el usuario desea optimizar, mientras que el segundo registro se construye a partir del primero, donde la principal diferencia está en que se calcula la matriz de distancia entre los nodos a visitar. Una vez obtenida la *PeticiónAlgortimo*, esta está en condiciones de ser ingresada a la heurística de optimización.

La empresa ha ido almacenando la información en el formato antes mencionado desde agosto del año 2017 hasta el día de hoy. Sin embargo, dadas las limitaciones de tiempo que tiene este trabajo, solamente se utilizará una porción de la información generada.

Como se puede apreciar en la figura 7.1, la cantidad de peticiones realizadas a la heurística han ido aumentando con el tiempo, prácticamente duplicándose año tras año. También se aprecia que los meses de febrero, marzo, abril, mayo y junio del año 2019 tuvo un alza importante en la cantidad de peticiones (columnas azules). La presencia de estos *peaks* hace pensar que existen registros duplicados u otros problemas, razón que motiva el pre-procesamiento de los datos realizados en la sección 7.2.

Ahora bien, durante los últimos meses del año 2017, durante todo el año 2018 y el primer semestre del año 2019 se generaron 57.632, 258.804 y 327.412 peticiones respectivamente. En base a esto se puede inferir que nuevos clientes han llegado a la plataforma y/o que los clientes están enviando más problemas a ser resueltos. Independiente de cuál sea el fenómeno que está ocurriendo, se puede presumir que durante el 2019 se almacenó una mayor variedad de instancias distintas. Por esta razón, este trabajo se centrará en las instancias generadas durante el año 2019.

Es importante destacar que lo relevante en la confección de la base de datos, es procurar que la base pueda representar a todas las instancias que entran a la heurística, así, al momento de entrenar los modelos, estos serán capaces de predecir de forma más certera para instancias nuevas. Usualmente, efectos importantes que suelen afectar a los modelos son los de estacionalidad y tendencia. Para estos efectos, no sería problema limitar la base de datos a los meses indicados, ya que los fenómenos de estacionalidad y tendencia sólo pueden afectar el tráfico y esta variable se encuentra almacenada en la matriz de distancia de cada instancia.

7.2. Pre-Procesamiento de datos

Tal como se mencionó en la sección anterior, se desea que la base de datos sea lo más representativa posible en términos de las instancias con las que se entrenará el modelo. Es

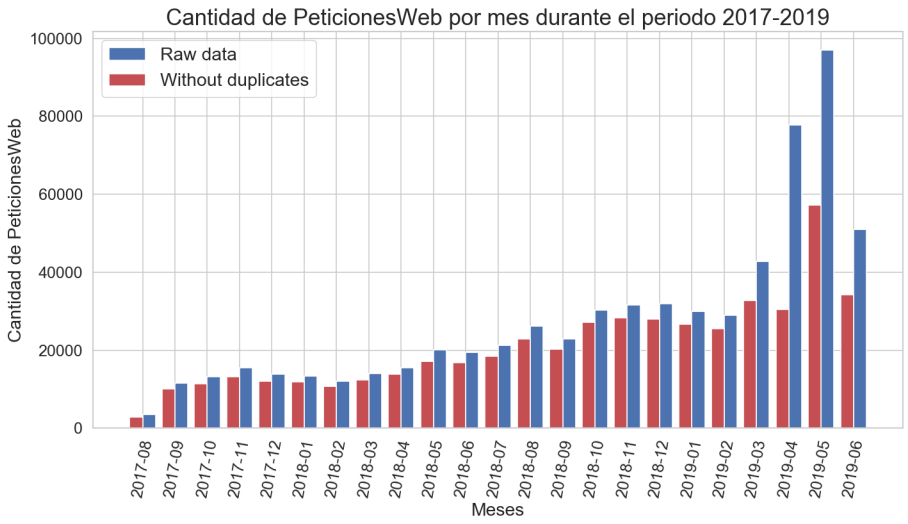


Figura 7.1: Comparación *PeticionesWEB* en la base y removiendo los duplicados exactos.

por esto, que el objetivo del pre-procesamiento de los datos es lograr quitar aquellas instancias que se encuentran repetidas o son *similares*. La razón de excluir estas instancias es evitar que el modelo se sobre ajuste a estos casos y así disminuir su capacidad predictiva.

Ahora bien, las instancias repetidas pueden suceder por diversas razones, como por ejemplo cuando un usuario envía dos veces la misma instancia a optimizar, esta situación puede ocurrir porque el usuario se equivocó o simplemente quería probar la plataforma.

También existen casos en que el usuario envía a optimizar dos instancias muy *similares*, es decir, con unos pocos puntos de diferencia, o con algunos ajustes en la información del vehículo. Estas situaciones se pueden dar, por ejemplo, porque el usuario envió a optimizar una instancia, pero había olvido registrar todos los puntos a visitar o se olvidó de actualizar la información del vehículo. En estos casos, a pesar de que la instancia no es completamente igual, se desean remover, debido a que al ser similares, nuevamente el modelo trataría de ajustarse a estos casos.

Previo a comenzar con el proceso de limpieza, es importante destacar que para comparar las instancias entre sí, solamente se compararán las instancias que hayan sido generadas el mismo día. La razón de considerar esto, es que a pesar de que en dos días distintos se envíe a optimizar exactamente la misma instancia, estos casos sí interesa tenerlos, ya que de esta manera, clientes que deseen resolver su problema de ruteo todos los días, deberían tener un peso relevante en el ajuste del modelo. En particular, si es que su situación se mantiene en el tiempo, se espera que puedan generar un cluster de configuración propio (dada la solución propuesta en la figura 2.7).

Antes de comenzar la descripción del proceso de limpieza, es necesario introducir la estructura que tienen las instancias con las que se trabajarán. En primer lugar, el proceso de limpieza solamente se realizará con las *PeticionesWEB*, ya que una vez que se hayan seleccionado las instancias bastará que nos quedemos con las *PeticionesAlgoritmo* que les preceden.

Ahora bien, la información de las *PeticionesWEB* se divide en tres partes, la primera los nodos a visitar, la segunda la flota de vehículos del problema y la tercera las opciones de optimización. Para obtener más detalle de cómo están conformados cada uno de estos objetos, remitirse al Anexo B donde se encuentra resumida la documentación de la API de SimpliRoute[11].

De las tres partes de la *PeticiónWeb*, para determinar la similitud de las instancias se compararán características obtenidas solamente a partir de los nodos y de los vehículos. La principal razón de esta decisión está en que las opciones de optimización no son características intrínsecas del problema de ruteo, sino que representan preferencias de los clientes y como se mencionó en la sección 3.2, la opinión de los usuarios se dejará de lado en este estudio.

El proceso de limpieza de los datos se dividió en cinco pasos: **(i)** Eliminación de los duplicados, **(ii)** Determinación características a comparar, **(iii)** Escalado de las características, **(iv)** Similitud entre instancias y **(v)** Selección de las instancias.

7.2.1. Eliminación de los duplicados

El primer paso en el proceso de limpieza, es la eliminación de los duplicados exactos. En este paso no se ahondará, ya que es bastante auto-explicativo.

En la figura 7.1 se puede apreciar que una gran cantidad de instancias se encontraban duplicadas, más precisamente, un 24.7 % de las instancias lo estaban. Este fenómeno puede explicar en su mayoría los *peaks* de los meses de febrero, marzo, abril, mayo y junio del año 2019. Luego de remover los duplicados, se continuó a los pasos siguientes con un total de 484.593 instancias.

7.2.2. Determinación de las características a comparar

A partir de esta etapa en adelante, el foco está puesto en determinar la similitud que existe entre las instancias. Para determinar que tan similares son dos instancias se calculará la distancia que existe entre ellas.

Independiente de la medida de distancia que se utilice, es necesario determinar las características que se utilizarán para comparar, es decir, que características conformarán el vector de características.

Como ya se mencionó al comienzo de esta sección, de las tres partes que tienen una *PeticiónWeb* se extraerán características a partir del conjunto de nodos y del conjunto de vehículos. El detalle del significado de cada una de las partes se encuentra en el Anexo B.

Para cada instancia, el vector de características está conformado por las siguientes dimensiones:

[11]Link: https://documentation.simpliroute.com/es/

1. **Número de vehículos:** Corresponde a la cantidad de vehículos por la que está compuesta la flota de la instancia.

2. **Promedio capacidad 1 de los vehículos**[12]**:** Corresponde al promedio de las capacidades 1 que tienen los vehículos.

3. **Promedio capacidad 2 de los vehículos**[12]**:** Corresponde al promedio de las capacidades 2 que tienen los vehículos.

4. **Promedio capacidad 3 de los vehículos**[12]**:** Corresponde al promedio de las capacidades 3 que tienen los vehículos.

5. **Centro del depot:** Esta característica representa el centro de los *depots*[13]. Dado que el centro promedio tiene latitud y longitud, para poder representar esta característica en un solo valor, se sumó la latitud y longitud y dicho valor se multiplicó por su número de cuadrante, como se muestra en la ecuación 7.1. Los cuadrantes corresponden a la división del plano cartesiano, donde el eje X es la longitud y el eje Y la latitud.

$$(\overline{Lon} + \overline{Lat}) * Cuadrante(\overline{Lon}, \overline{Lat}) \tag{7.1}$$

   La razón de multiplicar por el número de cuadrante es que para las instancias que tienen sus centros en los cuadrantes II y IV, la suma de la latitud y la longitud puede tener el mismo valor, pero son instancias distintas (fueron generadas en lugares distintos del mundo), por lo que multiplicarlas por el número de cuadrante, se logra diferenciar estos casos.

6. **Promedio hora inicio turno:** La hora de inicio de turno corresponde a la hora en que un vehículo puede comenzar a ser utilizado, por lo que se utilizará el promedio de estos valores para todos los vehículos de la instancia.

7. **Promedio hora término turno:** La hora de término del turno corresponde a la hora en que el vehículo debe terminar de atender, por lo que se utilizará el promedio de estos valores para todos los vehículos de la instancia.

8. **Número de clientes:** Corresponde a la cantidad de clientes que se deben visitar.

9. **Promedio demanda 1 de los clientes**[14]**:** Corresponde al promedio de las demandas 1 de los clientes.

10. **Promedio demanda 2 de los clientes**[14]**:** Corresponde al promedio de las demandas 2 de los clientes.

11. **Promedio demanda 3 de los clientes**[14]**:** Corresponde al promedio de las demandas 3 de los clientes.

[12]Como muestra la documentación de SimpliRoute, el significado de los valores de las capacidades de los vehículos significan lo que los usuarios estimen conveniente.

[13]Depot: Corresponde al lugar donde los vehículos de transporte inician y/o terminan su recorrido.

[14]Al igual que la capacidad de los vehículos, el significado de los valores de las demandas de los clientes

12. **Centro de los clientes:** Esta característica representa el centro de todos los nodos. Dado que el centro tiene latitud y longitud, para representar esta característica en un solo valor, se sumo la latitud y la longitud y dicho valor se multiplicó por su número de cuadrante. Se utilizó la misma fórmula (eq. 7.1) utilizada para obtener el centro de los *depots*.

13. **Tiempo de servicio promedio:** Corresponde al promedio de los tiempos de servicio indicado para las visitas.

14. **Promedio inicio ventana horaria de los clientes:** Este valor corresponde al promedio de los inicios de las ventanas horarias de los clientes.

15. **Promedio término ventana horaria de los clientes:** Este valor corresponde al promedio de los términos de las ventanas horarias de los clientes.

16. **Clientes con Prioridades:** Esta variable toma el valor 1 si algún nodo tiene prioridad de atención y 0 si es que todos los clientes tienen la misma prioridad.

17. **Clientes tienen Skills Obligatorias:** Esta variable toma el valor 1 si es que algún nodo tiene skills obligatorias y 0 si es que ningún cliente tienen skills obligatorias.

18. **Clientes tienen Skills Opcionales:** Esta variable toma el valor 1 si es que algún nodo tiene skills opcionales y 0 si es que ningún cliente tienen skills opcionales.

19. **Clientes tienen Zonas:** Esta variable toma el valor 1 si es que algún nodo tiene zonas asignadas y 0 si es que ningún cliente tiene zonas asignadas.

20. **Vehículos con Skills:** Esta variable toma el valor 1 si es que algún vehículo tiene skills y toma el valor 0 si es que ningún vehículo tiene.

21. **Vehículos con Zonas:** Esta variable toma el valor 1 si es que algún vehículo trabaja en zonas delimitadas y toma el valor 0 si es que ningún vehículo tiene.

Ahora bien, como el objetivo de estas características es simplemente identificar a las instancias similares, no se busca que estas sean completamente interpretables, sino lo que se busca es crear características que puedan identificar de forma única a las instancias. Por esta razón es que tiene sentido utilizar fórmulas como la utilizada en la ecuación 7.1.

7.2.3. Escalado de las características

Una vez calculadas las características, es sencillo darse cuenta que las escalas de cada una de las variables puede llegar a ser bastante distintas entre sí.

Como se desea poder comprar todas las características entre sí, es necesario que estén en una misma escala, por lo que es necesario escalar. La técnica de escalado que se utilizará se

están definidos por los usuarios y significan lo que ellos estimen conveniente. No obstante, el significado de las demandas deben ser iguales al significado de las capacidades de los vehículos.

la denominada *MinMax Scaling*. Se escoge esta técnica porque permite que todos los valores queden en la escala estipulada, la que usualmente está entre 0 y 1.

La técnica de *MinMax Scaling* se define como sigue. Sea N el conjunto de características y sea V el conjunto de vectores de características donde V_{i} representa el vector de características del registro i. Se calcula el mínimo y el máximo para cada característica $n \in N$ sobre el conjunto V de la siguiente forma: $min_n = \text{mín}_{\text{i} \in V} \{V_{\text{i}}^n\}$ y $max_n = \text{máx}_{\text{i} \in V} \{V_{\text{i}}^n\}$. Luego, para transformar un vector de características $X \in V$, se aplica la siguiente transformación. Notar que x^n corresponde al valor de la característica n en X

$$x_{scaled}^n = \frac{x^n - min_n}{max_n - min_n} \qquad \forall n \in N \tag{7.2}$$

Tras aplicar la técnica de *MinMax Scaling* se obtiene una escala única para todas las características, la cual está definida en el intervalo $[0, 1]$.

Cabe destacar que para escalar utilizando esta técnica, el vector X representa una única característica, por lo que cada componente de X es el valor que tiene la característica x en la instancia i.

Una segunda técnica usualmente utilizada y que no se utilizará en este caso es la técnica de *Normalización*. Lo que hace esta técnica es normalizar los datos, es decir, ajustar los valores a una distribución normal. Esta metodología no sirve para este caso, ya que tras normalizar no necesariamente se logra que las características estén en una escala similar, por lo que seguirían sin ser comparables entre sí.

7.2.4. Similitud entre instancias

Como se menciona en el segundo paso del proceso de limpieza de datos (7.2.2), la similitud entre dos instancias se obtendrá a partir de la distancia que existe entre dos puntos. En particular se utilizará la distancia euclidiana. La razón de utilizar esta distancia se debe a que es una métrica bastante sencilla de interpretar y de aplicar.

Es importante recordar que la distancia euclidiana se obtiene de la siguiente forma. Se utilizará indistintamente $\text{d}_E(X, Y)$ y $\text{d}(X, Y)$ para referirse a la distancia euclidiana. Sean X, Y dos vectores con n dimensiones, entonces la distancia euclidiana se define como sigue.

$$\text{d}_E(X, Y) = \sqrt{\sum_{\text{i}=1}^{n} (x_{\text{i}} - y_{\text{i}})^2} \tag{7.3}$$

Habiendo descrito la métrica de distancia que se utilizará para determinar la similitud entre dos instancias se procede a calcular la similitud que existe entre estas. Este procedimiento será descrito para un par de instancias X, Y cualquiera, con $X \neq Y$. Ahora bien, se definirá

que la cardinalidad del vector de características es igual a n, y en este caso en particular $n = 21$.

Entonces, para ver qué tan similares son dos instancias se comenzará el análisis evaluando los casos extremos, es decir, cuando dos instancias son perfectamente distintas y cuando son perfectamente iguales. Para esto es importante recordar que las características fueron escaladas y todas están en el intervalo $[0, 1]$

En el primer caso, dos instancias son perfectamente distintas cuando para las n dimensiones se cumple que $(x_\mathrm{i} - y_\mathrm{i}) = 1$, entonces se puede verificar que utilizando la ecuación 7.3, se obtiene que la distancia máxima es igual a $\sqrt{n}$, con $n = 21$, entonces $\sqrt{n} = \sqrt{21} = 4.58$.

Para el segundo caso, dos instancias son perfectamente iguales cuando para las n dimensiones se cumple que $(x_\mathrm{i} - y_\mathrm{i}) = 0$, entonces se puede verificar que al aplicar la ecuación 7.3, se obtiene que la mínima distancia que puede existir entre dos instancias es 0.

Habiendo encontrado estos casos extremos, ahora se puede definir el porcentaje de desigualdad que existe entre dos instancias, la cual se definirá de la siguiente manera.

$$\%desigualdad = \frac{\mathrm{d}(X, Y)}{\sqrt{n} - 0} = \frac{\mathrm{d}(X, Y)}{\sqrt{n}} \tag{7.4}$$

A partir de lo anterior, se puede definir el porcentaje de igualdad, el que queda definido por la siguiente expresión.

$$\%igualdad = 1 - \frac{\mathrm{d}(X, Y)}{\sqrt{n}} \tag{7.5}$$

En base al porcentaje de igualdad, se puede establecer de forma sencilla una cota para determinar cuando dos instancias son similares y cuando no.

Para este trabajo, se considerará que dos instancias son iguales, cuando su porcentaje de igualdad es mayor o igual a un 95 %. Reemplazando este valor en la ecuación 7.5 se obtiene que cuando la distancia sea menor a 0.23, se está frente a dos instancias *iguales*. El despeje anterior se encuentra en la ecuación 7.6.

$$0.95 = 1 - \frac{\mathrm{d}(X, Y)}{\sqrt{21}} \Rightarrow \mathrm{d}(X, Y) = \sqrt{21} * (1 - 0.95) = 0.23 \tag{7.6}$$

Finalmente, se comparará el resultado obtenido en la ecuación 7.6 en el histograma de las distancias entre los pares, gráfico 7.2. La razón de realizar esto es que sólo interesa excluir de la base de datos la cola inferior de la distribución de los datos.

Sin pérdida de generalidad, para realizar dicha verificación, se escogió un día cualquiera del mes de febrero, ya que se puede suponer que las instancias que llegan durante un día son

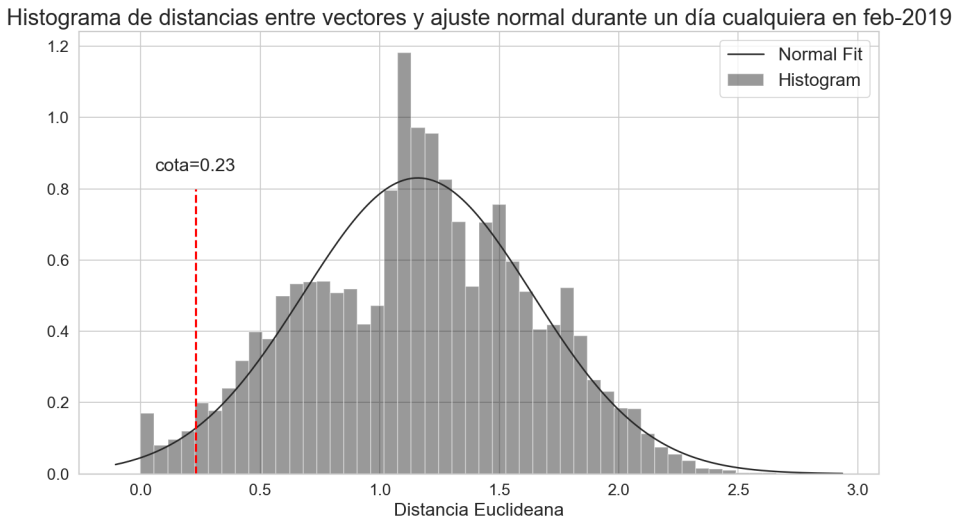


Figura 7.2: Histograma de distancia entre vectores durante un día cualquiera de febrero.

similares entre los días. En este día se generaron 1212 instancias que no estaban exactamente duplicadas.

Para este día, se llevaron a cabo todos los pasos descritos hasta el momento. Luego se procedió a calcular la distancia entre todos los pares de instancias, obteniendo 473851 pares. A partir de esta información se obtiene que el porcentaje de pares de instancias que están bajo la cota es de un 2.78 %. Esto quiere decir que una porción menor de los casos se desea que queden fuera de la base.

Además se puede verificar empíricamente que el valor de esta cota es igual a 1.93 veces la desviación estándar, lo que significa que se está manteniendo la variabilidad de los datos y no se está perdiendo información relevante. El promedio de las distancias es 1.16 y la desviación estándar es 0.48.

Es interesante también ver cómo se encuentra distribuida la información de la distancia entre los pares de instancias, ya que si estos valores hubiesen estado agregados cerca del cero o de la cota, podría significar que en su mayoría las instancias son bastante similares. Caso contrario, si hubieran estado centrados cerca de 4.58, mostraría que las instancias son muy distintas entre sí. La gráfica 7.2 permite verificar que ninguno de los casos extremos se da y que existe variabilidad entre las instancias y con la aplicación de esta metodología, dicha variabilidad se mantiene.

