Universidad de La Habana
Facultad de Matemática y Computación

Exploración de vecindades grandes en el Problema de Enrutamiento de Vehículos usando técnicas estadísticas

Autor: **Héctor Felipe Massón Rosquete**

Tutores: **MSc. Fernando Raúl Rodríguez Flores**

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Mayo de 2020

A mis personas favoritas: mis padres, mis abuelos, mi hermano, mi Gaby,
y a mi familia.

Agradecimientos

Un agradecimiento especial a mis padres y mis abuelos por estar siempre presentes en mi vida, y darme todo de sí para hacer este día realidad.

A mi hermano y mi cuñada, por ser mi soporte, mi mejor amigo y mi apoyo incondicional.

A mi Gaby, por compartir junto a mí todos estos años y ser mi compañera de viaje.

A mi familia, por hacer de mí la persona que soy hoy y acompañarme no solo estos cinco años sino toda mi vida.

A mi tutor Fernando por apoyarme, responder siempre el teléfono, dedicarme parte de su tiempo (incluso las tardes) y saber ser una guía para mí durante estos años.

A todos los profesores que de una forma u otra han contribuído a mi formación, gracias por su apoyo y profesionalidad.

A los grandes amigos que conocí en la carrera, gracias por los buenos momentos.

Este trabajo es suyo.

Opinión del tutor

Dentro de los algoritmos de búsqueda local, la exploración de la vecindad de la solución actual es un paso vital para el buen desempeño de estos y al mismo tiempo, estas exploraciones pueden resultar difíciles y complejas de programar.

Con el trabajo realizado por Héctor en esta tesis es posible realizar fácilmente exploraciones de una vecindad de tantas maneras como se desee, y además, es posible utilizar algunas que nunca antes se habían reportado en la literatura.

Todo esto es posible gracias al talento, las habilidades y la dedicación que Héctor ha demostrado durante estos años de trabajo, en los que no solo hemos compartido la investigación, sino también la docencia en la que él participó como alumno ayudante. Algunos de los (mejores) resultados y propuestas para este trabajo, y para las clases, son de su autoría.

Por todo lo anterior, me siento muy satisfecho con los resultados obtenidos en todas las áreas que hemos compartido y considero que estamos en presencia de un trabajo excelente, desarrollado por un excelente científico de la computación.

MSc. Fernando Raúl Rodríguez Flores
Facultad de Matemática y Computación
Universidad de la Habana
Mayo, 2020

Resumen

Una de las técnicas más utilizadas para la solución del Problema de Enrutamiento de Vehículos (VRP) son los algoritmos de búsqueda local. Dentro de estos, la forma en que se explora la vecindad de la solución actual es fundamental para su buen funcionamiento y al mismo tiempo, estas exploraciones pueden resultar ineficientes y difíciles de programar. En este trabajo se presenta una propuesta para particionar una vecindad cualquiera en regiones y estimar las "mejores" regiones de la vecindad para intensificar la búsqueda en estas regiones. Las mejores regiones se estiman a partir del uso de técnicas estadísticas. Esta idea se implementa en un algoritmo de dos fases, en el que se usan generadores de soluciones vecinas también definidos en este trabajo como una implementación perezosa de las estrategias de exploración. Se realizó una experimentación de este algoritmo sobre 300 vecindades del VRP y los resultados se compararon con los correspondientes a una exploración aleatoria de las vecindades involucradas. Los resultados indican que el algoritmo de exploración en dos fases propuesto es significativamente superior a la estrategia aleatoria, siendo el enfoque adecuado si se desea aumentar el número de soluciones mejores analizadas durante la exploración de la vecindad.

Palabras claves: Problemas de enrutamiento de vehículos, algoritmos de búsqueda local, estrategias de exploración de vecindades grandes, generación perezosa de soluciones vecinas.