7.2.5. Remoción de las instancias consideradas *similares*

Finalmente, el objetivo principal de este paso remover de la base de datos aquellas instancias que se consideren similares, logrando dejar solo una de ellas. Se debe tener en mente también, que lo que se busca con esta limpieza es disminuir lo menos posible la base, por lo que hay que tener cuidado al momento de realizar la selección.

Tras haber realizado el procesamiento de los datos en los pasos anteriores, se ha logrado determinar para **cada día de información**, cuales son los pares de instancias que están relacionados entre sí. Esta información se recopilada por la matriz M. Esta matriz posee la información de las distancias que existe entre todo par de instancias. Por ende, la dimensión de dicha matriz es de $|I|x|I|$, con I igual al conjunto de las instancias generadas en un día. Entonces, los valores de la matriz representan la distancia, *i.e.* $M_{\mathrm{i},j} = \mathrm{d}(\mathrm{i}, j)$ con $\mathrm{i}, j \in I$.

Recordar que dos instancias i, j se consideran similares cuando $\mathrm{d}(\mathrm{i}, j) < cota$

Ahora bien, la gran dificultad de este paso está en decidir con que instancia quedarse, es decir, cuando se cumple que $M_{\mathrm{i},j} \leq cota$ se debe decidir si escoger i o j para pasar a conformar la base final. A simple vista, esta decisión puede resultar trivial, razón por la cual se mostrarán algunos ejemplos de por qué no lo es.

Ej. 1 Sean dos instancias, representadas por los vectores X e Y. Si $M_{X,Y} < cota$ entonces se dice que $X \sim Y$. Luego, se puede escoger arbitrariamente con cual quedarse.

Ej. 2 Sean tres instancias, representadas por los vectores X, Y, Z. Si $M_{X,Y} < cota$, $M_{Y,Z} < cota$ y $M_{Z,X} > cota$, se tiene un caso donde el único elemento común es la instancia Y, por lo que convendría removerla, para tener una base con dos instancias. Sin embargo, remover X no traería ningún beneficio, ya que también habría que remover Y o Z y la base quedaría con una sola instancia. En este caso, es sencillo verificar que la elección correcta es remover Y, debido a que la cantidad de instancias es reducida.

Ej. 3 Considerar cinco instancias representadas por los vectores V, W, X, Y, Z. Se tiene que $V \sim W \wedge V \not\sim \{X, Y, Z\}$, $W \sim X \wedge W \not\sim \{Y, Z\}$, $X \sim Y \wedge X \not\sim \{V, Z\}$ y por último $Y \sim Z \wedge Y \not\sim \{V, W\}$. En este caso, ya no resulta tan sencillo determinar la mejor elección. Este caso puede ser considerado como bueno, ya que intuitivamente, basta con remover las instancias intermedias, para poder así obtener una base de datos lo más grande posible. Si una de estas decisiones fuese errada, entonces se podría terminar con una base bastante reducida.

Con los ejemplos anteriores, se busca mostrar que cuando se tiene un número de instancias grande ($|I| > 3$) la elección de instancias a remover es cada vez mas compleja, debido a que por cada instancia en I, se deben realizar $(n-1)$ comparaciones, lo que se traduce en un problema computacional del orden $O(n^2 - n)$.

Dado que el foco del trabajo no está en resolver el problema de la correcta remoción de instancias, se decidió elaborar una heurística sencilla para seleccionar las instancias más favorables y así no perjudicar fuertemente el tamaño de la base de datos.

La regla de selección se encuentra resumida en el algoritmo 1, esta consiste en lo siguiente. Para cada instancia se debe identificar con cuales otras instancias está relacionada, dicha información es guardada en D. Cada índice de D representa a una instancia y D_{i} indica con que instancias está relacionada i. Recordar que una instancia está relacionada con otra sí $\mathrm{d}(\mathrm{i}, j) < cota$. Luego D es ordenado de menor a mayor en función de la cardinalidad de D_{i}. Una vez ordenado D, se van seleccionando los elementos en orden, desde el primero hasta el último y son llevados a la base final si y solo sí el nuevo elemento no es similar a ningún otro

de la base.

Algoritmo 1 Regla de selección de instancias

Require: $M \leftarrow$ *Matriz de distancia entre instancias* $\wedge$ $c \leftarrow$ *cota*
  $n \leftarrow$ *Cantidad de instancias*
  $D \leftarrow$ *Inicializamos diccionario de relaciones vacío*
  for i $\in \{1, \ldots, n\}$ **do**
    d $\leftarrow$ *Creamos lista de relaciones vacía*
    for $j \in \{1, \ldots, n\}$ **do**
      if $M_{ij} < cota$ **then**
        d $\leftarrow$ *Añadimos a* j *a la lista*
      end if
    end for
  end for
  Ahora D_{i} *contiene una lista con todas aquellas instancias con que* i *no se puede relacionar.*
  $D \leftarrow$ *Ordenamos D de menor a mayor según el largo de* D_{i}
  BaseFinal $\leftarrow$ *Lista de instancias que quedan*
  for i $\in D$ **do**
    if i *no es similar a ninguna instancia en BaseFinal* **then**
      BaseFinal $\leftarrow$ *Añadimos a* i *a la lista*
    end if
  end for

7.2.6. Resultados pre-procesamiento de datos

Tras aplicar todo el proceso de pre-procesamiento de datos para el los meses de enero, febrero, marzo, abril, mayo y junio, se obtiene un set de instancias que tiene 29090 registros, como se puede apreciar en la figura 7.3.

Ahora bien, en la sección 3.2 (cap. 3) se indicó que el foco del trabajo se centrará en los meses de enero, febrero y marzo. Además, se considerarán aquellas instancias que tengan al menos 40 puntos y al menos dos vehículos. Tras hacer este filtro, se obtiene una base con 2896 registros que serán las instancias con las que se trabajará en este trabajo de memoria.

7.2.7. Limitaciones

Finalmente, es necesario comentar y dejar documentadas cuales son las principales limitaciones que tiene esta metodología de pre-procesamiento de datos.

En primer lugar, es necesario evaluar si la distancia euclidiana es la mejor métrica para determinar la distancia que existe entre dos instancias. La razón de esto, es que a pesar de ser una métrica fuertemente usada en muchos ámbitos por su sencillez e interpretabilidad, cuando el número de dimensiones es grande, el peso de cada una de las características puede hacerse bastante irrelevante, por lo se podría llegar a considerar similares instancias que no lo son para nada. No obstante, esta métrica asegura que las instancias que efectivamente son iguales sean detectadas como iguales.

Una segunda consideración que se debe tener, es que la metodología para seleccionar las instancias en el último paso, puede no ser la óptima y puede que las decisiones que se estén tomando no sean las correctas, lo que desencadena que muchas instancias sean dejadas de lado cuando en realidad podrían ser consideradas. Para sortear este problema, se propone modelar este problema de programación entero que busque maximizar el tamaño de la base de datos.

De todos modos, a pesar de las limitaciones que tiene la metodología implementada, esta asegura que las instancias similares sean removidas de la base final y gracias a que se tiene una gran cantidad de instancias, a pesar de que la reducción es bastante significativa (fig. 7.3), la base queda con un número de instancias interesantes a estudiar.

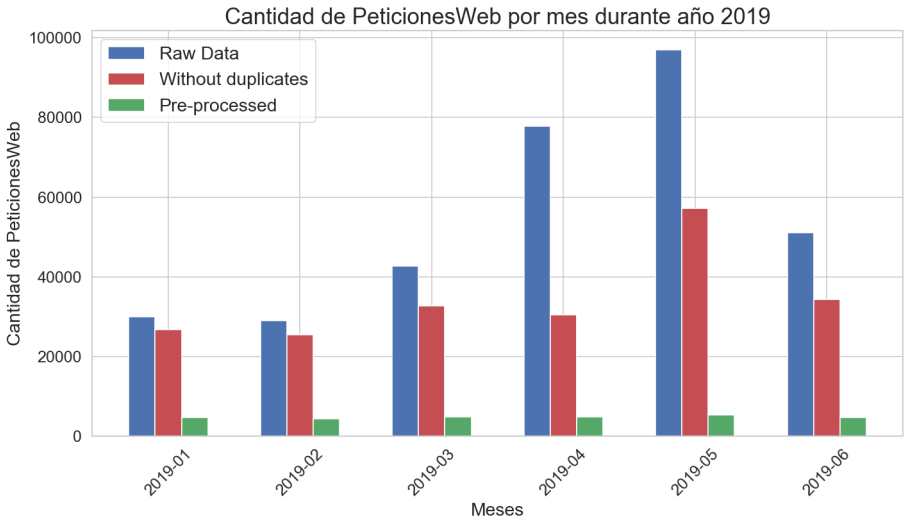


Figura 7.3: Cantidad de instancias que quedan tras el proceso de limpieza de datos.

Capítulo 8

Caracterización de las instancias

De acuerdo a la metodología expuesta en el capítulo 4, en este capítulo se llevará a cabo la tercera etapa de esta. En esta etapa se busca describir y caracterizar a las instancias, también en caso de ser necesario, realizar transformaciones relevantes a los datos.

Importante notar que las instancias con las que se trabajará de aquí en adelante, corresponden a las seleccionadas en el capítulo anterior. También es importante destacar que esta base fue separada en dos, con el objetivo de contar con una **base de entrenamiento** y una **base de testeo**. Esta separación se realizó de forma aleatoria y antes de llevar a cabo cualquier análisis, para así evitar generar un sobre-ajuste de los modelos propuestos (Géron 2017). Por esta razón, en este capítulo se analiza exclusivamente la **base de entrenamiento**.

Este capítulo está dividido en dos secciones. En la primera sección se revisarán los resultados obtenidos tras ejecutar la heurística de optimización para las instancias en la base de entrenamiento con las diferentes configuraciones propuestas en el capítulo 6 (ver tabla A.1 en el Anexos A).

En la segunda sección de este capítulo se presentarán las características que serán utilizadas para describir a las instancias. También en esa sección se realizará un análisis exploratorio de las características, con la finalidad de determinar si es posible identificar ciertos grupos de instancias.

8.1. Análisis descriptivo de las soluciones obtenidas tras la ejecución de la heurística con las diferentes configuraciones

A continuación se analizarán los resultados obtenidos tras la ejecución de la heurística de optimización con sus diferentes configuraciones (propuestas en el capítulo 6). También se ejecutó la heurística con la configuración actual, la descrita en la sección 6.1, para luego poder calcular si existe mejoría con la(s) nueva(s) configuraciones, de esta manera se calcularon 25

soluciones para la base de instancias.

Notar que la heurística de optimización fue ejecutada en una CPU con un procesador *dual-proccesor dual core Intel(R) Xeon(R) Platinium* de 3.00GHz, 4GB de RAM y un sistema operativo de 64-bit, el mismo que actualmente utiliza SimpliRoute para ejecutar su heurística de optimización.

Es importante recordar que para medir y comparar qué tan buena es una solución se utilizarán dos métricas, **SimplifyScore** y el **tiempo de ejecución** de la heurística de optimización.

También recordar que en el capítulo anterior se construyó la base de instancias con las que se probarían las diferentes configuraciones. Esta base de instancias posee 2896 instancias.

Tras haber resuelto estas instancias mediante la heurística de optimización de SimpliRoute con las 24 configuraciones propuestas y la configuración actual, se detectó que habían 68 instancias que resultaban ser infactibles, es decir, que no existía ninguna solución posible. Estas instancias fueron removidas de la base de datos, ya que no entregan información adicional, quedando una base con 2828 instancias.

A partir de esta base de datos se construyó la **base de entrenamiento** y la **base de testeo**. Un 30 % de los datos fueron destinados a conformar la base de testeo, por lo que en la base de entrenamiento quedaron 1979 instancias.

8.1.1. Desempeño de las configuraciones

Un primer resultado relevante obtenido tras resolver las instancias con todas las configuraciones propuestas, es que la heurística de optimización no siempre logra encontrar una solución antes de los 15 minutos, es decir, no entrega ninguna solución. Como se puede observar en el gráfico 8.1, en la mayoría de los casos la heurística de optimización alcanza a encontrar y entregar una solución. Sin embargo, hay cuatro configuraciones en que alrededor de 1300 instancias no se logra encontrar una solución. Una de las explicaciones que se le da a este fenómeno es que estas configuraciones entregan un muy mal punto de partida, por lo que la heurística de optimización no alcanza a entregar una solución en los 15 minutos.

Es importante destacar que no encontrar una solución es una situación bastante desfavorable para SimpliRoute, ya que básicamente no se le está brindando el servicio prometido al cliente. Por esta razón, dado que para estos casos no se tiene un valor del *SimplifyScore*, se consideró como si la heurística de optimización hubiese encontrado la peor solución posible. En esta base, la peor solución posible corresponde al máximo *SimplifyScore* encontrado en cualquier ejecución de la heurística de optimización multiplicado por 10. De esta forma, en los casos que no se encuentra una solución, su *SimplifyScore* es un orden de magnitud más grande que el máximo valor del *SimplifyScore* presente en la base.

Las configuraciones número 13, 14, 15 y 16 no serán excluidas del análisis, ya que a pesar de que en muchos casos encuentran una solución *indeseada*, existen otros casos en que la

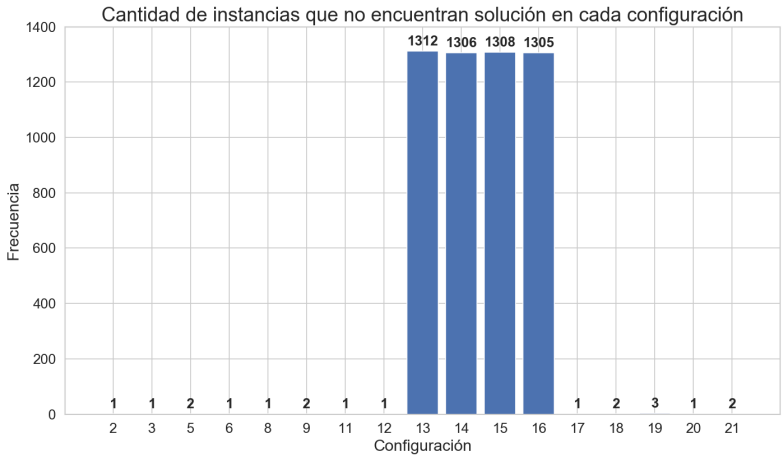


Figura 8.1: Cantidad de instancias que no encuentran solución en menos de 15 minutos por configuración probada. Las configuraciones en que siempre se encuentra una solución fueron omitidas de la gráfica.

solución mejora su **SimplifyScore** y/o su **tiempo de ejecución**, y estas mejoras son los beneficios que se quieren capturar de forma inteligente.

Por otro lado, para determinar cuánto mejora/empeora una solución con respecto a la situación actual de la heurística de optimización, se calculará la variación porcentual que existe entre una solución calculada con la configuración i y la solución entregada por la heurística de optimización en su situación actual. Se comparará tanto el valor del **SimplifyScore** y el **tiempo de ejecución** obtenido. La variación porcentual de dichas métricas se calculan usando las ecuaciones 8.1 y 8.2, a la situación actual se le denominará *asis*. Recordar que el modelo mejora cuando la variación porcentual es menor a cero, esto debido a que se busca minimizar el **SimplifyScore**.

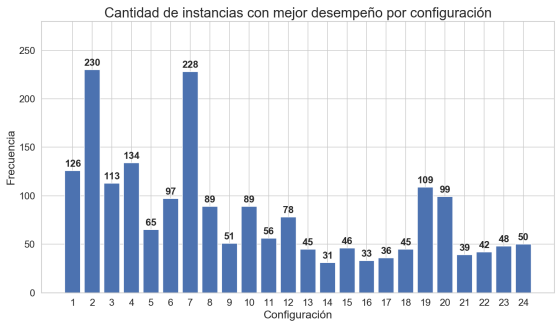
$$SS\ variación\ \%_{\mathrm{i}} = \frac{SS_{conf_{\mathrm{i}}} - SS_{asis}}{SS_{asis}} \tag{8.1}$$

$$ET\ variación\ \%_{\mathrm{i}} = \frac{ET_{conf_{\mathrm{i}}} - ET_{asis}}{ET_{asis}} \tag{8.2}$$

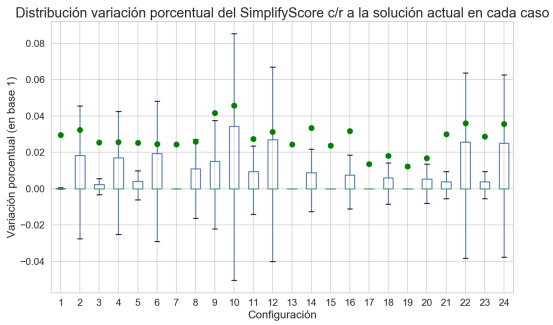
Habiendo calculado la variación porcentual de la solución obtenida con cada configuración para cada instancia, se procedió a determinar si dicha variación porcentual es estadísticamente distinta de cero.

Para llevar a cabo lo anterior se utilizó en test estadístico denominado *test de hipótesis para la media de una muestra*. Este test se realiza con la finalidad de excluir aquellas configuraciones que tengan una variación porcentual estadísticamente iguales a cero.

En este test, la hipótesis nula (H_0) plantea que el promedio de la variación porcentual es igual a cero, mientras que la hipótesis alternativa (H_A) indica que el promedio de la variación porcentual es distinta de cero, como lo muestra la ecuación 8.3. El estadístico



(a) Cantidad de instancias que tienen mejor desempeño por configuración.



(b) Distribución de la variación porcentual del SimplifyScore con respecto a la situación actual en cada configuración.

Figura 8.2: Resultados agrupados por configuración.

usado corresponde al mostrado en la ecuación 8.4.

$$\begin{aligned} H_0 : &\quad \overline{SSvariacion_{\mathrm{i}}\,\%} = 0 \\ H_A : &\quad \overline{SSvariacion_{\mathrm{i}}\,\%} \neq 0 \end{aligned} \tag{8.3}$$

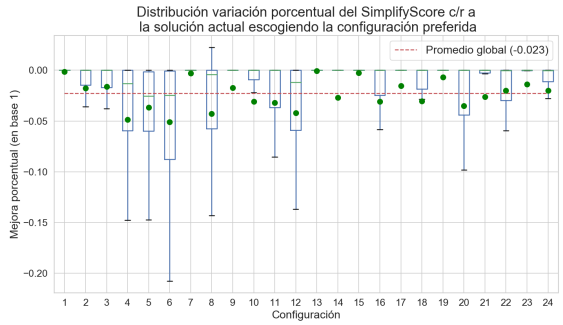
$$t = \frac{\overline{SSvariacion_{\mathrm{i}}\,\%} - 0}{\frac{\sigma}{\sqrt{n}}} \tag{8.4}$$

El resultado de este test arrojó que todas las configuraciones son estadísticamente distintas de cero, a un nivel de significancia del 99 %, lo que permite continuar con el análisis sin descartar ninguna configuración. El detalle de los resultados se encuentran adjuntos en la tabla C.1 del Anexo C.

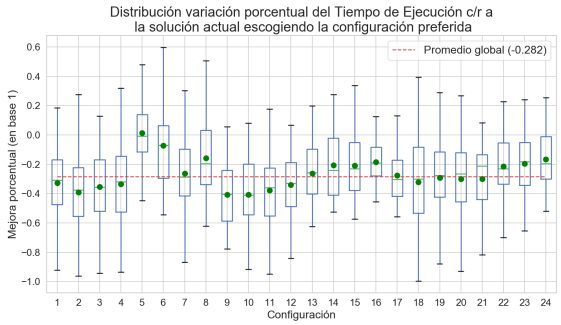
Ahora bien, ya calculadas las diferentes soluciones para las instancias con cada una de las 24 soluciones propuestas, es necesario determinar cuál de todas las configuraciones es la que tiene mejor desempeño. A la configuración que mejor desempeño logra se le denominará *configuración preferida*. Para realizar lo anterior, recordar que para comparar las soluciones se utilizarán las métricas **SimplifyScore** y el **tiempo de ejecución**.

La regla de comparación que se utilizará es la conocida como *comparación lexicográfica*. En este caso particular la regla indica que una solución es mejor que otra cuando el **SimplifyScore** es menor. Cuando el **SimplifyScore** es igual, se escoge la configuración que menor **tiempo de ejecución** tiene. Aplicando esta regla se obtiene la *configuración preferida* de cada instancia.

El gráfico 8.2a muestra la cantidad de veces que una configuración resulta tener el mejor desempeño. A partir de esta gráfica no se identifica una notoria preferencia por cierta configuración, claro está que las configuraciones 2 y 7 son las que más veces se escogieron, pero entre ambas representan solo un 23 % de los casos. Además, analizando la gráfica 8.2b se aprecia que ninguna configuración, en promedio, es mejor que la situación actual. Incluso, resulta muy difícil determinar cuál de las 24 configuraciones es mejor.



(a) Distribución de la variación porcentual del SimplifyScore con respecto a la situación actual, escogiendo la *configuración preferida*.



(b) Distribución de la variación porcentual del tiempo de ejecución con respecto a la situación actual, escogiendo la *configuración preferida*.

Figura 8.3: Resultados de seleccionar la *configuración preferida*.

A partir de los resultados anteriores, se reafirma el problema abordado en este trabajo, ya que no existe una única configuración de la heurística de optimización que de forma consistente logra el mejor desempeño en todos los casos.

Ahora bien, la solución propuesta en la sección 2.4.2 (cap. 2) indica que lo que se busca hacer es para cada instancia ejecutar la heurística de optimización con aquella configuración que genera el mejor desempeño. Entonces, a continuación se analizará cuáles son los beneficios de escoger la configuración correcta para cada instancia. Para comenzar el análisis se determinará en cuántos casos se mejora o empeora la solución con respecto a la situación actual escogiendo la *configuración preferida*.

Cabe destacar que cuando se comparan dos soluciones y estas tienen el mismo **SimplifyScore**, se escoge aquella solución que tiene un menor **tiempo de ejecución**. Cuando se comparan los **tiempos de ejecución** resulta necesario definir un intervalo en que dos tiempos de ejecución son iguales, ya que es muy poco probable que estos sean perfectamente iguales. En particular, se definió una vecindad de dos segundos alrededor del **tiempo de ejecución** de la situación actual, es decir, t_{i} se dice igual a t_{asis} si $t_{\text{i}} \in [t_{asis} - 2, t_{asis} + 2]$, donde t_{i} es el *tiempo de ejecución* de la configuración i.

Los resultados de dicha comparación se encuentran expuestos en la matriz de confusión mostrada en la tabla 8.1. Siguiendo la regla lexicográfica ya descrita, se obtiene que del total de casos, 1354 (68.4 %) instancias mejoran su solución, 45 (2.3 %) la empeoran y 580 (29.3 %) no la varían, es decir, de escoger la *configuración preferida* en su mayoría se mejora la solución entregada por la heurística de optimización.

Complementado el análisis, en los gráficos 8.3a y 8.3b se aprecia la distribución de la variación porcentual que tiene el **SimplifyScore** y el **tiempo de ejecución** sobre la base de instancias. En ambas métricas se aprecia que de seleccionar la *configuración preferida* para cada instancia, en promedio se mejora el **SimplifyScore** en un 2.3 % y el **tiempo de ejecución** en un 28.2 %.

Las gráficas 8.3a y 8.3b muestran las mejorías obtenidas a nivel general. Sin embargo, si

		Tiempo de ejecución			
		Mejoran	*Igual*	*Empeoran*	*Total*
SimplifyScore	*Mejoran*	350	308	82	**740**
	Igual	614	580	44	**1238**
	Empeoran	0	0	1	**1**

Tabla 8.1: Matriz de confusión de la cantidad de casos que mejora y empeoran su solución al escoger la *configuración preferida* con respecto a la situación actual, analizando las métricas **SimplifyScore** y el **tiempo de ejecución**.

se separan los casos en que el **SimplifyScore** varía de los que no varía se obtienen resultados interesantes. Por un lado, analizando los casos donde el *SimplifyScore* varía ($N = 741$), se obtiene que las soluciones presentan una mejoría promedio del 6.0 % (fig. 8.4a) y el **tiempo de ejecución** mejora en un 22.3 % promedio (fig. 8.4b). Mientras que si analizamos los casos en que el **SimplifyScore** no varía ($N = 1238$), es decir, independiente de la configuración usada la solución encontrada es la misma, el **tiempo de ejecución** mejora en promedio un 31.8 % (fig. 8.4c).

Comparando la gráfica 8.3a con la 8.4a, en la segunda se logra apreciar claramente cómo el hecho de escoger la configuración correcta entrega mejores soluciones, ya que este efecto no se ve atenuado por los casos en que no varía. Y las gráficas 8.4b y 8.4c muestran que en ambos casos el tiempo de ejecución mejora.

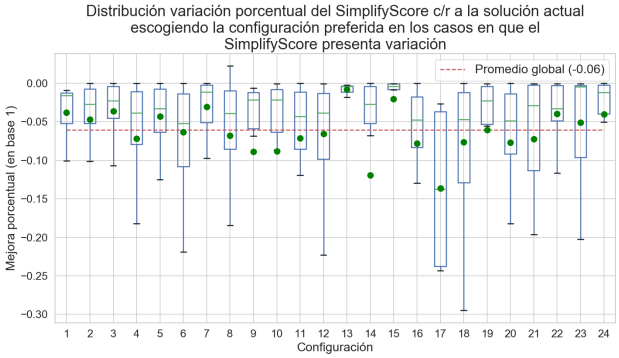
A partir de lo expuesto anteriormente, se puede reafirmar que escoger la *configuración preferida* para cada instancia permite mejorar tanto la función objetivo como el tiempo de ejecución de la heurística de optimización, por ende mejorar su desempeño. Además, las configuraciones propuestas tienen un impacto significativo en la mejora de las soluciones.

8.2. Caracterización de las instancias

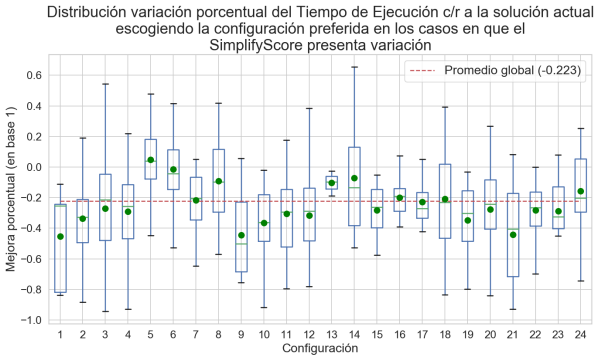
Para continuar con el desarrollo de la solución propuesta (sección 2.4.2), en esta sección se planteará cuáles son las características que se utilizarán para describir a las instancias y posteriormente poder realizar la clusterización. También se realizará un análisis exploratorio de estas características.

8.2.1. Descriptores de las instancias

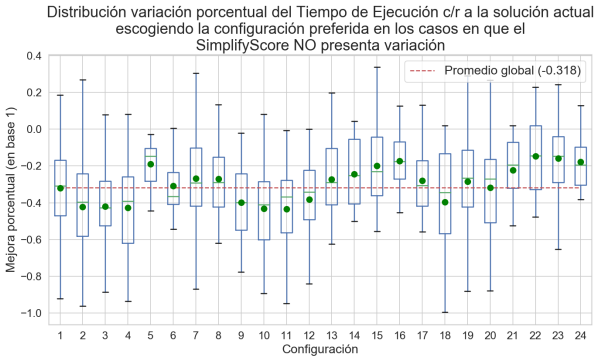
En términos generales, las características que se utilizarán fueron basadas en el trabajo realizado por Rasku et al. (2017), en algunos casos, estas debieron ser adaptadas para el tipo de instancias con las que se está trabajando. También mencionar que las características que se proponen son extraídas exclusivamente del problema de optimización y no se incluyen características propias de los clientes, ya que la heurística de optimización es ciega a las particularidades de los clientes, tales como la industria a la que pertenece el cliente, tipo de



(a) Distribución de la variación porcentual del SimplifyScore con respecto a la situación actual, escogiendo la *configuración preferida*. Caso en que el SimplifyScore presenta variación.



(b) Distribución de la variación porcentual del tiempo de ejecución con respecto a la situación actual, escogiendo la *configuración preferida*. Caso en que el SimplifyScore presenta variación.



(c) Distribución de la variación porcentual del tiempo de ejecución con respecto a la situación actual, escogiendo la *configuración preferida*. Caso en que el SimplifyScore NO presenta variación.

Figura 8.4: Resultados de seleccionar la *configuración preferida* separando cuando el **SimplifyScore** varía y cuando no.

carga que transporte, entre otras.

Por lo general, las características de las instancias son obtenidas a partir de las coordenadas 2D de los nodos o a partir de la matriz de costo del problema, la cual usualmente contiene la distancia entre los nodos (Rasku et al. 2017). A diferencia de la generalidad, en este trabajo se considerarán ambos datos de entrada para realizar la caracterización. En este caso, la matriz de costo hace referencia al tiempo requerido para ir de un nodo a otro, los cuales son obtenidos de la plataforma *OSRM*[15].

Dado que las instancias representan problemas reales y son generadas por clientes en distintas partes del mundo, el espacio geográfico que abarcan puede ser muy distinto entre las instancias. Entonces, para poder hacerlas comparables entre sí, se llevará a cabo un escalado de sus coordenadas geográficas.

El escalado consiste en realizar una proyección de las coordenadas geográficas de los nodos a una grilla de 1x1, es decir, proyectarlas en cuadrado [(0,0), (1,1)]. Para realizar la proyección, es necesario utilizar una escala común entre los ejes X e Y, de esta manera no perder la relación existente entre los puntos, por lo que la escala es calculada como muestra la ecuación 8.5. Luego, para obtener las nuevas coordenada se utiliza la transformación mostrada en la ecuación 8.6. A diferencia de lo realizado por Rasku et al. (2017), la matriz de distancia no requiere ser escalada, ya que los costos representan el tiempo requerido para ir de un nodo a otro, por lo que la transformación no altera el significado de la matriz de costo.

$$scale = \text{mín}\left\{(X_{max} - X_{min}), (Y_{max} - Y_{min})\right\} \tag{8.5}$$

$$(x_{escalado}, y_{escalado}) = \left(\frac{x - X_{min}}{scale}, \frac{y - Y_{min}}{scale}\right) \tag{8.6}$$

Las características que se utilizarán para describir a las instancias se pueden agrupar en cuatro familias de características, las cuales son **distribución de los nodos**, **minimum spanning tree**, **características geométricas** y **características intrínsecas del VRP**. En general, resulta necesario utilizar descriptores estadísticos para calcular las características, debido a que estas no necesariamente tienen un único valor. Para aquellos casos, se utilizan los siguientes descriptores; los cuatro momentos estadísticos (promedio, desviación estándar, *skewness*[16] y la curtosis), el coeficiente de variación, el mínimo, el máximo y la mediana. Además, se determinará el tiempo que tarda en llevar a cabo el cálculo de las características, para así evaluar cuánto tiempo de ejecución se agrega al proceso de optimización.

La elección de estos estadísticos se debe a que permiten describir la distribución de los datos de una forma bastante completa. El promedio entrega una idea del valor sobre el que están centrados los datos, mientras que la desviación estándar habla de la dispersión de estos. Se incluyó el tercer momento estadístico, ya que este permite describir hacia donde

[15]OSRM: Open Source Routing Map. Esta es una plataforma que entrega cuál es la mejor ruta a realizar para ir entre dos puntos, similar a *Google Maps*. Además, OSRM entrega el tiempo de viaje entre los nodos.
[16]La traducción al español es Asimetría Estadística o Sesgo.

están distribuidos los datos en relación al promedio, el signo hace referencia al sentido hacia donde está cargada la distribución y su magnitud que tan cargada está. La curtosis permite describir la forma de la distribución de los datos, es decir, que tan propensa es la distribución a tener *outliers*. Para obtener la curtosis, en este trabajo se utilizará la definición de Fischer, esto quiere decir que, cuando la distribución es igual a una distribución normal, el valor de la curtosis es 0. El coeficiente de variación ayuda a describir el grado de variabilidad de los datos, el cual no depende de la magnitud ni de la escala de los datos, se calcula como $\frac{\sigma}{\bar{\mu}}$. Finalmente, el mínimo, el máximo y la media permite conocer el rango en el que se mueven los datos.

A continuación se describirán las características escogidas en cada una de las familias antes mencionadas. Además, en la tabla D.1 del anexo D se encuentran resumidas todas las características aquí descritas.

Distribución de los nodos

El objetivo de esta familia de características es describir la distribución de los puntos en el plano. Importante recordar que en este caso la matriz de costo hace referencia al tiempo de viaje requerido entre los nodos. Recordar también que las coordenadas de los nodos fueron escaladas en el cuadrado $[(0,0),(1,1)]$.

Las características extraídas fueron las siguientes:

(i) Valores de la matriz de costo: Comúnmente se estudia esta matriz para conocer las *distancias* que existen entre los nodos. Sin embargo, en este caso la matriz de costo corresponde a la matriz de tiempos de viaje. Destacar que la matriz de costos no es simétrica y que no se considerarán los valores de la diagonal. De estos valores se calculan 7 descriptores ya mencionados, ya que el mínimo usualmente es cero, dada la forma en que se construye la matriz[17].

(ii) Cantidad de elementos distintos en la matriz de costo: En Hutter et al. (2015) se propone utilizar esta variable, ya que permite describir cierta regularidad entre los nodos. En este caso no se utiliza sensibilidad, ya que la unidad de los valores es segundos, por lo que una fracción de estos no es relevante para el viaje.

(iii) Centroide de los puntos: También se calcula el centroide de todos los nodos, generando dos variables, X e Y.

(iv) Distancia Euclidiana de los nodos al centroide: Se calculan los 8 estadísticos para las distancias entre los nodos y el centroide del problema. Esto ayuda a describir la dispersión de los puntos.

(v) Coeficiente de correlación: Para describir la ubicación de los puntos en el espacio se calcula el coeficiente de correlación entre las coordenadas X e Y del plano.

[17]El mínimo usualmente es cero debido a el número de *depots* que tiene la matriz de distancia es igual al número de vehículos que tiene la instancia, aunque el problema tenga sólo un *depot*, lo que genera que el tiempo que tarda en viajar del mismo *depot* al mismo *depot* es cero.

En la sección 6.2 se menciona la realización de cluster de clientes, por lo que también puede ser importante describir cómo es posible agrupar a los nodos en el espacio. Para realizar lo anterior se utiliza un método de clusterización de nodos conocido como DBSCAN, se escoge este método debido a que es bastante utilizado en trabajos similares (Hutter et al. 2015, Steinhaus 2015, Smith-Miles & Van Hemert 2011) y permite clusterizar siguiendo un criterio de cercanía de los nodos, sin obligar a que todos los nodos pertenezcan a un cluster. Esto es beneficioso porque permite identificar aquellos nodos que se encuentran *lejos* de otros nodos. El método DBSCAN requiere que se le indique la distancia máxima permitida para que un nodo pertenezca a un cluster, a este parámetro se le conoce como *eps*. Para determinar el *eps* se seguirá lo propuesto por Steinhaus (2015, pg. 70-71), donde se busca incorporar a los cuatro vecinos más cercanos, para lo cual, el *eps* se calcula como $\frac{b}{\sqrt{N}}$, donde b en este caso es igual al valor obtenido en la ecuación 8.5 y N el número de puntos.

A partir de la clusterización de los nodos anterior, se extraen las siguientes características.

(vi) Número de clusters: Identifica cuántos clusters se generaron.

(vii) Número de nodos en cluster: Identifica la cantidad de puntos que tiene asignado un cluster.

(viii) Número de outliers: Identifica la cantidad de puntos no asignados a un cluster.

(ix) Proporción de puntos en cluster: Identifica el porcentaje de nodos asignados a un cluster.

(x) Alcance de los cluster: Se obtienen los 8 estadísticos para describir el alcance que tienen los cluster. Con alcance del cluster se hace referencia al número de puntos que tiene cada cluster.

(xi) Coeficiente de Silhouette: Este coeficiente permite evaluar la consistencia de los cluster obtenidos, evaluando la *cohesión* y *separación* de los cluster. El coeficiente toma valores entre -1 y 1, donde valores altos indican que los nodos están correctamente asignados al cluster que les corresponde.

Minimum Spanning Tree (MST)

El objetivo de este set de características es describir la conexión que existe entre los nodos. Para eso se utiliza el MST, el cual es una metodología que permite generar un grafo completamente conectado entre sí, con la característica que los arcos que conectan los nodos son los mínimos requeridos para lograr la conexión. En este caso, los arcos representan el tiempo requerido para ir de un nodo a otro.

A partir de este árbol, se obtienen las siguientes características:

(i) Costo de los arcos: Se obtienen 7 de los 8 estadísticos descritos para los arcos del árbol. En este caso se excluye el mínimo debido a la misma razón mencionada para la variable *Valores de la matriz de costo*.

(ii) Grado de profundidad los nodos: Por grado de profundidad de un nodo se entiende como la cantidad antecesores directos que tiene el nodo. Esta característica permite describir la cantidad de pasos *mínimos* necesarios para llegar a un determinado nodo. De los 8 descriptores mencionados al comienzo de esta sección, en este caso se excluye el cálculo del mínimo grado de profundidad, ya que su valor siempre será cero.

Características geométricas

El objetivo de esta familia de características es poder describir la forma geométrica del problema. La forma del problema puede hacer que un problema sea más complejo de resolver que otro, como lo muestra Pihera & Musliu (2014, pg. 50). Lo anterior se logra capturar en base a características derivadas de la envoltura convexa (EC). Cabe mencionar que para el caso de un conjunto P de puntos en un plano, su envoltura convexa corresponde al mínimo polígono que los envuelve a todos.

Las características propuestas son las siguientes:

(i) Área de la envoltura convexa: Permite describir que tanto espacio está utilizando la distribución de puntos. Mientras mayor sea este valor, se interpreta como una mayor dispersión de puntos.

El área de la envoltura convexa en este caso corresponde al área del polígono que encierra a todos los nodos de la instancia de optimización.

(ii) Ratio de Puntos en el área sobre el Total de puntos: Permite conocer la proporción de puntos que se encuentran en las aristas de la envoltura convexa. Este ratio se calcula como:

$$ratio = \frac{\text{cant. de nodos dentro de la EV}}{\text{total de nodos}}$$

(iii) Distancia de los nodos dentro de la envoltura convexa a la envoltura: Se calculan los 8 estadísticos ya descritos. Permite describir que tan lejos están los nodos de su envoltura.

La distancia de un nodo a su envoltura convexa se define como la mínima de las distancias entre el nodo y las aristas de la envoltura, tal como lo muestra la ecuación 8.7, donde $dist_{n,CH}$ es la distancia del nodo n a la envoltura convexa y A el conjunto de las aristas de envoltura convexa.

$$dist_{n,CH} = \min_{a_i \in A} \{dist_{n,a_i}\} \tag{8.7}$$

Luego, la distancia que existe entre un nodo n y la arista $a_i \in A$ corresponde a la altura del triángulo formado por n y los extremos de a_i. Para encontrar dicha altura basta

con despejar de la fórmula clásica para calcular el área de un triángulo (eq. 8.8) y la fórmula Herón (eq. 8.9).

$$A_{\text{triangulo}} = \frac{base * altura}{2} \quad , \; con \quad base = a_{\text{i}} \wedge altura = \text{dist}_{n,a_{\text{i}}} \tag{8.8}$$

$$A_{\text{triangulo}} = \sqrt{s * (s-a) * (s-b) * (s-c)} \quad , \; con \quad s = \frac{a+b+c}{2} \tag{8.9}$$

(iv) Largo aristas envoltura convexa: Se calculan los 8 estadísticos ya descritos. En conjunto a las características anteriores, permite dilucidar cómo es la distribución de los puntos dentro de la envoltura convexa.

Características intrínsecas de VRP

El objetivo de estas características es recopilar información propia del problema de ruteo. Para más detalle de las características propuestas a continuación, referirse a Steinhaus (2015).

(i) Centroide de los puntos SIN escalar: Permite ubicar el problema en un lugar del planeta, se generan dos variables, X e Y, con coordenadas *sexagesimales*.

(ii) Número de clientes.

(iii) Número de depots.

(iv) Número de vehículos

(v) Ratio Clientes sobre Vehículos.

(vi) Centroide de los clientes.

(vii) Centroide del depot.

(viii) Distancia Euclidiana entre el centroide de los clientes y de los depots.

(ix) Distancia Euclidiana entre los clientes y el centroide de los depots: Se obtienen los 8 estadísticos ya descritos.

(x) Demanda de los clientes: Se obtienen los 8 estadísticos ya descritos.

(xi) Demanda total de los clientes.

(xii) Capacidad de los vehículos: Se obtienen los 8 estadísticos ya descritos.

(xiii) Capacidad total de los vehículos.

(xiv) Ratio Demanda Total sobre Capacidad Total.

(xv) Costo por uso de vehículo: Los clientes pueden indicar el costo por utilizar los vehículos. Este costo tiene las unidades que el cliente defina. Para esta característica se obtuvieron los 8 estadísticos ya descritos.

(xvi) Si el problema tiene demanda: Esta característica toma el valor de 1 cuando la demanda total es mayor a cero, sino toma el valor 0.

(xvii) Si el problema tiene capacidad: Esta característica toma el valor de 1 cuando la capacidad total es mayor a cero, sino toma el valor 0.

8.2.2. Análisis características de las instancias

Como primer análisis se debe estudiar el tiempo que lleva calcular las características en cada instancia para evaluar el impacto que tiene este tiempo de ejecución en el total del proceso de optimización. Recordar que el tiempo es un recurso limitado y el ideal es mantenerlo bajo.

Tras calcular las características en el set de entrenamiento, se obtuvo que en la generalidad de las instancias, el tiempo de computo de las características es bajo. Como se puede apreciar en la gráfica 8.5, en el 95 % de los casos, el rango del tiempo que lleva calcular las características se encuentra entre $[0.01, 0.49]$ segundos, con la mediana en 0.07 segundos y el promedio en 0.31 segundos. Estos valores muestran que el tiempo de cómputo de las características es prácticamente despreciable y no agrega significativamente tiempo al proceso de optimización.