Abstract

One of the most used techniques for solving the Vehicle Routing Problem (VRP) are local search algorithms. Within these, the way in which the neighborhood of the current solution is explored is fundamental for its good operation and at the same time, these explorations can be inefficient and difficult to program. This paper presents a proposal to partition any neighborhood in regions and estimate the "best" regions of the neighborhood to intensify the search in these regions. The best regions are estimated from the use of statistical techniques. This idea is implemented in a two-phase algorithm, in which neighboring solution generators also defined in this work are used as a lazy implementation of the exploration strategies. An experimentation of this algorithm was carried out on 300 neighborhoods of the VRP and the results were compared with those corresponding to an random exploration of the neighborhoods involved. The results indicate that the proposed two-phase exploration algorithm is significantly superior to the randomized strategy, being the appropriate approach if you want to increase the number of better solutions analyzed during the neighborhood analysis.

Key words: Vehicle routing problems, local search algorithms, large neighborhood exploration strategies, lazy generation of neighboring solutions.

Índice general

Introducción

Nuestra sociedad se caracteriza por un constante y creciente flujo de personas, bienes y mercancías. Actualmente, la población mundial es de 7,8 billones de habitantes y se espera que alcance los 9,8 billones para el año 2050 [65]. Este incremento poblacional producirá un aumento en el movimiento de insumos y productos, con un costo y gasto de recursos enorme relacionado con el transporte de los mismos, por lo que el ahorro incluso de un pequeño porciento de estos recursos puede representar un impacto considerable para la sociedad.

En 2002, Toth y Vigo [55] reportaron que el uso de métodos computarizados en los procesos de distribución a menudo resulta en ahorros de entre el 5% y 20% del costo de transportación. Estos procesos de transportación se estudian bajo el nombre de Problema de Enrutamiento de Vehículos.

El Problema de Enrutamiento de Vehículos (VRP, por sus siglas en inglés), es un problema ampliamente estudiado en el ámbito de la optimización combinatoria [89]. En su forma más sencilla, está compuesto por un depósito central, un conjunto de vehículos idénticos, un conjunto de clientes distribuidos geográficamente, y su objetivo es encontrar una asignación de rutas para los vehículos tal que se optimice cierto criterio como: el costo de transportación, el tiempo de espera de los clientes, entre otros.

Dantzig y Ramser [34], fueron los primeros en introducir el "Problema de Despacho de Camiones", modelando cómo una flota de camiones idénticos podrían servir las demandas de combustible de un número de gasolineras, recorriendo una distancia mínima desde un depósito central.

Cinco años más tarde, Clarke y Wright [30], generalizaron este problema a uno de optimización lineal que se encuentra en el dominio de la logística de operaciones y el transporte: cómo servir a un conjunto de clientes, distribuidos geográficamente alrededor de un depósito central, usando una

flota de camiones de distintas capacidades. Este problema se conoce en la literatura como el "Problema de Enrutamiento de Vehículo" (VRP, por sus siglas en inglés).

Sin embargo, los modelos de VRP actuales son diferentes de aquellos introducidos por Dantzig y Ramser [34] y Clarke y Wright [30]. Estas diferencias se pueden considerar con respecto a las decisiones que se deben tomar en los mismos. En los VRP "clásicos" estas tienen que ver con la asignación de los clientes a las rutas y el orden en el cual los vehículos deben visitar los clientes que pertenecen a su ruta. En el VRP con Ganancias [5], es necesario además decidir a cuáles clientes visitar. En el VRP con Entregas Divididas [6], se decide la cantidad del producto que los vehículos deben descargar en los clientes. En el VRP con Múltiples Productos [91], es necesario decidir cuánto se debe descargar de cada producto por separado en los clientes. Además, existen otros VRPs que se caracterizan por la decisión de cuándo comenzar una ruta, entre estos se encuentran: el VRP Periódico [24] (PRP), el VRP de Inventario [4] (IRP), el VRP con Fechas de Lanzamiento [25], y el VRP con Múltiples Viajes [26] (MTVRP).