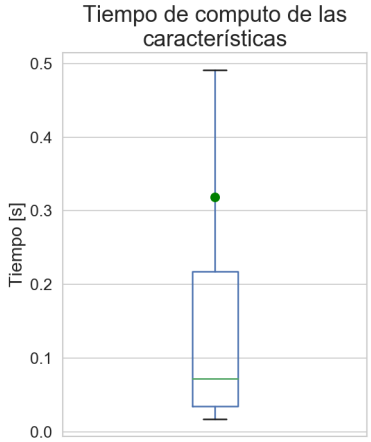


Figura 8.5: Distribución del tiempo de cómputo de las características.

En un análisis preliminar se determinó que para algunas variables era necesario realizar una transformación logarítmica, la razón de realizar esto es que poseen valores muy extremos entre sí. A las variables que se les realizó la transformación logarítmica fueron las que provenían de la matriz de costo, las cuales son; el promedio, la desviación estándar, el máximo y la mediana de los valores de la matriz. Otras variables a las que se le aplicó la transformación fueron; el promedio, la desviación estándar, el máximo y la mediana de los valores del costo de los arcos obtenidos por el MST.

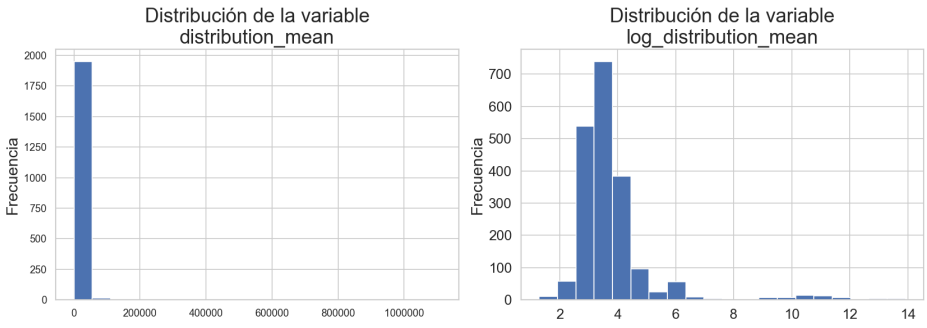


Figura 8.6: Ejemplo de la transformación logarítmica de la variable que representa el promedio de los valores de la matriz de costo. A la izquierda se encuentra el histograma de los valores sin transformar y a la derecha el histograma de los valores con la transformación logarítmica.

En la gráfica 8.6 se puede apreciar un ejemplo de que estas transformaciones resulta útiles. En el anexo F se encuentra el resto de las figuras en donde se realizó dicha transformación.

Además de la transformación anterior, como los valores de las características pueden llegar a ser muy distintas entre sí es necesario escalarlas, para poder así hacerlas comparables. La principal razón de que sean comparables es que los modelos que se propondrán más adelante utilizan métricas de distancia para entrenarse. El método de escalado utilizado fue el de *MinMaxScaler*, ya que este asegura que todas las variables serán comparables entre sí. Otra opción era escoger el método de estandarización de las variables, es decir, que tuvieran media 1 y desviación estándar 0, sin embargo, esta metodología no asegura que las variables serán comparables entre sí, los valores de estas pueden seguir siendo muy diferentes entre sí.

Dado que se calcularon bastantes características, 112 en total, se evaluará si existe correlación entre ellas. Es necesario evitar incluir variables que tengan una alta correlación (tanto negativa como positiva), ya que tener variables con una alta colinealidad genera ruido en los modelos lo que empeora el ajuste de los mismos.

Como se puede apreciar en la matriz de correlación de variables (gráfico E.1 del Anexo E), existen muchas variables que están fuertemente relacionadas entre sí. Tras este análisis se pretende remover aquellas correlaciones que no estén en el intervalo $[-0.7, 0.7]$. A continuación se explicarán algunos de los casos con alta correlación, el resto no serán detalladas pero si serán consideradas para su remoción. Tras la remoción de las variables con una alta correlación, quedaron 53 características, las cuales se encuentran resumidas en la tabla D.2 del Anexo D. La matriz de correlación final se encuentra resumida en la gráfica E.2 del Anexo E.

Un caso de alta correlación se encuentran que la variable **log distribution max**[18] tiene una alta correlación con las variables **log distribution mean**[19] y **log distribution std**[20]. Esta relación se explica, ya que en los casos en que el máximo es muy grande, el promedio y desviación también tienden a serlo. Por esta razón, se prescindirá de la variable **log distribution max**.

[18]Esta variable representa el máximo valor de la matriz de costo.
[19]Esta variable representa el promedio de los valores de la matriz de costo.
[20]Esta variable representa la desviación estándar de los valores de la matriz de costo.

Otro caso en que existe una alta correlación la variable **DBcentroideAndDepot**[21] y el par de variables **clientDistToDepot mean**[22] y **clientDistToDepot median**[23]. La relación entre estas variables se explica debido a que si la distancia entre el centroide de los clientes y la ubicación de *depot* es alta, entonces resulta lógico pensar que tanto el promedio como la mediana de la distancia promedio entre los clientes y el *depot* también sea alta. Se dejará en la base la variable **DBcentroideAndDepot**.

Como ya se mencionó, tras la remoción de las variables con una alta colinealidad se redujo el set de características a 53 variables descriptivas.

Ahora bien, ya habiendo removido las variables que tienen una alta correlación se procederá a analizar las distribuciones de algunas de las variables restantes. Las primeras variables que se analizarán corresponden a la demanda de los clientes y la capacidad de los vehículos. En particular se analizará el ratio de demanda sobre capacidad, debido a que la demanda y la capacidad de los vehículos puede tener un significado distinto para cada instancia, ya que es el cliente quien le entrega la unidad a dicho campo, por lo que las demandas y capacidades no son necesariamente comparables entre instancias.

Para el siguiente análisis se analizará principalmente la mediana de los datos, ya que esta representa donde se encuentra el 50 % de los datos y resulta ser un buen indicador para generar reglas de agrupación.

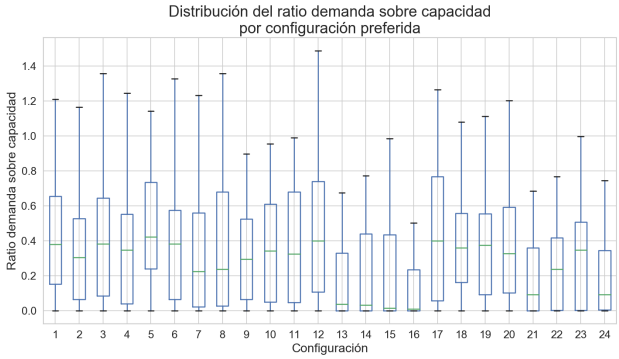
Si se observa la gráfica 8.7a, se apreciar cierta relación entre este ratio demanda sobre capacidad y la configuración óptima. Por un lado es factible identifica que aquellas instancias que tienen un ratio bajo, menor a 0.2, tienen a ser resueltas mejor por las configuraciones c13, c14, c15, c16, c21 y c24. El común denominador entre estas diferentes configuraciones, es que todas utilizan el criterio de agrupación de demanda.

Otra característica que permite generar un grupo de configuraciones preferidas es el ratio de puntos dentro de la envoltura convexa. A partir de la gráfica 8.7b se aprecia que existe un grupo de características que son preferidas cuando el ratio de puntos dentro de la envoltura convexa es menor al 92 %, en este grupo se identifican las configuraciones c17, c18, c19 y c21. El común denominador entre estas diferentes configuraciones, es que todas utilizan el método de agrupación de clientes *ByPaper*.

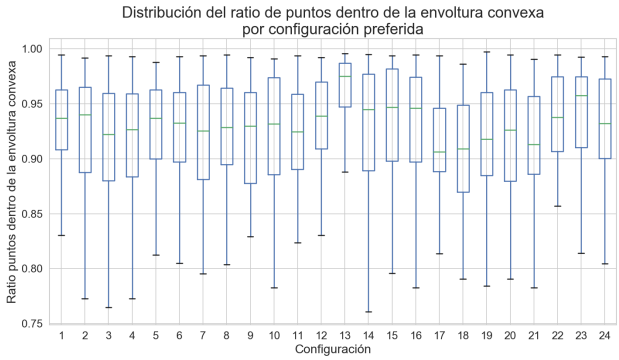
Una última característica analizada que permite generar un grupo de configuraciones preferidas es el ratio de clientes por vehículo. En la gráfica 8.7c se aprecia que se pueden agrupar las configuraciones c13, c15, c16 y c22 cuando el ratio indica que hay más de 15 clientes por vehículos.

Finalmente, en base a las características mostradas es posible identificar grupos de características en base a ciertos atributos. Con este resultado se plantea la hipótesis de que es posible agrupar las configuraciones en base a ciertas características, lo que permite seguir desarrollando la solución propuesta en el capítulo 2.

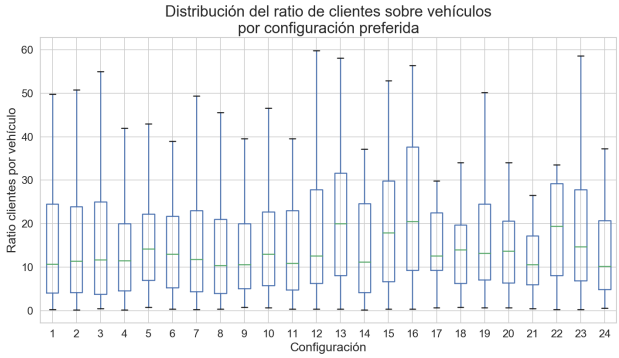
[21] Esta variable representa la distancia euclidiana entre el centroide de los clientes y de los depots.
[22] Esta variable representa el promedio de la distancia entre los clientes y el centroide de los depots.
[23] Esta variable representa la mediana de la distancia entre los clientes y el centroide de los depots.



(a) Distribución del ratio demanda sobre capacidad por configuración preferida.



(b) Distribución del ratio de puntos dentro de la envoltura convexa por configuración preferida.



(c) Distribución del ratio de clientes sobre vehículos por configuración preferida.

Figura 8.7: Gráficas análisis de características.

Capítulo 9

Modelos de clusterización

En este capítulo se plantearán y evaluarán tres modelos para determinar cuál es la configuración ideal de la heurística de optimización para cada una de las instancias. Recordar que según la solución propuesta en 2.4.2 lo que se pretende realizar es una clusterización de las instancias y para cada uno de estos clusters determinar cuál es la configuración ideal.

Resulta importante recordar que el desafío de resolver la problemática abordada en este trabajo de memoria es determinar cuál es la configuración ideal para cada instancia previo a ejecutar la heurística de optimización. Esto quiere decir, lógicamente, que previo a ejecutar la heurística de optimización, no se tiene conocimiento alguno de cómo se comportará este. Por esta razón, lo ideal es plantear modelos que sólo consideren las características de las instancias y no información obtenida *post* ejecución de la heurística de optimización.

Los modelos propuestos son los siguientes: *K-means*, *Gaussian Mixture* y *k-NN*. Los primeros dos corresponde a modelos de clusterización y el tercero se plantea directamente como un modelo de predicción.

Antes de comenzar los modelos, es importante recordar con qué elementos se cuenta para desarrollarlos. Los cuatro elementos principales para realizar los modelos son:

- **Espacio de problemas ($\mathcal{P}$):** Representado por la base de instancias calculada en el capítulo 7, en particular la base de entrenamiento, la cual posee 1979 instancias.
- **Espacio de características ($\mathcal{F}$):** Las cuales corresponde a las características propuestas en el capítulo 8. Se cuenta con 53 características distintas (tabla D.2 del Anexo D).
- **Portafolio de algoritmos ($\mathcal{A}$):** El portafolio corresponde a las 24 distintas configuraciones de la heurística de optimización, propuestas en el capítulo 6.
- **Desempeño de los algoritmo ($\mathcal{Y}$):** Corresponde a las soluciones encontradas tras la ejecución de la base de instancias $\mathcal{P}$ en cada uno de las heurísticas del portafolio $\mathcal{A}$.

9.1. Modelo de *K-means*

Como se mostró en el capítulo 5, para llevar a cabo la clusterización de los puntos, el modelo de *k-means* lo que hace es definir los k centros y asignar los puntos a alguno de los clusters, representados por estos k centros. El objetivo de *k-means* es realizar una asignación tal que minimiza la distancia que existe entre los puntos de un mismo cluster. A esta función objetivo se le conoce como *within-cluster sum-of-squares* (WCSS). Esta métrica permite conocer qué tan buena es la clusterización. Mientras menor sea, mejores son los clusters.

Este método de clusterización requiere que se le indique cuántos clusters crear. Como se mencionó en la sección 5.3.1, no existe una regla estricta que indique cuál es el número de clusters k a realizar, por lo que se debe tomar una decisión arbitraria. Para tomar esta decisión se estudiarán las métricas WCSS y el coeficiente de Silhouette. El análisis de estas métricas consiste en graficar los valores que toman estas métricas para los distintos valores de k en el modelo y construir las gráficas 9.1a y 9.1b y en base a estas decidir el mejor valor de k.

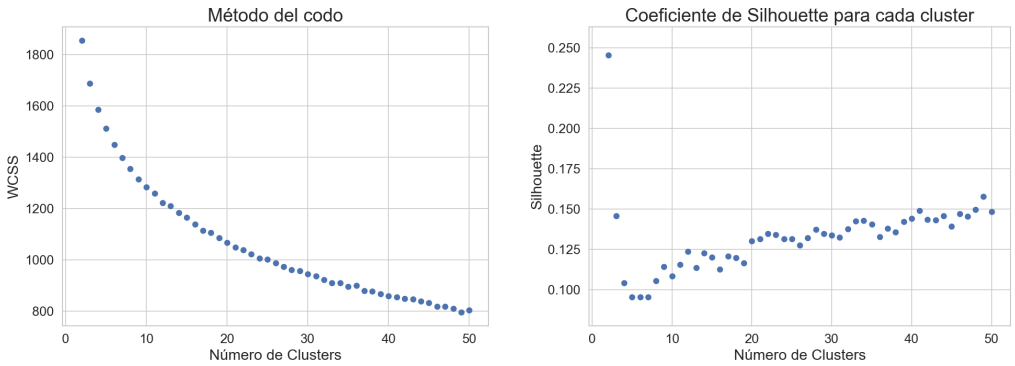
Cuando se estudia la métrica WCSS se desea que esta sea la menor posible, lo que se logra cuando $k = N$, con N igual al número de instancias en la base. Sin embargo definir k igual al tamaño de la base carece de sentido y de capacidad predictiva. Entonces, para escoger el k indicado se utilizará el *método del codo*. Este método permite gráficamente escoger un k tal que agregar un cluster más al modelo no agrega significativamente más información, es decir, no disminuye significativamente el WCSS. Por otro lado, cuando se estudia el coeficiente de Silhouette lo que se busca es seleccionar el k que tenga el coeficiente más alto. Importante destacar que estos métodos de análisis se analizan en conjunto, por lo que ninguna por si sola es determinante.

Entonces, realizando la clusterización a partir del set de características $\mathcal{F}$ se procedió a analizar el WCSS y el coeficiente de Silhoutte para los valores de k en un rango de valores entre 2 y 50, obteniendo las gráficas 9.1a y 9.1b. De la gráfica 9.1a no resulta claro cuál es el k ideal, pero se logra apreciar que se encuentra en el intervalo entre $[5, 14]$, ya que entre $k = 14$ y $k = 20$ la disminución del WCSS es pequeña, aproximadamente de 100 unidades. Mientras que la disminución del WCSS que existe entre $k = 5$ y $k = 14$ es de aproximadamente 330 unidades.

Ahora bien, si se analiza la gráfica 9.1b, se aprecia que el $k \in [5, 14]$ que tiene el mayor coeficiente de Silhouette es $k = 12$, por lo que se escoge dicho valor. Es cierto que para valores de $k > 20$ el coeficiente de Silhouette es más alto, sin embargo esos casos no se consideran relevantes ya que el WCSS no disminuye sustancialmente.

Una vez que las instancias se encuentran clusterizadas, para determinar cuál es la configuración ideal para ejecutar las instancias de dicho cluster, se escogerá aquella configuración que en promedio tenga el mejor desempeño para las instancias del cluster. Realizar esta elección asegura que nunca se escoja una configuración que no termina y además, que en el promedio siempre va a ser la mejor.

Una de las hipótesis de este trabajo es que mediante la caracterización de las instancias,



(a) Valor del WCSS para cada cluster.

(b) Coeficiente de Silhouette para cada cluster.

Figura 9.1: Gráficas para escoger el k ideal en el modelo de clusterización *k-means* utilizando las características en $\mathcal{F}$.

sería posible crear clusters de instancias que funcionaran bien con ciertas configuraciones, de manera tal de determinar que para las instancias de dicho cluster se logra un mejor desempeño con cierta configuración. Ahora bien, analizando cómo se agrupan los clusters en las 24 configuraciones, gráfica 9.2, se puede apreciar que este modelo no permite ratificar la hipótesis propuesta, ya que los clusters se encuentran repartidos en todas las configuraciones, no pudiéndose apreciar ninguna tendencia ni ninguna agrupación.

A partir de lo observado en la gráfica 9.2, el resultado de esta clusterización resulta no ser buena, ya que al estar los clusters repartidos entre todas las configuraciones no se logra definir cuál es la *configuración preferida* para los clusters, de forma tal que permita capturar los casos en que se tiene un mayor beneficio.

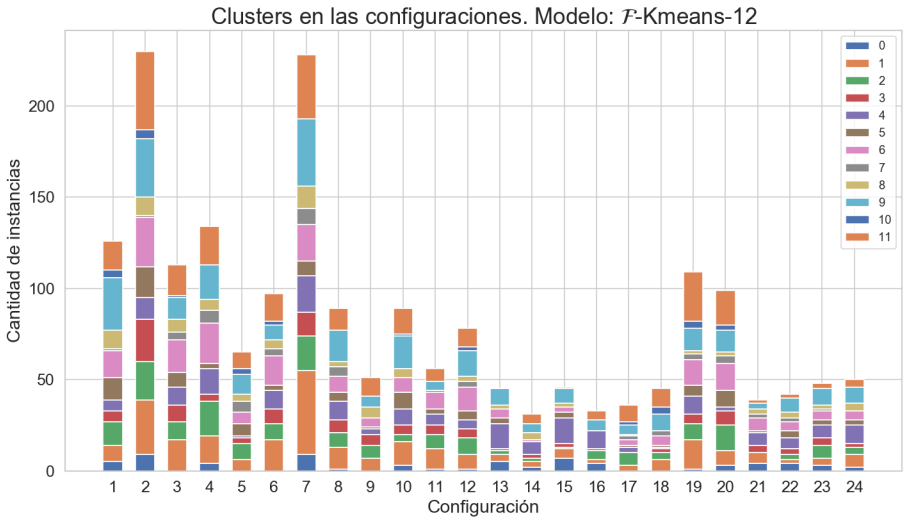


Figura 9.2: Distribución de los clusters por configuración, usando *k-means* utilizando las características en $\mathcal{F}$.

Para este modelo, el desempeño promedio obtenido empeora la solución de la situación actual en un 0.93 %. En términos del tiempo de ejecución, se logra una mejoría del 5.95 %.

Además, si se analiza la proporción de los casos en que esta forma de escoger la *configuración preferida* mejora o empeora la situación actual, se aprecia que en términos generales, en un 30.04 % de los casos las soluciones mejora, mientras que en un 34.46 % de los casos las soluciones empeoran. En la matriz de confusión (tabla 9.1) se aprecia la distribución de los casos.

		Tiempo de ejecución			
		Mejoran	*Igual*	*Empeoran*	*Total*
SimplifyScore	*Mejoran*	91	86	59	**236**
	Igual	367	694	229	**1290**
	Empeoran	171	209	73	**453**

Tabla 9.1: Matriz de confusión de la cantidad de casos que mejora y empeoran su solución al escoger la *configuración preferida* mediante la clusterización realizada con *k-means*, con respecto a la situación actual. Se analizan las métricas **SimplifyScore** y el **tiempo de ejecución**.

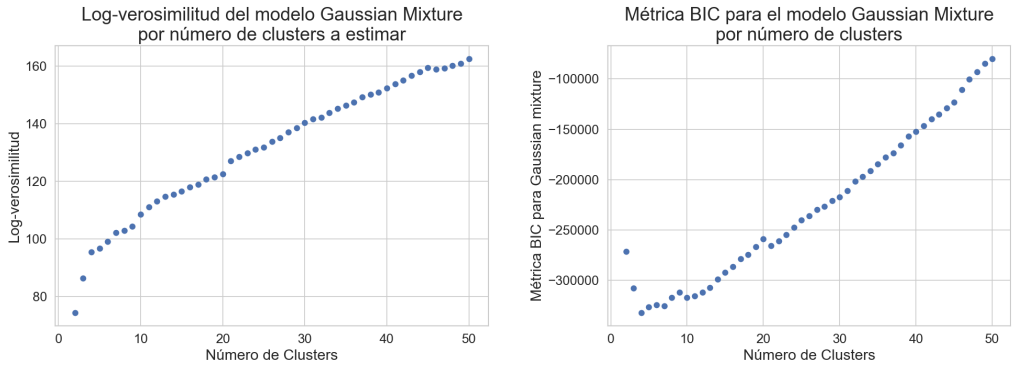
Ahora bien, analizando el desempeño obtenido dentro de cada uno de los clusters se puede apreciar que a ningún cluster encontrado le sustantivamente mejor (ver tabla 9.2). Sin embargo, si se observa al cluster 10, se aprecia que existe un nicho pequeño de instancias bien clasificadas. A una proporción importante del cluster le va bastante bien en términos de su tiempo de ejecución, ya que lo disminuye en un 40.12 % sin empeorar sustantivamente su SimplifyScore.

Label	**Size**	**Var. SS (%)**	**Var. ET (%)**	**Mejoran (%)**	**Empeoran (%)**
0	68	0.025	-1.84	25.0	29.41
1	257	0.583	-5.72	21.79	16.73
2	194	1.034	-11.78	36.6	32.47
3	128	1.137	3.17	21.09	53.91
4	197	0.942	7.36	11.17	34.01
5	115	1.278	-8.14	33.04	29.57
6	236	1.483	-9.73	39.83	35.59
7	58	1.795	-6.39	32.76	31.03
8	96	0.474	1.01	33.33	48.96
9	303	0.964	-1.25	41.25	50.5
10	31	-0.007	-40.20	80.65	3.23
11	296	0.758	-15.62	26.01	28.04

Tabla 9.2: Desempeño obtenido por cluster utilizando el método *k-means* utilizando las características en $\mathcal{F}$, con $k = 12$. Comparación realizada contra la situación actual.

9.2. Modelo *Gaussian Mixture*

Como siguiente modelo a evaluar, se propone utilizar un modelo de clusterización que permita generar buenos clusters en situaciones en que estos son más difusos, en este caso se



(a) *Log-verosimilitud* del modelo para cada número de clusters.

(b) Métrica BIC para cada número de clusters.

Figura 9.3: Gráficas para escoger el número de clusters ideal en el modelo *Gaussian Mixture* utilizando las características en $\mathcal{F}$.

escogió el modelo de *Gaussian Mixture*. La razón de escoger este método es que en general logra identificar clusters incluso cuando la separación entre estos es difusa, como se mostró en el capítulo 5 (sec. 5.3.2).

Similar al modelo anterior, este tipo de clusterización requiere que se le indique el número de componentes a clusterizar. Para escoger cuál es el número de componentes ideal se estudiarán dos métricas asociadas a este método de clusterización. La primera es la función objetivo del método de clusterización y la segunda es el coeficiente BIC. En este caso se graficaron dichas métricas para cada número de componentes, los cuales están en un rango de valores entre 2 y 50, obteniendo las gráficas 9.3a y 9.3b.

Cuando se estudia la *log-verosimilitud* del modelo, se desea que este sea lo más grande posible, ya que de esta manera la probabilidad de que los parámetros estimados repliquen la data es mayor. Sin embargo, si se aumenta el número de componentes a estimar, siempre va a aumentar dicha probabilidad. Entonces, analizando la gráfica 9.3a se aprecia el número ideal de componentes puede ser 4, 5 o 6, ya que para valores mayores, la *log-verosimilitud* no aumenta drásticamente, esto quiere decir que ajustar el modelo a más distribuciones mejora el modelo, pero no de forma relevante.

Como ya se mencionó, aumentar el número de componentes que se estima, siempre va a mejorar la *verosimilitud* de los datos del modelo, pero lo que se está haciendo al aumentar el número de componentes es sobre-ajustar el modelo. Para evitar este sobre-ajuste, resulta útil estudiar la métrica BIC[24]. Esta métrica penaliza la mejora de la *log-verosimilitud* del modelo por el número de componentes que están siendo estimados, permitiendo comparar los modelos. Se dice que aquel modelo que tiene un menor BIC es mejor modelo.

Analizando la gráfica 9.3b, se aprecia que el número de componentes ideal es 4, ya que cuando el número de componentes es mayor a 4, la métrica BIC es menor.

Se procedió a realizar la clusterización utilizando este método con 4 componentes. Al

[24]BIC: Bayesian Information Criterion.

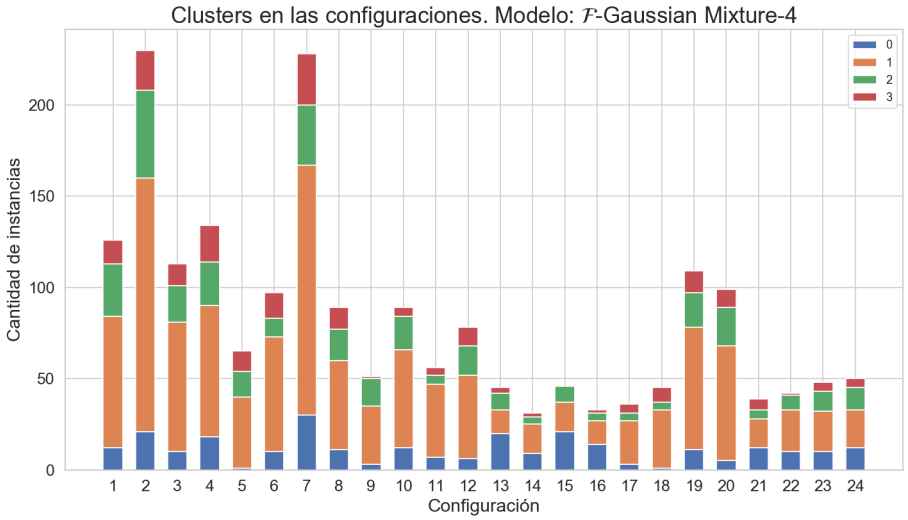


Figura 9.4: Distribución de los clusters por configuración, usando el modelo *Gaussian Mixture* utilizando las características en $\mathcal{F}$.

igual que el caso anterior, se definirá que la mejor configuración de cada cluster será la que en promedio funciona mejor.

Ahora bien, analizando cómo se distribuyen los clusters en las diferentes configuraciones (gráfica 9.4), nuevamente se aprecia que los clusters obtenidos se encuentran repartidos entre las configuraciones. Esto provoca que la *configuración preferida* para cada cluster no sea la que mejor desempeño genera en las instancias del cluster.

Como es de esperar, el desempeño promedio que tienen las instancias al ejecutar la heurística de optimización con la configuración especificada para cada cluster genera que las soluciones empeoren en promedio un 1.2 %. Sin embargo, el tiempo de ejecución logra mejorar un 12 %. Además, si se analiza la proporción de los casos en que esta forma de escoger la *configuración preferida* mejora o empeora la situación actual, se aprecia que en términos generales, en un 30.52 % de los casos las soluciones mejora, mientras que en un 34.46 % de los casos las soluciones empeoran. En la matriz de confusión (tabla 9.3) se aprecia la distribución de los casos.

		Tiempo de ejecución			
		Mejoran	*Igual*	*Empeoran*	*Total*
SimplifyScore	*Mejoran*	53	75	18	**146**
	Igual	513	716	215	**1444**
	Empeoran	171	181	37	**389**

Tabla 9.3: Matriz de confusión de la cantidad de casos que mejora y empeoran su solución al escoger la *configuración preferida* mediante la clusterización realizada con *Gaussian Mixture* con respecto a la situación actual. Se analizan las métricas **SimplifyScore** y el **tiempo de ejecución**.

Analizando el desempeño en particular de cada uno de los clusters, tabla 9.4, se aprecia que en ninguno de los clusters se obtiene un mejor desempeño que en la situación actual.

Label	Size	Var. SS (%)	Var. ET (%)	Mejoran (%)	Empeoran (%)
0	269	0.73	2.98	13.75	30.86
1	1140	1.25	-15.55	33.68	29.21
2	359	1.67	-11.55	39.83	34.54
3	211	1.29	-13.02	45.02	30.33

Tabla 9.4: Desempeño obtenido por cluster utilizando el método *Gaussian Mixture* utilizando las características en $\mathcal{F}$, estimando para 4 componentes. Comparación realizada contra la situación actual.

9.3. Modelo *k-NN*

Dada la poca eficacia que han mostrado los métodos de clusterización anteriores, a continuación se propone utilizar el método conocido como *k-Nearest Neighbot* (*k-NN* por sus siglas en inglés). La razón de utilizar este modelo es que corresponde a un modelo supervisado, de esta manera se puede entrenar el modelo teniendo en consideración cuál es la mejor configuración para cada instancia.

Como se mencionó en la sección 5.3.3 (cap. 5), la particularidad de este método es que es una metodología que intenta realizar predicciones en base a la información de los k vecinos más cercanos. En este caso en particular, en vez de definir a qué cluster pertenece cada instancia y definir cuál es la configuración ideal para ese cluster, lo que se pretende realizar es predecir que la mejor configuración para una determinada instancia X es la misma que la de los k vecinos más cercanos a ella.

Como se identificó en el capítulo 8, existen casos en que las instancias no terminan y estos casos son considerados indeseables, por lo que para evitar que en este método se escojan dichas configuraciones, simplemente se descartan las configuraciones mostradas en la figura 8.2a.

Como no existe una regla estricta para escoger cuántos vecinos cercanos considerar, además de probar distintos valores de k, se implementará una metodología similar a los casos anteriores, y así determinar cuál debiese ser el k ideal para el modelo.

En este caso, lo que se realizará es separar la base de entrenamiento en dos, en una nueva base de entrenamiento y en otra de prueba. La base de testeo corresponderá al 30 % de la base de entrenamiento. En esta nueva base de entrenamiento, que ahora posee 1385 registros, se entrenará el modelo de *k-NN* para diferentes valores de k y se realizará la predicción de las etiquetas en la base de testeo. El resultado de este ejercicio se encuentra en la gráfica 9.5. En dicha gráfica se muestra el *accuracy promedio* que se obtiene al modelar para los diferentes valores de k. Se puede apreciar que el valor de k que mejor *accuracy promedio* tiene es para $k = 30$.

De todas formas se evaluará el modelo para los siguientes valores de k: 5, 10, 15, 25 y 43.

En la tabla 9.5 se encuentran resumidos el desempeño promedio que logra la heurística de optimización al utilizar el modelo *k-NN*, con diferentes valores de k, para predecir la mejor

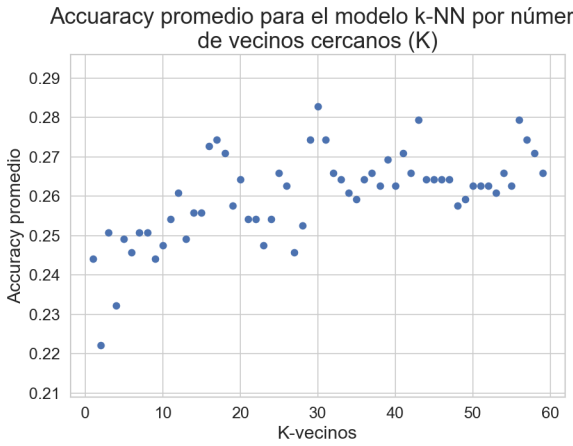


Figura 9.5: *Accuracy promedio* para diferentes valores de k en el modelo *k-NN* utilizando las características en $\mathcal{F}$.

Modelos	**Var. SS (%)**	**Var. ET (%)**	**Mejoran (%)**	**Empeoran (%)**
k-NN-25	2.89	-24.23	35.73	37.9
k-NN-30	2.89	-24.68	35.73	37.65
k-NN-43	2.93	-24.20	35.42	37.95
k-NN-15	3.12	-24.76	35.83	37.59
k-NN-5	3.2	-26.21	36.48	38.55
k-NN-10	3.48	-24.56	35.52	38.45

Tabla 9.5: Resultados de las predicciones realizadas con *k-NN* utilizando las características en $\mathcal{F}$. Comparación realizada contra la situación actual.

configuración de la heurística de optimización.

Como se puede apreciar en la tabla 9.5, el valor de k que entrega un mejor resultado es $k = 25$. Cabe destacar que para este valor de k, la cantidad de instancias que mejoran su situación actual, ya sea en términos del **SimplifyScore** o del **tiempo de ejecución** corresponde al 35.73 %, mientras que la porción que empeora corresponde al 37.9 %. Este modelo no resulta completamente confiable ni mejora la situación actual, ya que en promedio empeora las soluciones entregadas en un 2.89 % en comparación a la situación actual, pero mejora el tiempo de ejecución en un 24.23 %. En la matriz de confusión (tabla 9.6) se aprecia cómo se distribuyen los casos.

9.4. Selección de características

La principal hipótesis que se plantea de por qué ninguno de los modelos anteriores logra tener un buen desempeño es que las características escogidas siguen produciendo ruido en el modelo. Por esta razón, se propone llevar a cabo una nueva selección de las características.

Para llevar a cabo la selección de las características relevantes se usará el método del *chi-*

		Tiempo de ejecución			
		Mejoran	*Igual*	*Empeoran*	*Total*
SimplifyScore	*Mejoran*	120	89	17	**226**
	Igual	483	526	114	**1123**
	Empeoran	328	267	35	**630**

Tabla 9.6: Matriz de confusión de la cantidad de casos que mejora y empeoran su solución al escoger la *configuración preferida* mediante la predicción realizada con *k-NN* con respecto a la situación actual, utilizando las características en $\mathcal{F}$, con $k = 25$. Se analizan las métricas **SimplifyScore** y el **tiempo de ejecución**.

Familia de características	**Características**	**Estadísticos**
Distribución de los nodos	Valores de la matriz de costo	Mediana
	Proporción de puntos en clúster	Valor único
Minimum Spanning Tree	Grado profundidad de los nodos	Skew
Características geométricas	Distancia de los nodos dentro de la envoltura convexa	Skew
	Largo aristas envoltura convexa	Skew
Características intrínsecas del VRP	Distancia euclidiana entre los clientes y el centroide de los depots	Máximo
	Demanda de los clientes	Skew
	Capacidad de los vehículos	Skew
	Si el problema tiene capacidad	Valor único
	Costo de los vehículos	Skew y Kurtosis

Tabla 9.7: Variables seleccionadas para conformar el set de características $\mathcal{F}'$.

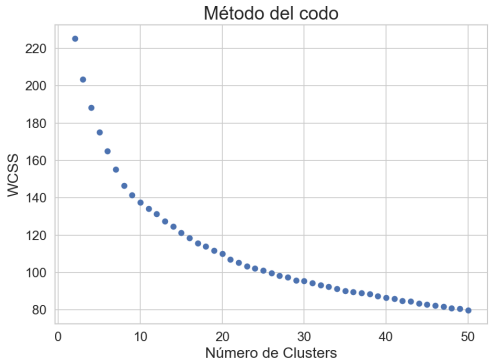
cuadrado para las características. Este método permite identificar si las distribuciones entre las variables es igual o distinta entre sí. En caso de ser iguales, se prescindirá de una de las dos.

Tras aplicar esta metodología de selección de características, el set de características $\mathcal{F}$ queda reducido a las variables mostradas en la tabla 9.7. A este nuevo set de características se les denomina $\mathcal{F}'$.

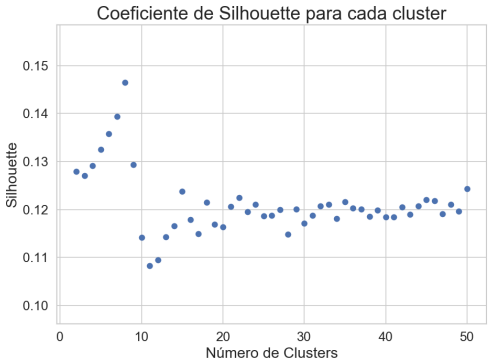
No se entrará en detalle de los resultados de los modelos anteriores tras haber hecho la selección de características, ya que se mostrarán en la siguiente sección.

Sin embargo, se mostrarán resultados relevantes, que muestran que gracias a esta selección, los métodos de clusterización aparentemente logran realizar una mejor clusterización.

En primer lugar, cuando se analizan nuevamente las métricas WCSS y Silhouette para el método de clusterización *k-means*, realizado con las características en $\mathcal{F}'$ se observa una mejoría. En este caso se logra apreciar que los clusters se encuentran mejor definidos con estas variables, ya que para el k escogido, las métricas WCSS y Silhoutte son mejores. Esto se puede apreciar en las gráficas 9.6a y 9.6b. A partir de estas gráficas, para el modelo de

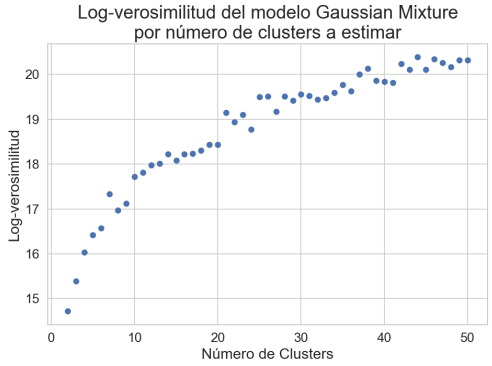


(a) Valor del WCSS para cada cluster.

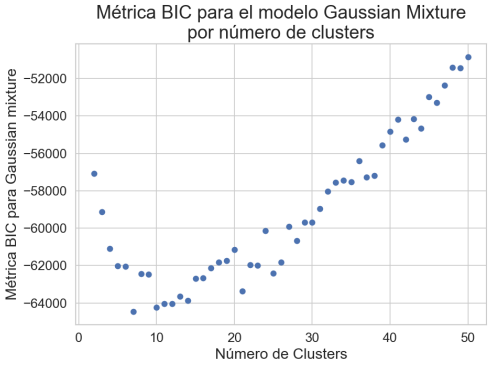


(b) Coeficiente de Silhouette para cada cluster.

Figura 9.6: Gráficas para escoger el k ideal usando el método de clusterización *k-means* sobre $\mathcal{F}'$.



(a) Log-verosimilitud del modelo para cada número de clusters.



(b) Métrica BIC para cada número de clusters.

Figura 9.7: Gráficas para escoger el número de componentes ideales en el modelo *Gaussian Mixture* sobre $\mathcal{F}'$.

k-means se escogió $k = 8$.

Luego, realizando el mismo análisis para las métricas del método de clusterización *Gaussian Mixture*, gráficas 9.7a y 9.7b, se puede apreciar claramente que el número ideal de componentes es 7.

Esto permite suponer que los clusters conformados son mejores con este set de características reducidos ($\mathcal{F}'$) al set de características completo ($\mathcal{F}$).

9.5. Comparación de resultados

Antes de comenzar la comparación de los resultados de los modelos antes expuestos, es importante destacar que se agregaron tres modelos sencillos, los cuales tienen la finalidad de

Modelos	Var. SS (%)	Var. ET (%)	Mejoran (%)	Empeoran (%)
VirtualBestSolver	-2.26	-28.23	68.42	2.27
$\mathcal{F}$-Kmeans-12	0.93	-5.95	30.47	34.46
$\mathcal{F}'$-Kmeans-8	1.18	-9.75	31.53	31.28
$\mathcal{F}'$-GM-7	1.19	-11.29	32.29	33.0
$\mathcal{F}$-GM-4	1.26	-12.04	33.3	30.52
BestSingleSolver	2.43	-10.44	26.78	32.49
$\mathcal{F}$-kNN-25	2.89	-24.23	35.73	37.9
$\mathcal{F}'$-kNN-25	3.22	-23.06	33.65	39.06
RandomModel	335293.33	21078.46	28.35	46.49

Tabla 9.8: Comparación de los resultados obtenidos con los diferentes modelos entrenados. También se incluyen los modelos entrenados a partir de la selección de características realizada en 9.4. Para diferenciar las características usadas, antes del nombre del modelo se indica que set de características se usó. *SS* se refiere al *SimplifyScore* y *ET* se refiere al *tiempo de ejecución*. Comparación realizada contra la situación actual.

servir como modelos de control para medir el desempeño de los otros modelos.

El primero de estos modelos corresponde al denominado **VirtualBestSolver**. Este modelo consiste en indicar cuál es la *configuración preferida* de cierta instancia sabiendo cual es, es decir, este modelo hace las veces de *oráculo*. Este modelo permite saber exactamente cuál es el límite de mejora de la solución propuesta en el capítulo 2. Este modelo en la práctica es imposible de utilizar, ya que para las instancias nuevas, *a priori* no se conoce cuál es la *configuración preferida* para dicha instancia.

El segundo modelo usado como control corresponde al denominado **BestSingleSolver**. Este modelo escoge una única configuración, la que resulta ser, en la base de entrenamiento, mejor en promedio.

Por último, el tercer modelo corresponde al denominado **RandomModel**. Este modelo lo que hace es seleccionar de forma aleatoria una configuración.

Finalmente, en la tabla 9.8 se aprecia la mejora en el desempeño esperado que tendría la heurística de optimización si se utilizara la configuración indicada por los diferentes modelos propuestos en este trabajo. Estos resultados están expresados en términos de la variación porcentual que se obtiene.

En primer lugar se analizarán los modelos de control planteados en esta sección. Como se puede apreciar, el modelo *VirtualBestSolver* indica que la mejora máxima que se puede lograr es de un 2.26 % y de un 28.2 %, en términos del **SimplifyScore** y el **tiempo de ejecución** respectivamente. También se aprecia que el método de *BestSingleSolver* empeora la situación actual en un 2.4 % en términos del **SimplifyScore**, pero mejora el **tiempo de ejecución** en un 10.4 %. Por último, el modelo *RandomModel* no presenta ninguna mejoría.

Nuevamente, se puede apreciar que ninguno de los modelos propuestos logra capturar el beneficio de escoger *inteligentemente* una configuración determinada.

Se esperaba que los modelos en que se redujo el set de características al conjunto $\mathcal{F}'$ tuvieran un mejor resultado, debido a que la clusterización resultaba ser mejor, sin embargo, no logró superar la solución entregada por la situación actual.