El VRP es un problema NP-duro [53], una generalización del problema del viajante [41] (TSP, por sus siglas en inglés), por lo que la utilización de métodos para computar una solución exacta del mismo solo son factibles para problemas de pequeña dimensión. A diferencia del TSP, donde instancias del problema para 1000 clientes pueden ser resueltos exactamente, instancias del VRP con más de 100 clientes pueden ser difíciles de resolver [12]. Por ese motivo, se han desarrollado diferentes heurísticas y metaheurísticas para resolver de forma aproximada este tipo de problemas, entre las que se encuentran los algoritmos de búsqueda local [15, 67].

Los algoritmos de búsqueda local, son métodos en los que se define una solución actual, una estructura de entorno (o criterio de vecindad, como también se conoce) y de forma iterativa se generan soluciones vecinas de la solución actual a partir del uso de la estructura de entorno.

En algoritmos como Búsqueda de Vecindad Variable (VNS) [57, 58] o Búsqueda de Vecindad Infinitamente Variable (IVNS) [63], se pueden considerar varias estructuras de entorno (o incluso infinitas, como en el caso de IVNS). Al aplicar estas estructuras sobre la solución actual del problema, pueden surgir vecindades muy grandes para las cuales resulta prohibitivo realizar una búsqueda exhaustiva de la vecindad, y realizar una búsqueda aleatoria no garantiza encontrar soluciones vecinas con una menor evaluación que la solución actual.

El objetivo general de este trabajo es diseñar e implementar métodos que permitan encontrar la mejor solución posible en una vecindad de un problema de enrutamiento de vehículos cualquiera.

Los objetivos específicos son:

- Estudio de las estrategias de exploración de una vecindad en un VRP.
- Estudio del sistema al que se le agregará la funcionalidad.
- Estudio, diseño e implementación de técnicas para calcular la cardinalidad de una vecindad dada en cualquier variante de un VRP.
- Estudio, diseño e implementación de las técnicas estadísticas para estimar las mejores regiones dentro de la vecindad.
- Realizar experimentación numérica de la propuesta realizada.

Este documento está estructurado de la siguiente forma: en el capítulo 1 se presentan los elementos fundamentales del trabajo: problemas de enrutamiento de vehículos, algoritmos de búsqueda local, criterios de vecindad, proceso de exploración de una vecindad, algoritmos híbridos de solo-exploración solo-explotación y técnicas estadísticas. En el capítulo 2 se define una nueva solución y operaciones para resolver el problema del conteo de vecinos en una vecindad del VRP, así como el algoritmo en sí y una estructura arbórea llamada árbol de vecindad, la cual constituye una representación de la propia vecindad. En el capítulo 3 se definen las funciones de indexación que determinan un orden entre las soluciones de la vecindad, y se presenta el concepto de región así como una forma de particionar una vecindad en las mismas. En el capítulo 4 se introducen los generadores para explorar una vecindad del VRP usando y combinando diferentes estrategias de exploración, además en este capítulo se presenta un algoritmo para explorar una vecindad del VRP usando los conceptos definidos en los anteriores y los generadores introducidos en este. En el capítulo 5 se muestran los resultados obtenidos al utilizar el algoritmo; finalmente se presentan las conclusiones, recomendaciones, trabajos futuros y la bibliografía consultada.

Capítulo 1

Preliminares

En este capítulo se presentan los elementos sobre los que se desarrolla el resto del trabajo, que son: el problema de enrutamiento de vehículos (VRP) y algunas heurísticas utilizadas para su solución, las características fundamentales de los criterios de vecindad utilizados en los algoritmos de búsqueda local y las operaciones que los componen, además del proceso de exploración de una vecindad, y algunas técnicas estadísticas.

En la sección 1.1 se introduce el problema de enrutamiento de vehículos y algunas de sus variantes. En la sección 1.2 se presentan una clase de algoritmos para la solución aproximada de los VRPs, las metaheurísticas de Búsqueda Local. En la sección 1.3 se definen algunos de los criterios de vecindad utilizados para resolver el VRP y las operaciones que los componen. En la sección 1.4 se presenta el proceso de exploración de una vecindad, dentro de un algoritmo de búsqueda local, y en la sección 1.5 se introduce un tipo de algoritmos que definen una forma distinta y novedosa de realizar dicho proceso dividiendo este en dos etapas. Por último, en la sección 1.6 se presentan las técnicas estadísticas utilizadas en este trabajo para determinar las regiones más prometedoras dentro de una vecindad.

1.1. Problema de Enrutamiento de Vehículos

El Problema de Enrutamiento de Vehículos (VRP, por sus siglas en inglés) ha sido estudiado por más de 60 años, desde que Dantzig y Ramser [34] formularan el "Problema de Despacho de Camiones". La literatura sobre este tema está creciendo a un ritmo mayor que nunca ([38] reporta un

crecimiento exponencial con un aumento anual del 6%), al punto de que una simple búsqueda por la frase "vehicle routing" entre los años 2010 y 2020 produce un total de 41,600 resultados en Google Schoolar[1]. Debido a la creciente literatura en el tema surge la necesidad de clasificar las distintas variantes de este problema, así como agrupar los artículos donde se presentan los mismos. En este sentido, algunas taxonomías y estudios sobre el VRP han sido publicados en los últimos años [38, 52, 81, 90, 19, 89].

El VRP consiste en un conjunto de clientes que deben ser visitados y un conjunto de vehículos que deben visitar esos clientes. Una ruta es el recorrido que realiza un vehículo dado para visitar un grupo de clientes en un orden determinado, comenzando y teminando siempre en un depósito. El objetivo de este problema, es encontrar una asignación de rutas para los vehículos tal que se optimice uno o varios criterios como la distancia total recorrida por los vehículos [84], el tiempo total del recorrido [21], el tiempo de espera de los vehículos para servir a los clientes [21], el número de vehículos requeridos para servir a todos los clientes [33], entre otros.

La versión más simple del VRP es el Problema de Enrutamiento de Vehículos con restricciones de Capacidad (CVRP) [84], que puede definirse formalmente como un conjunto de n clientes que deben ser servidos por un m vehículos idénticos. Cada cliente i tiene una demanda r_i, y la suma de las demandas de los clientes asignados a un vehículo k deben de ser menor que la capacidad del vehículo Q. Todos los vehículos deben comenzar y terminar su ruta en el depósito. El objetivo de este problema es minimizar la suma de los costos de transportación de cada uno de los vehículos. El VRP Acumulativo [66, 72, 48], surge a partir de tomar como función objetivo la que minimiza la suma de los tiempos de arrivo en los clientes.

Es posible extender el CVRP variando las capacidades de los vehículos y el costo de los mismos, el problema que resulta de esta modificación es conocido en la literatura como el VRP con Flota Heterogénea (HFVRP) [42, 11], también conocido como el VRP con FLota Mixta. El VRP con Ventanas de Tiempo (VRPTW) [21, 22, 88], asume que las entregas de un cliente determinado deben ocurrir dentro de un cierto intervalo de tiempo, que varía de un cliente a otro. Las ventanas de tiempo se definen como duras (o estrictas) cuando no se permite la entrega fuera del intervalo de tiempo [2, 87]. Las ventanas de tiempo blandas, por otro lado, permiten las entregas fuera de los límites del intervalo contra un costo de penalización [79, 39].

[1]Los resultados de la búsqueda en Google Schoolar fueron obtenidos el 27 de Mayo de 2020 para la frase encerrada en comillas dobles.

La adición al VRP de restricciones de tiempo hace el problema mucho más complicado, tan solo la búsqueda de una solución factible (no necesariamente óptima) para el mismo es NP-duro [74].