Además, como se revisó en las secciones anteriores, ningún modelo mostró ser consistente en dicha mejoría, es decir, toda mejoría viene con un grupo de casos que se ven perjudicados y en general, la porción de casos que mejora su solución es similar a la porción de casos que empeora su solución. Por esta razón, no se puede afirmar que alguno de estos modelos sea mejor que la situación actual.

De todas formas, el modelo $\mathcal{F}$-*Kmeans* con 12 clusters resulta ser de los mejores modelos, ya que en comparación al resto, es el de los que más se acerca a la situación actual en términos del **SimplifyScore**, obteniendo una reducción de 5.95 % en el **tiempo de ejecución** y la proporción de casos que empeoran su situación es parecida a la de los demás modelos. Además, cabe destacar que como se mostró en la tabla 9.2 para el cluster etiquetado como 10 se obtiene una mejora del 40.2 % en el *tiempo de ejecución* sin aumentar su *SimplifyScore*.

Resulta importante destacar y recordar que en la situación actual se ejecutan cuatro configuraciones diferentes y se escoge aquella que entregue la mejor solución (sec. 6.2.2). Esto hace que para los modelos aquí propuestos les sea bastante difícil ganarle a la situación actual, debido a que en esta se calculan cuatro soluciones, mientras que en los modelos se calcula sólo una. No obstante, en términos del *tiempo de ejecución*, gracias a los modelos propuestos se puede obtener una mejoría.

Ahora bien, si se compara el *BestSingleSolver* contra los modelos propuestos, se puede apreciar que los modelos planteados son mejores que escoger una sola configuración por defecto, pero no son mejores que la solución implementada por el equipo de SimpliRoute.

Capítulo 10

Resultados de los modelos planteados

En este capítulo se mostrarán los resultados obtenidos tras aplicar los modelos propuestos en el capítulo anterior y además se discutirán los resultados obtenidos.

10.1. Resultados en la base de testeo

En esta sección se revisarán los resultados que se obtienen al aplicar los modelos propuestos en el capítulo anterior pero en la base de testeo. Recordar que en el capítulo 8 se llevó a cabo una división de la base de instancias en dos, una base de entrenamiento y una base de testeo. La base de entrenamiento fue usada en los capítulos anteriores para describir y entrenar los modelos de predicción y la base de testeo será usada en este momento para evaluar el desempeño de los modelos.

A pesar de que los resultados obtenidos en el capítulo anterior fueron negativos y que no permiten suponer que en la base de testeo vayan a ser mejores, de todas formas se mostrarán estos resultados.

Modelos	**Var. SS (%)**	**Var. ET (%)**	**Mejoran (%)**	**Empeoran (%)**
VirtualBestSolver	-2.17	-28.05	71.02	2.59
$\mathcal{F}$-Kmeans-12	1.01	-4.51	28.15	37.22
$\mathcal{F}$-GM-4	1.02	-11.79	33.22	30.86
$\mathcal{F}'$-Kmeans-8	1.19	-10.06	29.56	34.63
$\mathcal{F}'$-GM-7	1.24	-11.07	31.21	35.69
BestSingleSolver	1.97	-10.07	28.62	33.69
$\mathcal{F}$-kNN-25	2.48	-21.68	34.39	40.28
RandomModel	342461.65	20303.82	28.27	49.00

Tabla 10.1: Resultados de la predicción de la mejor configuración de la heurística de optimización en la base de testeo. *SS* se refiere al *SimplifyScore* y *ET* se refiere al *tiempo de ejecución*. Comparación realizada contra la situación actual.

Label	Size	Var. SS (%)	Var. ET (%)	Mejoran (%)	Empeoran (%)
0	24	0.60	-8.72	20.83	33.33
1	95	0.64	3.29	15.79	16.84
2	84	0.87	-8.70	27.38	33.33
3	67	1.45	3.19	29.85	40.3
4	77	0.48	4.82	12.99	35.06
5	52	-0.58	-3.36	28.85	15.38
6	114	1.29	-6.68	36.84	39.47
7	18	4.45	-12.60	38.89	38.89
8	43	-0.43	1.55	39.53	41.86
9	143	1.62	-2.11	37.06	56.64
10	18	0.43	-43.92	44.44	11.11
11	114	1.36	-14.05	21.05	42.98

Tabla 10.2: Desempeño obtenido por cluster utilizando el método *k-means* utilizando las características en $\mathcal{F}$, con $k = 12$, en la base de testeo. Comparación realizada contra la situación actual.

En la tabla 10.1 se encuentran resumidos los resultados de los modelos propuestos en el capítulo anterior pero ejecutados sobre la base de testeo.

Tal como se esperaba, los resultados obtenidos en la base de testeo no son mejores que los obtenidos en la base de entrenamiento.

De lo modelos mostrados en la 10.1, el modelo que mejor desempeño obtiene en la base de testeo es el modelo $\mathcal{F}$-*Kmeans* con 12 clusters. Sin embargo, se considera mejor el modelo $\mathcal{F}$-*GM* con 4 componentes, ya que tiene una variación del **SimplifyScore** similar al modelo anterior pero logra una mayor disminución en el **tiempo de ejecución**, mejorando una mayor cantidad de casos y empeorando una menor cantidad de veces.

Por otro lado, en la sección 9.1 se identificó que para los casos en que se indicaba que cuando la instancia pertenecía al cluster con la etiqueta 10, esta presentaba una mejoría en su *tiempo de ejecución* sin disminuir su *SimplifyScore*. Dicha hipótesis no se puede validar en la base de testeo. Como se puede observar en la tabla 10.2, los casos etiquetados con el número 10 aumentan su **SimplifyScore** en un 0.43 % pero de igual forma disminuyen su **tiempo de ejecución** en un 43.92 %.

A partir de la tabla 10.2 se identifican dos nuevos clusters que logran mejorar su *SimplifyScore*, estos son el cluster con el número 5 y el número 8. Estos clusters en promedio logran disminuir el **SimplifyScore** en un 0.58 % y en un 0.43 % respectivamente. En particular, para el cluster número 5 sólo un 15.4 % de los casos empeora su situación actual y un 28.9 % la mejora. Sin embargo, a partir de los resultados obtenidos en la sección 9.1 (tabla 9.2), no era posible vislumbrar dichos beneficios, por lo que para validar este descubrimiento es necesario re-entrenar el modelo $\mathcal{F}$-*Kmeans-12* con otra(s) base(s) de instancia(s) y/o utilizar metodologías de *cross validations*, para así asegurarse que los resultados obtenidos son independientes a la partición realizada y son generalizables.

A partir de los resultados expuestos en las tablas 9.8 y 10.1 no se puede indicar que ninguno de estos modelos sea mejor que la situación actual en términos generales, por lo que no se recomienda implementar ninguno de estos en el proceso de optimización.

10.2. Propuesta nueva configuración

En esta sección se aprovecharán los datos obtenidos en el desarrollo de este trabajo de memoria para evaluar y plantear si existe una combinación de cuatro configuraciones diferentes a la actual que permitan mejorar el resultado entregado por la heurística de optimización.

Como se mencionó en el capítulo 6, al momento de ejecutar la heurística de optimización, esta ejecuta cuatro configuraciones distintas y de estos cuatro resultados escoge la que mejor solución entregue, es decir, la solución que menor **SimplifyScore** tenga. Sin embargo, tomar la decisión de cuáles cuatro configuraciones ejecutar y en que orden es una tarea que requiere de mucho tiempo, tanto como para calcular las soluciones con cada una de las configuraciones como para evaluar las diferentes combinaciones.

Dado que en este trabajo ya se llevó a cabo la primera parte de la tarea recién mencionada, es decir, calcular las soluciones que entrega la heurística de optimización con cada una de las configuraciones para cada instancia, se aprovecharán estos resultados para evaluar si existe una combinación de configuraciones diferente a la actual que genere una mejora en el resultado que entrega la heurística de optimización.

Se evaluó la totalidad de combinaciones posibles con las 24 configuraciones disponibles, lo cual equivale a 255024 escenarios posibles. En cada una de estas evaluaciones, se simuló la solución que entregaría la heurística de optimización tras ejecutar 4 configuraciones específicas.

Para realizar dicha simulación es necesario definir a que se refiere cuando se habla de un escenario posible. Un escenario consiste en 4 configuraciones diferentes que intentará ejecutar

Configuración	**Var SS (%)**	**Mejoran (%)**	**Empeoran (%)**
(7, 4, 19, 6)	-1.41	28.36	4.56
(4, 7, 17, 6)	-1.40	28.25	4.74
(4, 8, 19, 6)	-1.39	28.85	6.54
(1, 4, 19, 6)	-1.38	27.9	5.13
(4, 1, 19, 6)	-1.37	27.93	5.2
(1, 6, 8, 17)	-1.36	25.14	2.51
(4, 2, 19, 6)	-1.35	29.03	6.68
(1, 17, 8, 6)	-1.34	25.07	2.51
(4, 5, 19, 8)	-1.33	28.43	6.4
(4, 7, 19, 5)	-1.32	27.48	4.53

Tabla 10.3: Variación del *SimplifyScore* obtenida por los 255024 escenarios posibles. En esta tabla se encuentran los 10 mejores resultados.

la heurística de optimización. Para evaluar el escenario se toma el *tiempo de ejecución* obtenido por la primera configuración, si este tiempo es menor a 15 minutos, entonces se agrega su *SimplifyScore* a un arreglo de soluciones. Luego se itera con la siguiente configuración. Si el *tiempo de ejecución* de esta nueva configuraciones más las anteriores es menor a 15 minutos, entonces se agrega el *SimplifyScore* de esta configuración al arreglo de soluciones. Así se itera hasta que se ejecutan las 4 configuraciones propuestas o la suma de los *tiempos de ejecución* sea mayor a 15 minutos, cuando se cumple esa condición se supondrá que la heurística de optimización estima que no debe continuar, por lo que no ejecuta esta nueva configuración. Finalmente del arreglo de soluciones construido se escoge aquella solución que tenga el menor *SimplifyScore*.

Realizando el proceso anterior es posible simular cuál sería el resultado que entregaría la heurística de optimización en cada escenario propuesto. Tras calcular los resultados entregados por los escenarios posibles, se construye la tabla 10.3, donde se muestra la variación del *SimplifyScore* para cada uno de los escenarios mostrados en esa tabla. Como se puede apreciar, si en vez de ejecutar las configuraciones 1, 9, 17, 7 (configuraciones ejecutadas por la situación actual) se ejecutasen las configuraciones 7, 4, 19, 6 se obtendría una mejora del *SimplifyScore* de un 1.41 %, mejora que se encuentra a mitad de camino de la mejora entregada por el *Virtual Best Solver*.

En base a lo anterior, se recomienda que la heurística de optimización *intente* ejecutar las configuraciones 7, 4, 19, 6 en vez de las actuales y así entregar mejores soluciones.

Cabe notar que en la tabla 10.3 se omitió el *tiempo de ejecución* debido a que los datos usados para este experimento no permiten hacer el *tiempo de ejecución* comparable pero si el *SimplifyScore*. La razón de esto es que la heurística de optimización para cada una de las configuraciones calculan elementos comunes que de ser ejecutados todos juntos se calcularían una sola vez ahorrando *tiempo de ejecución*.

Es importante destacar que la mejoría encontrada con la ejecución de estas nuevas configuraciones, a pesar de acercarse a la mejora proporcionada por el *Virtual Best Solver* no significa que la solución propuesta al comienzo del trabajo (2.4.2) carezca de sentido, sino que son soluciones que siguen caminos distintos. La solución propuesta en esta sección apunta a probar varias configuraciones/heurísticas y luego escoger la mejor solución, es decir, sigue el camino de la *fuerza bruta*. Mientras que la solución propuesta en este trabajo apunta en dirigir los esfuerzos y plantear una configuración/heurística personalizada a cada instancia.

De todas formas, mientras se sigue desarrollando un modelo que permita ofrecer una configuración/heurística de forma personalizada a cada instancia, la mejora encontrada en esta sección permite mejorar en un 1.41 % las soluciones que entrega la heurística de optimización de SimpliRoute

10.3. Discusión de los resultados

Tras los resultados obtenidos en la aplicación de los diferentes modelos (tablas 9.8 y 10.1) se puede verificar que ninguno de los modelos propuestos son sistemáticamente mejor que la situación actual, en términos del **SimplifyScore**. No así, si se evalúa el **tiempo de ejecución** de la heurística de optimización obtenido por los modelos, se aprecia que en todos los casos este se reduce.

La ganancia en el *tiempo de ejecución* viene dada principalmente porque en vez de intentar ejecutar cuatro diferentes configuraciones, el modelo permite ejecutar una sola de estas, por lo que lógicamente disminuye el *tiempo de ejecución*.

La principal hipótesis de este trabajo era que mediante la caracterización y posterior clusterización de las instancias se lograrían formar grupos de instancias cuya *configuración preferida* fuese la misma.

Sin embargo, como muestran los resultados de este trabajo no se logró validar dicha hipótesis, ya que mediante las características y modelos de clusterización propuestos, no se logra una mejoría en el desempeño global de la heurística de optimización.

Se identifican diversas posibles causas de por qué no se logró generar un modelo que permitiese resolver la problemática propuesta, las que serán revisadas a continuación.

10.3.1. Extracción de características

Una causa identificada y que puede ser bastante relevante en el hecho de que no se lograsen buenos resultados, tiene que ver con que en el proceso de extracción de las características de las instancias no se haya realizado correctamente. Como es lógico, en este proceso de extracción se consideró toda la información contenida en la instancias, es decir, fue considerado el conjunto de nodos y de vehículos íntegramente. Sin embargo, esto no necesariamente tiene que ser así, ya que en las instancias pueden existir nodos que *a priori* no es posible atenderlos o vehículos que no se pueden utilizar y considerar estos datos como parte de la información para extraer las características sólo genera ruido en la extracción de las mismas.

Resulta contradictorio pensar que la heurística de optimización sea capaz de resolver el VRP en instancias que tienen puntos que son infactibles de atender. Sin embargo, por la forma en que están implementadas las heurísticas es posible ignorar dichos nodos y/o vehículos, ya que simplemente esos nodos/vehículos no son utilizados y la respuesta que entrega la heurística de optimización indica que esos nodos/vehículos no se deben incluir en la planificación.

Para estudiar los casos en que un nodo/vehículo es infactible, es decir, no es posible que sea atendido o que atienda nodos, se implementarán las siguientes tres reglas. Considere un nodo cualquiera y un vehículo cualquiera:

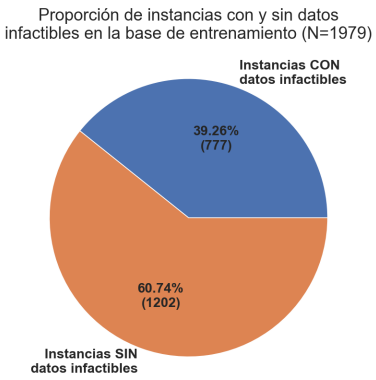


Figura 10.1: Proporción de instancias que tienen nodos/vehículos infactibles.

- Para atender dicho nodo se requiere que el vehículo posea cierta *skill*[25], si el vehículo no la posee, entonces dicho nodo no puede ser atendido por dicho vehículo.
- Si la demanda de dicho nodo es mayor a la capacidad del vehículo, entonces dicho nodo no puede ser atendido por el vehículo.
- El tiempo que le toma al vehículo en llegar, atender y volver del nodo es mayor a la jornada laboral indicada, entonces dicho vehículo no puede atender dicho nodo.

Ahora bien, para determinar si un nodo no se puede atender se revisan estas tres reglas para cada nodo de la instancia contra cada vehículo de la misma. Si no existe vehículo capaz de atenderlo, entonces se dice que el nodos es *inatendible*. Por el contrario, para determinar si un vehículo no se puede utilizar, nuevamente se revisan estas tres reglas para cada vehículo contra todos los nodos. Si dicho vehículo no es capaz de atender a ningún nodo, entonces se dice que el vehículo es *inusable*.

Tras la implementación de las reglas anteriores se pudo obtener la proporción de casos en que las instancias incluían algún nodo y/o vehículo infactible, dicha proporción se encuentra representada en la figura 10.1. Como se puede apreciar, en un 39.3 % de las instancias se da esta situación, proporción no menor, que es capaz de afectar los resultados de los modelos, ya que en ese mismo porcentaje de casos las características van a estar mal calculadas.

Analizando los casos en que existen datos infactibles, es posible identificar que, en promedio, sólo un 91.0 % de los nodos son visitas factibles, mientras que para el caso de los vehículos, en promedio, sólo un 72.4 % de la flota es factible de utilizar. En base a los números anteriores es posible indicar que las características presentan ruido en su cálculo.

Para evitar esta causa y mejorar los modelos propuestos en este trabajo y los modelos futuros, en el proceso de extracción de características se debe incluir un filtro que remueva aquellos nodos y vehículos que son infactibles, para que así las características que se extraigan correspondan efectivamente a la instancia que es resuelta por la heurística de optimización. Como punto de partida se propone comenzar con las reglas ya descritas, pero también se propone extender las mismas mediante la inclusión de la ventana horaria de los clientes, las

[25]Recordar que esta característica se encuentra descrita en la tabla B.1 del Anexo B.

Modelos	Var. SS (%)	Var. ET (%)	Mejoran (%)	Empeoran (%)
VirtualBestSolver	-2.26	-28.52	64.98	2.41
$\mathcal{F}$-Kmeans-22	0.66	-7.9	32.7	31.86
$\mathcal{F}$-GM-4	1.13	-9.75	30.53	28.04
$\mathcal{F}'$-Kmeans-7	1.19	-10.88	31.36	32.11
$\mathcal{F}'$-GM-8	1.24	-7.74	29.95	32.03
BestSingleSolver	2.21	-9.9	24.29	33.86
$\mathcal{F}$-kNN-25	3.35	-23.2	32.36	40.85
$\mathcal{F}'$-kNN-25	3.61	-22.82	31.2	41.51
RandomModel	313996.94	28317.93	25.87	47.09

Tabla 10.4: Resultados de la predicción de la mejor configuración de la heurística de optimización en la base de entrenamiento, removiendo aquellas instancias con puntos infactibles. *SS* se refiere al *SimplifyScore* y *ET* se refiere al *tiempo de ejecución*. Comparación realizada contra la situación actual.

zonas de atención a las que pertenecen los clientes y los vehículos.

Ahora bien, tras identificar las instancias que tienen nodos y/o vehículos infactibles, se procedió a calcular los modelos propuestos en el capítulo 9 de nuevo, obteniendo los resultados de la tabla 10.4. Como se puede apreciar en dicha tabla, los resultados son levemente mejores que los obtenidos sin realizar el filtrado de las instancias con filtros infactibles, sin embargo, siguen sin ser mejores que la situación actual.

10.3.2. Características y modelos propuestos

Otra causa identificada, tiene que ver con que en este trabajo se utilizaron instancias generadas en situaciones reales, lo que complica la utilización de características y de modelos que en *sets* clásicos de prueba suelen funcionar bien, como los utilizados y propuestos en la bibliografía presentada. De esta forma, quizás sea necesario replantear las características escogidas así como plantear modelos con otro enfoque o que sean más complejos.

El resultado de la tabla 10.4 permite reforzar la idea de que las características extraídas pueden no ser suficientemente buenas para describir a las instancias y en consecuencia representar el *espacio* de las configuraciones, la razón de esto es que a pesar de haber removido la información *infactible* que produce ruido en las características, los resultados no mejoran significativamente. De todas formas, para futuros análisis, se recomienda realizar esta limpieza en la información de las instancias, ya que es fuente de ruido en el cálculo de las características. En esta línea, resulta importante orientar la elección de nuevas características a aquellas que puedan describir algunos comportamientos y/o reglas de la heurística de SimpliRoute, para que estas sean más representativas de las decisiones que se toman en el proceso de búsqueda de la solución.

Como se mencionó, es posible que los modelos de clusterización y predicción propuestos no hayan sido los adecuados para lograr el objetivo, el cual a grandes rasgos consistía en

encontrar una relación entre la clusterización de las instancias en el plano de las características y la clusterización en el plano de las soluciones. Dado que lo anterior no se logró, se propone continuar con modelos de clasificación supervisados, en donde la *etiqueta* a mirar sea la *configuración preferida* y de esta forma *explicitar* la relación entre ambos planos. También se propone estudiar modelos que permitan predecir el desempeño de cada una de las configuraciones de la heurística, para luego así escoger aquella configuración con mejor predicción.

Una última posible causa identificada tiene relación con las configuraciones escogidas. En este sentido a pesar de haber mostrado que las diferentes configuraciones propuestas generan que la heurística de optimización tenga un mejor desempeño, las soluciones entregadas por muchas de estas resultan ser muy parecidas entre sí, razón por la cual es bastante más difícil encontrar una clara diferencia entre las configuraciones. En trabajos en que se lleva a cabo una tarea similar a la realizada en este, se utilizan heurísticas de optimización que son completamente distintas entre sí, razón por la que puede ser más evidente determinar cuando estas funcionan bien y cuando no.

Capítulo 11

Conclusión

En base a los esfuerzos realizados para solucionar la problemática expuesta al comienzo de este trabajo de memoria, y a los resultados obtenidos tanto en la base de entrenamiento (tabla 9.8) como en la base de prueba (tabla 10.1) se aprecia que ninguno de los modelos propuestos logran satisfacer el objetivo perseguido, ya que la configuración actual obtiene consistentemente mejores resultados que la solución examinada en este trabajo. No obstante, en el desarrollo de los objetivos específicos sí se lograron obtener resultados interesantes.