En el VRP con Recogida y Descarga (VRPPD), los productos deben recogerse en un lugar determinado y entregarse en su destino. La recogida y descarga deben realizarse por el mismo vehículo por lo que la ubicación de recogida y descarga deben incluirse en la misma ruta [80]. Un problema relacionado es el VRP con Backhauls (VRPPB), donde un vehículo realiza entregas y recogidas en una ruta [69]. Algunos clientes requieren entregas (denominadas linehauls) y otros requieren recogidas (denominadas backhauls).

El VRP con Múlitples Depósitos (MDVRP) [61], asume que múltiples depósitos están distribuidos geográficamente entre los clientes [76, 51]. El VRP Periódico (PVRP) [24], se definió por primera vez en [13] motivado por una aplicación en el contexto de la recolección de basura, este se utiliza cuando la planificación se realiza durante un cierto período y las entregas al cliente se pueden realizar en diferentes días [45, 93]. En el PVRP, los clientes pueden ser visitados más de una vez, aunque a menudo con frecuencia limitada.

Todos estos problemas son NP-Duro [53], y constituyen una generalización del problema del viajante (TSP, Traveling Salesman Problem) [41]. El incremento del número de clientes en el mismo produce un aumento exponencial en la cantidad de soluciones factibles del problema, por lo tanto, la utilización de algoritmos exactos solo es posible para resolver instancias pequeñas del mismo. Para darle solución al resto de estos problemas, se usan métodos aproximados como las heurísticas y metaheurísticas [49], un ejemplo de estos métodos se presenta a continuación.

1.2. Metaheurísticas Búsqueda Local

Las métodos basados en búsquedas locales, son algoritmos en los que se define una solución actual, una estructura de entorno y de manera iterativa se exploran las soluciones vecinas a la actual, en busca de una solución mejor. La vecindad de la solución actual se define como aquel conjunto de soluciones que se pueden formar a partir de la aplicación de la estructura de entormo sobre la solución actual. El número de soluciones que pertenecen a la vecindad es un aspecto muy importante: mientras más grande es la

misma, mayores posibilidades existen de que contenga buenas soluciones. Sin embargo, mientras más grande es la vecindad más costosa se hace su exploración.

En los algoritmos de búsqueda local, se examina la vecindad de la solución actual para encontrar soluciones mejores. En dependencia de la estrategia de selección (sección 1.4) usada se puede tomar la primera mejora o la de menor costo entre todos los vecinos.

Las metaheurísticas son usadas para escapar de esos mínimos locales mediante dos formas: permitiendo la posibilidad de aceptar soluciones que no mejoren a la actual, o por una expansión de la vecindad.

En Recocido Simulado [1] la selección de soluciones peores que la actual se controla mediante un parámetro llamado temperatura, por analogía con el proceso de recocido de un cristal. Se acepta un aumento δ en el costo con probabilidad proporcional a $e^{-\delta/T}$, donde T representa el tiempo en dicho proceso, de modo que el algoritmo pueda converger a un mínimo global. En [14, 28, 85] se muestra la utilización de esta metaheurística para la solución del VRP.

La Búsqueda Tabú [43, 44] permite la selección de soluciones peores que la actual, y luego coloca el movimiento inverso en una "lista tabú" para que este no se pueda "deshacer". Esto permite que la vecindad de la nueva solución pueda ser explorada adecuadamente, previniendo la ocurrencia de ciclos. Una gran cantidad de métodos han sido sugeridos para resolver el VRP utilizando la Búsqueda Tabú, en [22] se destacan catorce aplicaciones notables de la misma. Algunos de los más exitosos son [29, 31, 78, 83].

En Búsqueda de Vecindad Variable [46] (VNS, por sus siglas en inglés), se considera una secuencia de vecindades de tamaño creciente alrededor de la solución actual. Una vecindad se analiza completamente antes de moverse a la siguiente vecindad más grande, y se volverá a la más pequeña cuando se encuentre una solución vecina que mejore a la actual. Algunos ejemplos de la utilización de estos métodos para la solución del VRP son [20, 32, 68, 37, 3].

En Búsqueda de Vecindad Infinitamente Variable [63] (IVNS), se sigue la misma idea presentada en VNS de cambiar sistemáticamente de estructura de entorno, pero considerando un conjunto infinito de los mismos. Cada vez que sea necesario cambiar la estructura de entorno, en lugar de seleccionar otra del conjunto finito y predefinido al comienzo del algoritmo, se genera un nuevo criterio de vecindad del conjunto (infinito) de todas

las posibles estructuras de entorno del problema que se desea resolver.

Los algoritmos de búsqueda local usan las estructuras de entorno, o criterios de vecindad, para definir el conjunto de soluciones a partir del cual la solución actual será mejorada. En la siguiente sección se presentan algunos de los criterios utilizados para la solución del VRP, así como las características de los mismos.

1.3. Criterios de Vecindad o estructuras de entorno

Los criterios de vecindad o estructuras de entorno, como también se les conoce, son estructuras fundamentales en el diseño de un algoritmo de búsqueda local. Tanto así, que en la literatura existen variantes de este tipo de métodos que se diferencian solamente en la cantidad de criterios de vecindad que utilizan, como se mostró en la sección 1.2.

Formalmente, un criterio de vecindad se puede definir en el espacio de soluciones X como una aplicación $\eta : X \to 2^X$, tal que a cada solución $x \in X$ le asigna un conjunto de soluciones $\eta(x) \subset X$, las cuales se llaman soluciones vecinas de x [63].

Estos criterios de vecindad están relacionados con el problema que se desea resolver, y en el caso del Problema de Enrutamiento de Vehículos estos están compuestos por operaciones que modifican las rutas, variando el orden o la distribución de los clientes en las mismas. En [36] se presentan los siguientes criterios de vecindad para resolver un VRP:

1. Mover a un cliente dentro de su ruta.
2. Mover un cliente hacia otra ruta.
3. Intercambiar dos clientes en la misma ruta (cambiar su orden).
4. Intercambiar dos clientes de rutas diferentes.
5. Mover una subruta dentro de su ruta.
6. Mover una subruta hacia otra ruta.
7. Intercambiar dos subrutas
8. Invertir el orden de los clientes dentro de una ruta.

En estos criterios de vecindad se pueden identificar elementos comunes [63], los cuales representan las operaciones o acciones a partir de las cuales se pueden formar todos los criterios. Estas operaciones son:

- Seleccionar una ruta.
- Seleccionar un cliente de una ruta.
- Insertar un cliente en una ruta.
- Intercambiar dos clientes.
- Seleccionar una subruta de una ruta.
- Insertar una subruta en una ruta.
- Intercambiar dos rutas.
- Invertir el orden de los clientes en una subruta.

Existe un subconjunto de estas operaciones mediante la combinación de las cuales es posible reproducir la funcionalidad de todas las operaciones mencionadas enteriormente, estas operaciones serán nombradas operaciones elementales [36]. Entre las mismas se encuentran: la selección de una ruta, selección de un cliente e inserción de un cliente.

Por ejemplo, es posible definir una operación de intercambio de clientes como dos operaciones de inserción de clientes donde el primer cliente se inserta en la posición y ruta correspondientes a la selección del segundo y viceversa. Además, las operaciones sobre subrutas se pueden emular mediante el manejo de una secuencia de clientes (por la propia definición de subruta). En este sentido, una selección de subruta se transforma en seleccionar tantos clientes como longitud tenga la subruta de la posición donde esta comienza. Una inserción de subruta se representa por la inserción de tantos clientes como longitud tenga la subruta en la posición donde esta es insertada pero en el orden inverso de la selección. Por último, el intercambio de dos subrutas se transformará en dos inserciones de subrutas con un intercambio en las rutas y posiciones de las subrutas, estas inserciones de subrutas se manejarán como se definió anteriormente.