En primer lugar, se logró determinar un conjunto de parámetros de la heurística de optimización que generan que la misma heurística tenga comportamientos muy distintos, y en consecuencia se obtengan diferentes soluciones, las cuales en una cantidad relevante de casos permite mejorar la solución.

Sin embargo, pese a que se determinó una parametrización de la heurística de optimización que genera diferentes soluciones para las instancias, se debe tener en consideración que, en la generalidad, las variaciones de las soluciones encontradas pueden llegar ser pequeñas, lo que dificulta la personalización de las configuraciones, ya que obliga que los modelos sean bastante sensibles. Es por esto que para trabajos futuros, como primer paso, se recomienda utilizar parametrizaciones que produzcan que las soluciones sean fuertemente distintas entre sí, para luego ir calibrando los modelos. Un primer paso para abordar esto es comparar y utilizar heurísticas diferentes.

También se logró estipular una metodología la cual evita que en la base de datos a estudiar existan instancias muy similares entre sí. A pesar de las limitaciones que tiene esta metodología, identificadas en el trabajo, esta establece un marco de comparación de instancias que puede ser sencillamente calibrado. A partir de esta metodología se logró crear una base de instancias que evita tener instancias muy repetidas permitiendo realizar los modelos posteriores. Esta metodología además de ser útil en este trabajo, también resulta útil para estudios futuros que incluyan analizar a las instancias y/o a la heurística de optimización, ya que permite contar con una base de datos sin instancias repetidas y de esta forma enfocar los esfuerzos en la tarea que se esté llevando a cabo y no necesariamente en el proceso de limpieza de los datos.

Un resultado interesante e importante logrado durante este trabajo de memoria fue reafirmar la idea de que existe un beneficio real de escoger de forma *inteligente* la configuración de la heurística de optimización, ya que como se mostró, de escoger la *configuración preferida* de la instancia se puede llegar a mejorar la solución, en promedio, en un 2.26 % en términos del *SimplifyScore* y en un 28.2 % en términos del *tiempo de ejecución*.

A primera vista, una mejora del 2.26 % en el *SimplifyScore* puede no ser tan relevante, pero al analizarlo en detalle, como se hizo en la sección 8.1, se aprecia que en los casos en que el *SimplifyScore* varía, este mejora en un 6.0 % promedio y el *tiempo de ejecución* mejora en un 22.3 %. El caso en que el *SimplifyScore* no varía, el *tiempo de ejecución* mejora en un 31.8 %. El hecho de mejorar el tiempo de ejecución puede llevar a dos cosas, la primera y más directa es entregar la misma solución en menos tiempo y la segunda, complejizar la heurística actual para que ofrezca un mejor resultado en el mismo tiempo que se demora hoy en día.

El resultado anterior refuerza la intuición que se tenía al comienzo del trabajo de memoria, que era que mediante una personalización de la configuración de la heurística se pueden encontrar mejores soluciones con la misma heurística de optimización. Lo que motiva a continuar esta línea investigativa para encontrar una solución y/o modelo que permita capturar este beneficio encontrado. También surge una nueva línea investigativa relacionada a la cuantificación del impacto que tiene en los clientes de SimpliRoute las mejoras en las soluciones ofrecidas. Contar con esta información o una aproximación le permitiría a la empresa evaluar y priorizar sus desarrollos en términos del impacto generado.

Finalmente, los objetivos asociados a la determinación de las características que describen a las instancias y a la creación de los modelos de clusterización no se pudieron llevar a cabo con completo éxito. Sin embargo, las decisiones que se tomaron en este trabajo asociadas a las características escogidas así como a los modelos escogidos permite volver a abordar esta problemática pero desde un nuevo enfoque, como lo es utilizar características no incluidas en este trabajo, utilizar modelos de clusterización y/o de predicción más complejos o diferentes a los utilizados. Un ejemplo de otras características a considerar son las relacionadas a los *vecinos más cercanos*, y al *sondeo de los local search* propuestas por (Rasku et al. 2017), entre otras.

A partir de la discusión de los resultados, es importante destacar como conclusión que tanto para trabajos futuros como para el buen funcionamiento de la heurística de optimización es necesario limpiar la instancia de aquellos nodos y vehículos que son *a priori* infactibles de atender, ya que mantener esta información va en desmedro de los modelos que se alimenten de la información proveída por las instancias así como también va en desmedro del desempeño de la heurística de optimización, debido a que están evaluando puntos que nunca serán incorporados en la planificación de las rutas. Como se mostró en la discusión (sec. 10.3), un 39.3 % de las instancias estudiadas presentan nodos y/o vehículos infactibles, proporción no menor.

Un último punto relevante a destacar es que mediante el trabajo realizado en esta memoria se puede ratificar que las soluciones que entrega SimpliRoute actualmente son buenas, en el sentido de que no existe un camino obvio de mejora, es decir, la configuración por *default* con la que opera la heurística de optimización de SimpliRoute es difícil de mejorar.

Para poder ratificar la conclusión anterior con mayor certeza, es necesario medir la diferencia (GAP) que existe entre la solución entregada por SimpliRoute y la mejor solución conocida para un set de instancias. Esta información permitirá conocer el estado en el que se encuentra la heurística de optimización de SimpliRoute y qué tan urgente y necesario encontrar una solución a la problemática planteada en este trabajo de título. Como primera aproximación se propone estudiar set clásicos de *benchmark*, por ejemplo las instancias de Solomon.

Bibliografía

Adenso-Diaz, B. & Laguna, M. (2006), 'Fine-tuning of algorithms using fractional experimental designs and local search', *Operations Research* **54**(1), 99 – 114.

Audet, C. & Orban, D. (2006), 'Finfing optimal algorithmic parametres using derivative-free oprimization', *SIAM Journal on Optimization* **17**(1), 641 – 664.

Birattari, M., Stützle, T., Paquete, L. & Varrentrapp, K. (2002), A racing algorithm for configuring metaheuristics, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 11 – 18.

Bonet, C. (2018), 'Algoritmo basado en generacion de columnas para el problema de ruteo de vehículos dinámico', *Masters Thesis* .

Boyan, J., Moore, A. & Kaelbling, P. (2000), 'Learning evaluation functions to improve optimization by local search', *Journarl Machine Learning Research* **1**, 201 – 226.

Bräysy, O. & Gendreau, M. (2005), 'Vehicle routing problem with time windows, Part I: Route construction and local search algorithms', *Transportation Science* **39**(1), 104–118.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E. & Woodward, J. R. (2019), A Classification of Hyper-Heuristic Approaches : Revisited, *in* 'Handbook of Metaheuristics', Springer, Cham, chapter 14, pp. 453 – 477.

Christofides, N. (1976), 'Vehicle Routing Problem', *Revue française d'automatique, informatique, recherche opérationnelle* **10**(2), 55–70.

Christofides, N. & Eilon, S. (1969), 'An Algorithm for the Vehicle-Dispatching Problem', *OR* **20**(3), 309 – 318.

Christofides, N., Mingozzi, A. & Toth, P. (1981), 'Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relazations', *Mathematical Programming* **20**(1), 255 – 282.

Clarke, G. & Wright, J. (1964), 'Scheduling of vehicles from Central Depot to a Number od Delivery Points', *Operations Research* **12**(4), 568 – 581.

Dantzig, G. B. & Ramser, J. H. (1959), 'The Truck Dispatching Problem', *Management Science* **6**(1), 80–91.

Demkes, K. H.-j. (2014), 'Automated tuning of an algorithm for the vehicle routing problem', *Masters Thesis* .

Drexl, M. (2012), 'Rich vehicle routing in theory and practice', *Logistics Research* **5**(1-2), 47–63.

Echeverria, A. & Shats, E. (2018), 'SimpliRoute at Startup Battlefield Latin America 2018'.
URL: *https://techcrunch.com/video/simpliroute-at-startup-battlefield-latin-america-2018/*

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996), 'From Data Mining to Knowledge Discovery in Databases', *AI Magazine* **17**(3), 37 – 54.

Fukunaga, A. S. (2008), 'Automated Discovery of Local Search Heuristics for Satisfiability Testing', *Evolutionary Computation* **16**(1), 31 – 61.

Géron, A. (2017), *Hands-On Machine Learning with Scikit-Learn*, O'Reilly Media, Inc.

Ghaziri, H. (1991), 'Solving routing problems by a self-organizing map', *Artificial Neural Networks* pp. 829 – 834.

Glover, F. (1986), 'Future paths for integer programming and links to artificial intelligence', *Computers & Operations Research* **13**, 533 – 549.

González, V. & Uribe, L. (2018*a*), 'Como SimpliRoute resuelve el problema de ruteo de vehículos'.
URL: *https://www.simpliroute.com/post/como-simpliroute-resuelve-el-problema-de-ruteo-de-vehiculos*

González, V. & Uribe, L. (2018*b*), 'Ventanas horarias en el problema de ruteo de vehículos'.
URL: *https://www.simpliroute.com/post/ventanas-horarias-en-el-vrp*

Holland, J. H. (1975), 'Adaptation in Natural and Artifical Systems', *The university of Michigan Press* .

Hutter, F., Hoos, H. H., Leyton-Brown, K. & Stützle, T. (2009), 'ParamILS: An automatic algorithm configuration framework', *Journal of Artificial Intelligence Research* **36**, 267–306.

Hutter, F., Xu, L., Hoos, H. H. & Leyton-Brown, K. (2015), 'Algorithm runtime prediction: Methods & evaluation', *IJCAI International Joint Conference on Artificial Intelligence* **2015-Janua**, 4197–4201.
URL: *http://dx.doi.org/10.1016/j.artint.2013.10.003*

Khudabukhsh, A., Xu, L., Hoos, H. & Leyton-Brown, K. (2009), No SATenstein: automatically building local search SAT solvers from components, *in* 'International Joint Conference on Artificail Intelligence', pp. 517 – 524.

Laporte, G. (2009), 'Fifty years of vehicle routing', *Transportation Science* **43**(4), 408–416.

Lenstra, J. K. & Rinnooy, A. H. G. (1979), 'Complexity of Vehicle Routing and Scheduling Problems', **11**, 221–227.

Lu, Q. & Dessouky, M. M. (2006), 'A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows', *European Journal of Operational Research* **175**(2), 672–687.

Malitsky, Y. (2014), *Instance-Specific Algorithm Configuration*, Springer, New York.

Minton, S. (1996), 'Automatically configuring constraint satisfaction programs: A case study', *Constraints* **1**(1-2), 7 – 43.

O'Mahony, E., Hebrard, E., Holland, A., Nugent, C. & O'ullivan, B. (2008), Using case-based reasoning in an algorithm portfolio for constraint solving, *in* 'Irish Conference on Artificial Intelligence and Congitive Science'.

Partyka, J. & Hall, R. (2012), 'Software Survey: Vehicle Routing', *INFORMS ORMS-Today* .

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Pihera, J. & Musliu, N. (2014), 'Application of Machine Learning to Algorithm Selection for TSP', *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI* pp. 47–54.

Rasku, J., Kärkkäinen, T. & Musliu, N. (2017), 'Feature Extractors for Describing Vehicle Routing Problem Instances', *OASICS* **5**(7), 1–13.

Refalo, P. (2004), 'Impact-based search strategies for constraint programming', *Principles and Practice of Constraint Programming* **3258**, 557 – 571.

Savelsbergh, M. W. P. (2008), 'The Vehicle Routing Problem with Time Windows: Minimizing Route Duration', *ORSA Journal on Computing* **4**(2), 146–154.

Schumann, M. & Retzko, R. (1995), Self-organizing maps for vehicle routing problems - Minimizing an explicit cost function, *in* 'International Conference Artificail Neural Networks.', Paris, p. 401 / 406.

Smith-Miles, K. & Van Hemert, J. (2011), 'Discovering the suitability of optimisation algorithms by learning from evolved instances', *Annals of Mathematics and Artificial Intelligence* **61**(2), 87–104.

Steinhaus, M. (2015), 'The Application of the Self Organizing Map to the Vehicle Routing Problem'.

Wolpert, D. & Macready, W. (1997), 'No free lunch theorems for optimization', *Evolutionary*

Computation **1**(1), 67 – 82.

Xu, L., Hutter, F., Hoos, H. & Leyton-Brown, K. (2008), 'SATzilla: portfolio-based algorithm selection for SAT', *Journal of Artificial Intelligence Researchers* **32**(1), 565 – 606.

Anexos

A. Anexo A

En la tabla A.1 se muestran las distintas configuraciones de la heurística, las cuales son planteadas en el capítulo 6.

Nombre	**Nivel 1**	**Nivel 2**	**Nivel 3**	**Nivel 4**		
	Método	Criterio Agrupación	Criterio Balanceo	α	β	γ
Algoritmo 1*	ByArray	Ones	Falso	0.2	0.2	0.6
Algoritmo 2	ByArray	Ones	Falso	0.4	0.4	0.2
Algoritmo 3	ByArray	Ones	Verdadero	0.2	0.2	0.6
Algoritmo 4	ByArray	Ones	Verdadero	0.4	0.4	0.2
Algoritmo 5	ByArray	Demand	Falso	0.2	0.2	0.6
Algoritmo 6	ByArray	Demand	Falso	0.4	0.4	0.2
Algoritmo 7*	ByArray	Demand	Verdadero	0.2	0.2	0.6
Algoritmo 8	ByArray	Demand	Verdadero	0.4	0.4	0.2
Algoritmo 9*	ByNewMethod	Ones	Falso	0.2	0.2	0.6
Algoritmo 10	ByNewMethod	Ones	Falso	0.4	0.4	0.2
Algoritmo 11	ByNewMethod	Ones	Verdadero	0.2	0.2	0.6
Algoritmo 12	ByNewMethod	Ones	Verdadero	0.4	0.4	0.2
Algoritmo 13	ByNewMethod	Demand	Falso	0.2	0.2	0.6
Algoritmo 14	ByNewMethod	Demand	Falso	0.4	0.4	0.2
Algoritmo 15	ByNewMethod	Demand	Verdadero	0.2	0.2	0.6
Algoritmo 16	ByNewMethod	Demand	Verdadero	0.4	0.4	0.2
Algoritmo 17*	ByPaper	Ones	Falso	0.2	0.2	0.6
Algoritmo 18	ByPaper	Ones	Falso	0.4	0.4	0.2
Algoritmo 19	ByPaper	Ones	Verdadero	0.2	0.2	0.6
Algoritmo 20	ByPaper	Ones	Verdadero	0.4	0.4	0.2
Algoritmo 21	ByPaper	Demand	Falso	0.2	0.2	0.6
Algoritmo 22	ByPaper	Demand	Falso	0.4	0.4	0.2
Algoritmo 23	ByPaper	Demand	Verdadero	0.2	0.2	0.6
Algoritmo 24	ByPaper	Demand	Verdadero	0.4	0.4	0.2

Tabla A.1: Tabla resumen de las posibles configuraciones de la heurística. Con asterisco se identifican las configuraciones que actualmente la heurística intenta ejecutar (sec. 6.2.2).

B. Anexo B

En este anexo se encuentra la estructura de las *PeticionesWeb* que recibe la heurística de optimización. Esta información fue extraída de la documentación de la API de SimpliRoute, sección *optimización*[26].

Las *PeticionesWeb* constan de tres partes principales:

- **Vehículos:** Corresponden a un arreglo de elementos que contiene la información de los vehículos
- **Nodos:** Corresponde a un arreglo de elementos que contiene la información de los nodos a visitar
- **Opciones de optimización:** Corresponden a las diferentes opciones de optimización que se ofrecen

A continuación se detallarán cómo se conforman los elementos anteriores.

Vehículos

En primer lugar, se entiende por vehículo cualquier objeto o vehículo que sea capaz de moverse por la ciudad haciendo entrega del producto o servicio ofrecido por la empresa. Los atributos de los vehículos se encuentran resumidos en la tabla B.1.

Atributo	Descripción	Sub-atributos
Ident	Identificador único del vehículo	
Location Start	Ubicación donde el vehículo comenzará la ruta	ID, latitud y longitud
Location End	Ubicación donde el vehículo terminará la ruta	ID, latitud y longitud
Location Start	Ubicación donde el vehículo comenzará la ruta	ID, latitud y longitud
Capacidad	Capacidad máxima de productos que el vehículo puede llevar (Ej: Kilos)	
Capacidad 2	Segunda capacidad máxima de productos, complementaria a la primera (Ej: Metros Cúbicos)	
Capacidad 3	Tercera capacidad máxima de productos, complementaria a las anteriores (Ej: Número de cajas)	

[26]Link: https://documentation.simpliroute.com/es/#optimizacion

Atributo	Descripción	Sub-atributos
Shift Start	Hora en que el vehículo comienza su ruta	
Shift End	Hora máxima en que el vehículo debe terminar su ruta	
Skills	Habilidades que posee el vehículo. Deben haber visitas con habilidades también para que se use.	
Refill	Minutos que toma el vehículo en recargarse en bodega entre una ruta y otra (solo para casos en que el vehículo haga más de una ruta).	

Tabla B.1: Atributos de los **vehículos** en la *PetiónWeb*.

Nodos

Se entiendo por nodo como algo que debe hacer un vehículo en la calle, como por ejemplo entregar un pedido, visitar a un cliente, realizar una atención en terreno, etc. Los atributos de los nodos se encuentran resumidos en la tabla B.2.

Atributo	Descripción
Ident	Identificador único del objeto *delivery*
Latitud	Latitud donde se encuentra el lugar del *delivery*
Longitud	Longitud donde se encuentra el lugar del *delivery*
Load	Carga a entregar en el *delivery*. Debe estar en las mismas unidades que el parámetro capacidad de los vehículos.
Load 2	Unidad de carga secundaria del *delivery*. Debe estar en las mismas unidades que el parámetro capacidad 2 de los vehículos.
Load 3	Tercera unidad de carga del *delivery*. Debe estar en las mismas unidades que el parámetro capacidad 3 de los vehículos.
Window Start	Hora mínima en que este *delivery* puede ser realizado.
Window End	Hora máxima en que este *delivery* puede ser realizado.
Window Start 2	Segunda hora mínima en que este *delivery* puede ser realizado.
Window End 2	Segunda hora máxima en que este *delivery* puede ser realizado.
Duration	Tiempo en minutos que el vehículo estará realizando la entrega.
Skills Required	Habilidades que este *delivery* necesita para ser cumplido. Solo los vehículos que tengan todas estas habilidades podrán atender el *delivery*.
Skills Optional	Habilidades que este *delivery* necesita para ser cumplido. Solo los vehículos que tengan al menos una de estas habilidades podrán atender el *delivery*.
Priority Level	Permite definir que este *delivery* es más importante que otros. Los posibles valores son 1: Primero del día, 2: Último de su ruta, 3: Prioridad Alta, 4: Prioridad Media, 5: Prioridad Baja.

Tabla B.2: Atributos de los **nodos** en la *PetiónWeb*.

Opciones de optimización

Las opciones de optimización permite que las empresas puedan configurar la heurística de optimización de la forma que más se ajuste a su operación. Las opciones de optimización que se ofrecen se encuentran resumidas en la tabla B.3.

Atributo	**Descripción**
Balance	Si es *true*, la heurística intentará balancear las rutas de todos los vehículos, en tiempo y número de *deliveries* por ruta.
fmv	Corresponde al factor de tráfico. Puede ser 1.0 = Poco Tráfico, 1.5 = Tráfico Medio, 2.0 = Tráfico Alto o 3.0 = Tráfico Intensivo.
All Vehicles	Si es *true*, la heurística asignará rutas a todos los vehículos de la llamada. Si es *false*, la heurística decidirá cuántos y cuáles vehículos deben usarse.
Open Ended	Si es *true*, las rutas se crearán asumiendo que los vehículos no deben retornar a sus bodegas al final del día.
Single Tour	Si es *true*, los vehículos tendrán máximo 1 ruta cada uno.

Tabla B.3: Atributos de las **opciones de optimización** en la *PeticiónWeb*.

C. Anexo C

En la tabla C.1 se muestran los datos y resultados del test de hipótesis para la media de una muestra realizadas en el capitulo 8.

Configuración	**Promedio**	σ	N	$t - value$	$p - value$
Algoritmo 1	0.03	0.226	1979	5.827	0.0
Algoritmo 2	0.032	0.253	1978	5.66	0.0
Algoritmo 3	0.026	0.214	1978	5.3	0.0
Algoritmo 4	0.026	0.209	1979	5.457	0.0
Algoritmo 5	0.025	0.24	1977	4.654	0.0
Algoritmo 6	0.025	0.222	1978	4.912	0.0
Algoritmo 7	0.024	0.236	1979	4.583	0.0
Algoritmo 8	0.026	0.256	1978	4.496	0.0
Algoritmo 9	0.042	0.254	1977	7.28	0.0
Algoritmo 10	0.046	0.28	1979	7.294	0.0
Algoritmo 11	0.027	0.126	1978	9.681	0.0
Algoritmo 12	0.031	0.173	1978	8.049	0.0
Algoritmo 13	0.024	0.135	667	4.657	0.0
Algoritmo 14	0.034	0.227	673	3.828	0.0
Algoritmo 15	0.024	0.137	671	4.499	0.0
Algoritmo 16	0.032	0.226	674	3.632	0.0
Algoritmo 17	0.013	0.069	1978	8.652	0.0
Algoritmo 18	0.018	0.154	1977	5.267	0.0
Algoritmo 19	0.012	0.064	1976	8.518	0.0
Algoritmo 20	0.017	0.132	1978	5.652	0.0
Algoritmo 21	0.03	0.161	1977	8.285	0.0
Algoritmo 22	0.036	0.225	1979	7.159	0.0
Algoritmo 23	0.029	0.148	1979	8.618	0.0
Algoritmo 24	0.036	0.224	1979	7.082	0.0

Tabla C.1: Resultados test de hipótesis para la media de una muestra para las configuraciones. La hipótesis nula dice que la variación porcentual es igual a cero, como lo muestra la ecuación 8.3.

D. Anexo D

En la siguiente tabla (D.1) se encuentran resumidas todas las características propuestas en la sección 8.2 del capítulo 8.

Familia de características	**Características**	**Estadísticos**
Distribución de los nodos	Valores de la matriz de costo	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Máximo y Mediana.
	Cantidad de elementos distintos en la matriz de costo	Valor único
	Centroide de los puntos	Valor único para X y valor único para Y
	Distancia euclidiana de los nodos al centroide	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.
	Coeficiente de correlación	Valor único
	Número de clusters	Valor único
	Número de nodos en cluster	Valor único
	Número de outliers	Valor único
	Proporción de puntos en cluster	Valor único
	Alcance de los clusters	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.
	Coeficiente de Silhouette	Valor único
Minimum Spanning Tree	Costo de los arcos	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.
	Grado profundidad de los nodos	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Máximo y Mediana.
Características Geométricas	Área de la envoltura convexa	Valor único
	Ratio de puntos en el área sobre el total de puntos	Valor único
	Distancia de los nodos dentro de la envoltura convexa	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.
	Largo aristas envoltura convexa	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.

Tabla D.1: (Continúa...)

Familia de características	**Características**	**Estadísticos**
Características intrínsecas del VRP	Centroide de los puntos SIN escalar	Valor único para X y valor único para Y
	Número de clientes	Valor único
	Número de depots	Valor único
	Número de vehículos	Valor único
	Ratio clientes sobre vehículos	Valor único
	Centroide de los clientes	Valor único para X y valor único para Y
	Centroide del depot	Valor único para X y valor único para Y
	Distancia euclidiana entre el centroide de los clientes y de los depots	Valor único
	Distancia euclidiana entre los clientes y el centroide de los depots	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.
	Demanda de los clientes	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.
	Demanda total de los clientes	Valor único
	Capacidad de los vehículos	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.
	Capacidad total de los vehículos	Valor único
	Ratio demanda total sobre capacidad total	Valor único
	Si el problema tiene demanda	Valor único
	Si el problema tiene capacidad	Valor único
	Costo de los vehículos	Promedio, Desv. Est., Skew, Kurtosis, Coef. de Var., Mínimo, Máximo y Mediana.

Tabla D.1: Resumen de características usadas para describir a las instancias (cap. 8, sec. 8.2).

En la siguiente tabla (D.2) se encuentran resumidas las características seleccionadas tras haber filtrado por aquellos casos en que se tiene una alta correlación realizada en la sección 8.2.2 del capítulo 8. A este set de características se le denominó $\mathcal{F}$.

Familia de características	**Características**	**Estadísticos**
Distribución de los nodos	Valores de la matriz de costo	Promedio, Skew y Mediana.
	Cantidad de elementos distintos en la matriz de costo	Valor único
	Centroide de los puntos	Valor único para X y valor único para Y
	Distancia euclidiana de los nodos al centroide	Promedio, Desv. Est., Mínimo y Máximo
	Coeficiente de correlación	Valor único
	Número de clusters	Valor único
	Número de outliers	Valor único
	Proporción de puntos en cluster	Valor único
	Alcance de los clusters	Desv. Est.
	Coeficiente de Silhouette	Valor único
Minimum Spanning Tree	Costo de los arcos	Mediana
	Grado profundidad de los nodos	Skew y Kurtosis
Características Geométricas	Área de la envoltura convexa	Valor único
	Ratio de puntos en el área sobre el total de puntos	Valor único
	Distancia de los nodos dentro de la envoltura convexa	Skew y Mínimo
	Largo aristas envoltura convexa	Promedio, Skew, Coef. de Var., Mínimo

Tabla D.2: (Continúa...)

Familia de características	**Características**	**Estadísticos**
Características intrínsecas del VRP	Centroide de los puntos SIN escalar	Valor único para X y valor único para Y
	Número de depots	Valor único
	Número de vehículos	Valor único
	Ratio clientes sobre vehículos	Valor único
	Centroide del depot	Valor único para Y
	Distancia euclidiana entre el centroide de los clientes y de los depots	Valor único
	Distancia euclidiana entre los clientes y el centroide de los depots	Desv. Est., Mínimo y Máximo
	Demanda de los clientes	Skew, Kurtosis, Mínimo y Mediana.
	Demanda total de los clientes	Valor único
	Capacidad de los vehículos	Promedio, Skew y Kurtosis
	Capacidad total de los vehículos	Valor único
	Si el problema tiene demanda	Valor único
	Si el problema tiene capacidad	Valor único
	Costo de los vehículos	Promedio, Desv. Est., Skew, Kurtosis, Máximo

Tabla D.2: Resumen de características usadas para describir a las instancias post remoción de variables con alta correlación (cap. 8, sec. 8.2.2).

E. Anexo E

En este anexo se presentan las matrices de correlación de las características propuestas en la sección 8.2.2, capítulo 8.

La gráfica E.1 corresponde a la matriz de correlación de todo el set de características. La gráfica E.2 corresponde a la matriz de correlación tras haber removido los casos en que la correlación era mayor a -0.7 y menor a 0.7.

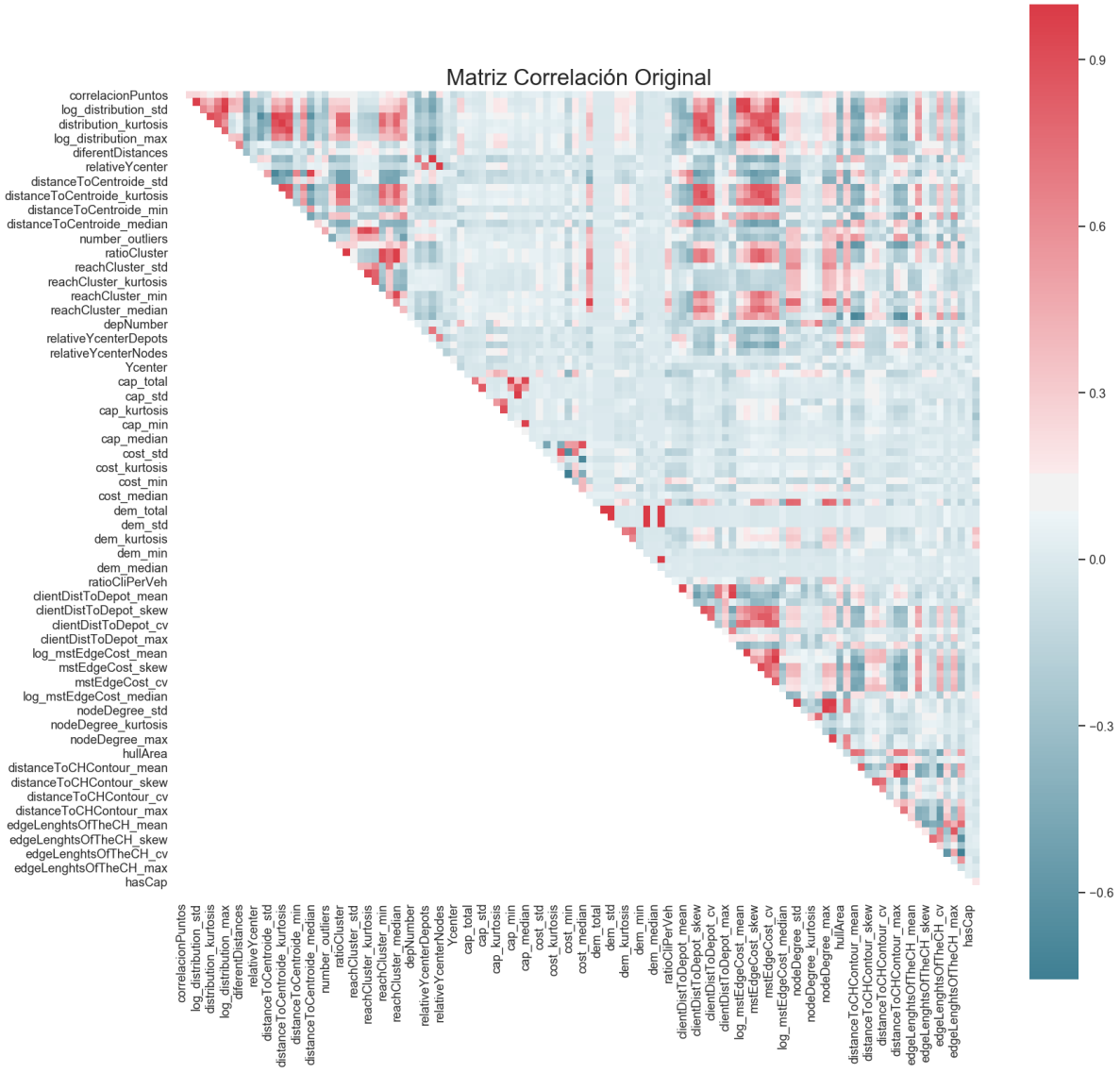


Figura E.1: Matriz de correlación del total de características propuestas.

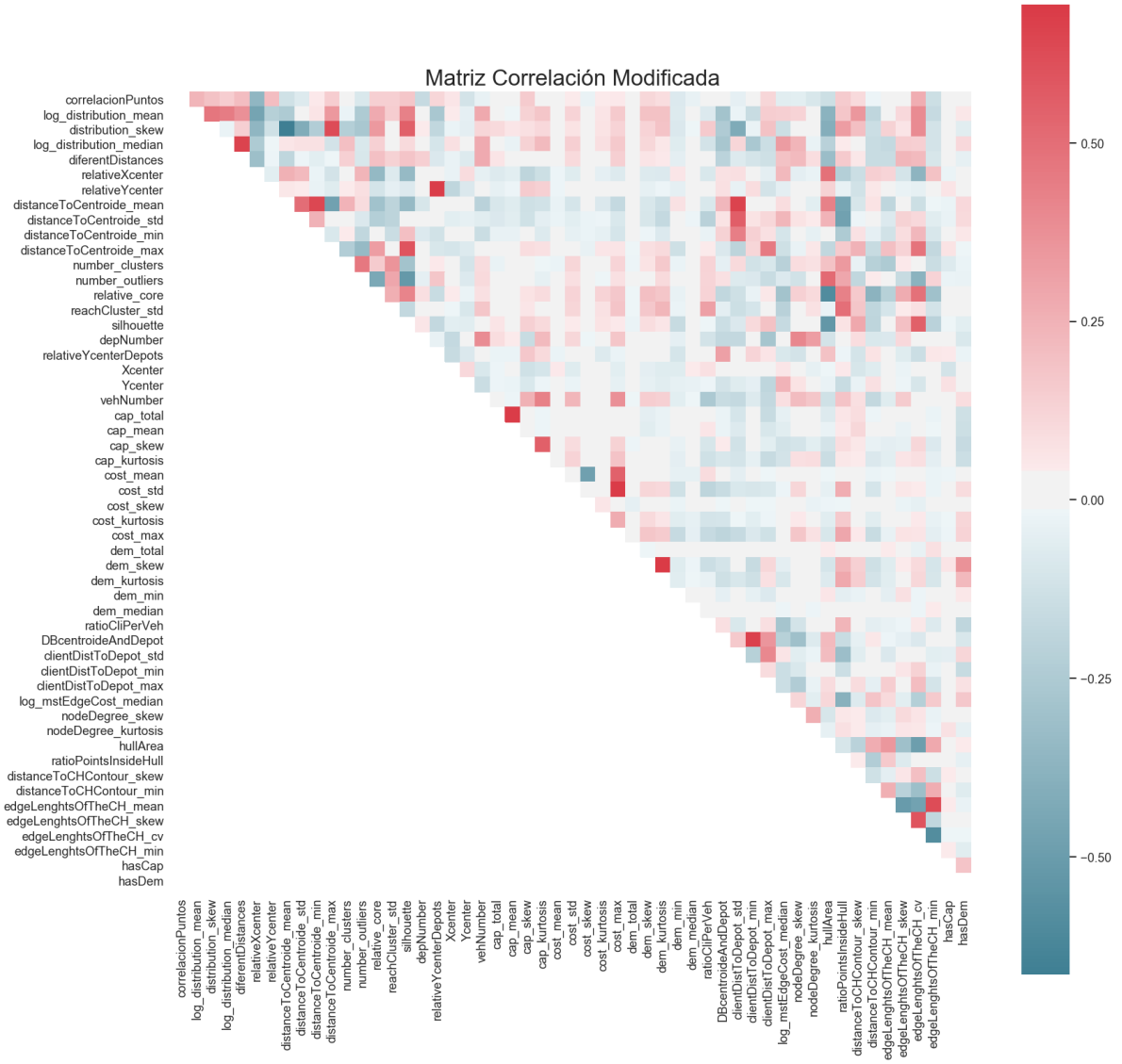


Figura E.2: Matriz de correlación con las características con una correlación mayor a -0.7 y menor a 0.7.

F. Anexo F

En este anexo se presentan los histogramas de las variables que se les aplicó la transformación logarítmica descrita en la sección 8.2.2. A la izquierda se encuentran la distribución de los valores sin transformar y a la derecha la distribución de los valores transformados.

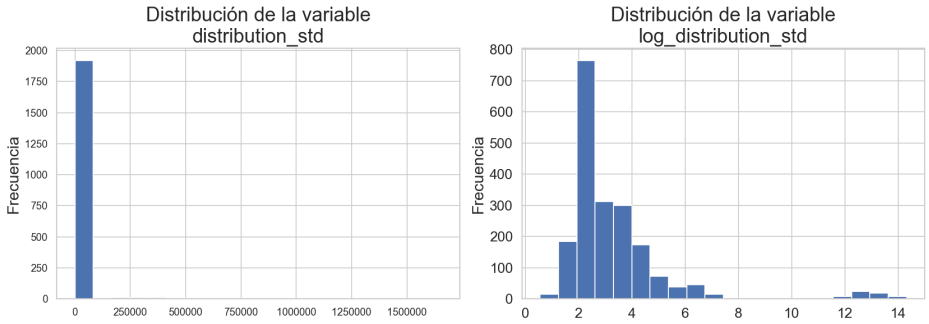


Figura F.1: Transformación logarítmica de la variable que representa la desviación estándar de los valores de la matriz de costo.

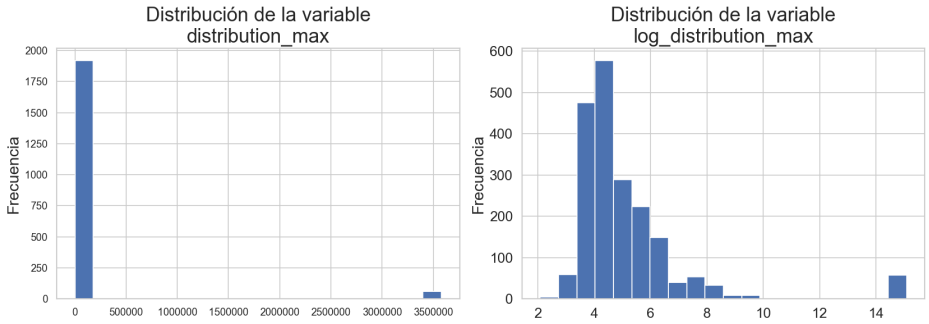


Figura F.2: Transformación logarítmica de la variable que representa el máximo de los valores de la matriz de costo.

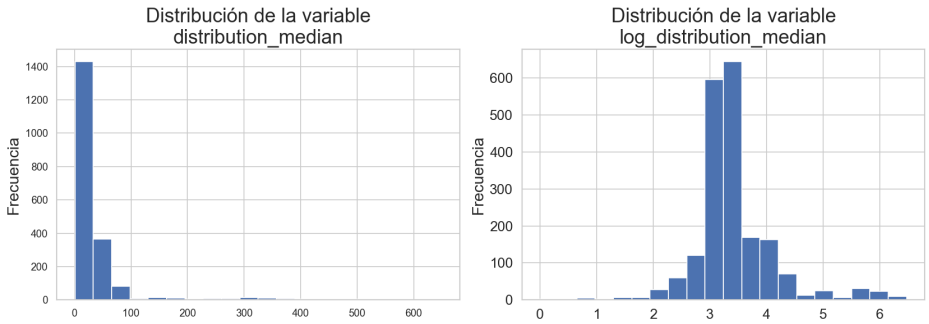


Figura F.3: Transformación logarítmica de la variable que representa la mediana de los valores de la matriz de costo.

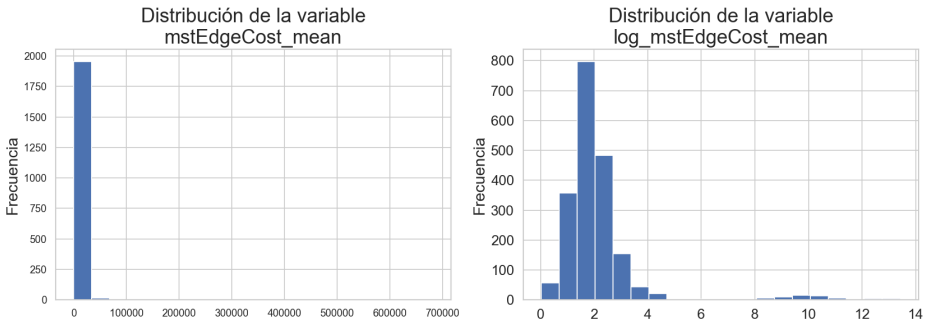


Figura F.4: Transformación logarítmica de la variable que representa el promedio de los valores del costo de los arcos obtenidos por el MST.

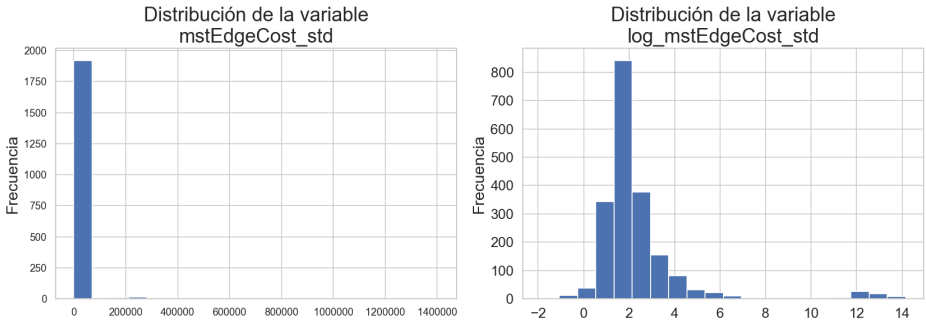


Figura F.5: Transformación logarítmica de la variable que representa la desviación estándar de los valores del costo de los arcos obtenidos por el MST.

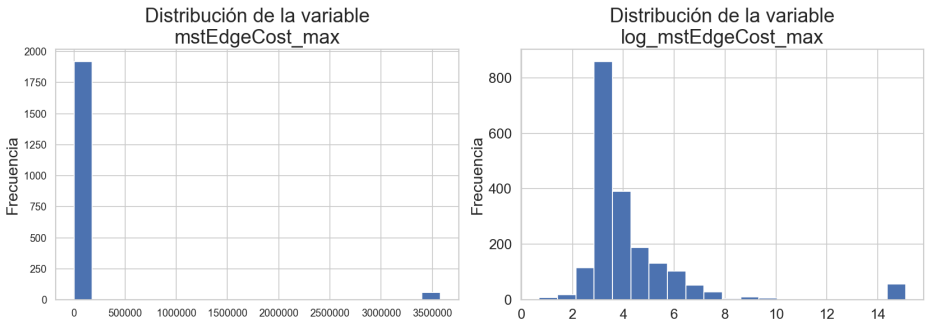


Figura F.6: Transformación logarítmica de la variable que representa el máximo de los valores del costo de los arcos obtenidos por el MST.

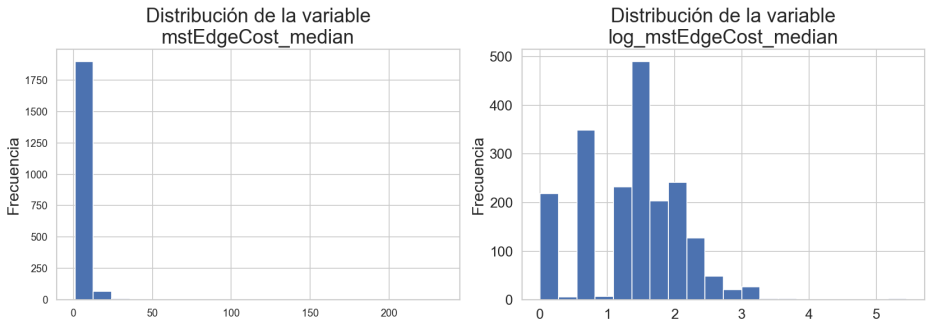


Figura F.7: Transformación logarítmica de la variable que representa la mediana de los valores del costo de los arcos obtenidos por el MST.