El criterio de vecindad define el conjunto de soluciones que debe ser explorado a partir de la solución actual, para seleccionar ciertas soluciones que permitan mejorar los resultados obtenidos. La forma en la que dicha

exploración y selección se realizan, estará sujeta a una estrategia de exploración y selección respectivamente. En la siguiente sección se presentan algunas de estas estrategias.

1.4. Exploración de una vecindad

En los algoritmos de búsqueda local, la búsqueda de un vecino más conveniente en el entorno de la solución actual puede realizarse de diferentes formas. En la literatura [47, 63], se pueden encontrar las siguientes variantes para realizar la búsqueda de tal solución:

Búsqueda aleatoria: Se selecciona una solución aleatoria del entorno.

Búsqueda exhaustiva: Se exploran todas las soluciones del entorno y se devuelve la mejor.

Búsqueda primera mejora: Se comienza a explorar la vecindad exhaustivamente, hasta que se encuentra una solución mejor que la actual, y esa es la que se devuelve. Si no se encuentra esta solución, entonces esta exploración es equivalente a la exhaustiva.

Búsqueda primera mejora aleatoria: Se generan N soluciones aleatorias y se devuelve la mejor de ellas, a no ser que antes de generar N, aparezca una mejor que la actual.

Todas estas variantes tienen dos elementos fundamentales: la forma de explorar el entorno de las solución actual y seleccionar uno de los vecinos analizados.

La forma en que se exploran las soluciones en la vecindad de la solución actual representa una estrategia de exploración. A continuación se reportan las más notables en la literatura:

Exploración exhaustiva: Se analizan todos los vecinos partiendo de una solución inicial [7].

Exploración aleatoria: Se analiza solamente un subconjunto aleatorio de los posibles vecinos [7].

La forma en que se selecciona el vecino para sustituir a la solución actual al explorar la vecindad de la misma representa una estrategia de selección. En la literatura se pueden encontrar los siguientes ejemplos [7]:

Mejor vecino: Se devuelve el mejor vecino encontrado entre todos los explorados.

Mejor vecino aleatorio: Se devuelve un vecino aleatorio entre todos los vecinos que mejoren la solución actual.

Primera mejora: Se devuelve el primer vecino que mejore la solución actual.

De los ejemplos anteriores resulta llamativa la existencia de estrategias de selección para las cuales la solución retornada no tiene por qué ser necesariamente la mejor de la vecindad. Esto se debe a que seguir una estrategia golosa donde siempre se seleccione la mejor solución en la vecindad puede provocar que el algoritmo quede atrapado en un óptimo local. En muchos casos, la selección de una solución mejor que la actual, aunque no sea la mejor, puede producir muy buenos resultados [8].

Independientemente de la estrategia de exploración utilizada, el principal problema al explorar una vecindad del VRP es el nulo conocimiento previo que se tiene sobre la estructura de la misma. Al realizarse dicha exploración a oscuras, puede usarse una estrategia de exploración que no sea la más ideal para explorar una vecindad dada, como por ejemplo: seguir una estrategia exhaustiva para una vecindad muy grande, o seguir una estrategia aleatoria para una vecindad pequeña.

En este trabajo se dan algunos pasos para aliviar este problema, con la definición de una estrategia para calcular eficientemente el número de soluciones de una vecindad cualquiera del VRP (capítulo 2), y una forma de particionar a la misma en grupos llamados “regiones” (capítulo 3). Estos conceptos serán utilizados en la construcción de un algoritmo de exploración en dos fases que se caracteriza por realizar un muestreo de la vecindad en cuestión (que consume muy poco tiempo), identificar las regiones más “prometedoras” de esta e intensificar la búsqueda en estas regiones. Este algoritmo se basa en los algoritmos híbridos de solo-exploración solo-explotación presentados en la siguiente sección.

1.5. Algoritmos híbridos de solo-exploración solo-explotación

Una novedosa línea de investigación en la optimización heurística es el desarrollo de híbridos de solo-exploración solo-explotación [16]. En tales híbridos los procesos de exploración y explotación del espacio de búsqueda están explícitamente separados.

El objetivo de estos algoritmos es mejorar la exploración del espacio de búsqueda evitando los efectos negativos de la exploración y explotación simultáneas en las primeras etapas de la búsqueda [27]. El principal desafío en este enfoque es el desarrollo de un método puramente exploratorio capaz de realizar una exploración imparcial del espacio de búsqueda.

Un problema importante al diseñar híbridos de solo-exploración solo-explotación es determinar cuándo debe ocurrir la transición entre la fase de exploración y explotación. En la literatura existen dos formas para determinar esta transición. La primera es establecer un punto de transición fijo entre las dos fases. La segunda y más novedosa, es la utilización de clasificadores (técnicas de aprendizaje de máquinas) que identifiquen automáticamente el punto de transición óptimo entre la exploración y explotación [18].

La idea clave detrás de los híbridos de solo-exploración solo-explotación es reconocer que ambos procesos se basan en diferentes estrategias de búsqueda [17], y que la ocurrencia simultánea de los mismos afecta la capacidad de realizar una exploración imparcial del espacio de búsqueda [27]. En el diseño de los híbridos que se proponen en este trabajo se usará una fase de solo-exploración para encontrar las "mejores" regiones en el espacio de búsqueda, seguida de una fase de solo-explotación en la cual se intensifica el análisis en estas regiones para encontrar los óptimos locales de las mismas.

En este trabajo, las regiones más "prometedoras" de una vecindad del VRP son estimadas con el uso de técnicas estadísticas, las que serán presentadas en la sección siguiente.

1.6. Técnicas estadísticas

El objetivo de este trabajo es utilizar técnicas estadísticas para estimar las regiones de la vecindad en las que es más probable que se encuentren las mejores soluciones. Para esto es necesario particionar la vecindad invo-

lucrada en regiones y con el uso de estas técnicas, detectar las más prometedoras. Las técnicas estadísticas utilizadas en este trabajo son el análisis de varianza, la estadística descriptiva y los árboles de decisión que se presentan a continuación.

El análisis de varianza [75, 50] (ANOVA, por sus siglas en inglés) es una herramienta estadística que se utiliza para detectar diferencias en las medias de grupos experimentales. ANOVA es una prueba paramétrica, lo que significa que asume ciertas características acerca de la distribución y los parámetros de la misma, en las poblaciones de los grupos que se deseen comparar. Para que los resultados de utilizar ANOVA tengan significación estadística, existen 3 supuestos que deben cumplir las poblaciones a las que pertenecen dichos grupos:

1. Independencia de las muestras.
2. Poblaciones con ditribuciones normales.
3. Poblaciones con varianzas iguales (homocedasticidad).

En el contexto de este trabajo estas poblaciones se refieren a las regiones de la vecindad, de las cuales se generan las muestras o grupos para estimar las medias. La idea es utilizar ANOVA para identificar aquellos grupos que sus medias tengan una diferencia significativa y tomar aquellas regiones a las que pertenecen los grupos de menor media como las mejores de la vecindad.

Además para estimar las regiones más prometedoras en una vecindad del VRP se emplea la estadística descriptiva, o el cálculo de la media de estos grupos. De esta forma se computa la media de cada una de las muestras de las regiones y se ordenan de menor a mayor media, siendo las de menor media las que se encuentran en las mejores regiones pues esto implica un menor costo de evaluación.

Por último, los árboles de decisión también se utilizan para identificar las mejores regiones de la vecindad. Estas técnicas son métodos de aprendizaje de máquinas para construir modelos de predicción de datos, que son obtenidos particionando recursivamente el espacio de datos y ajustando un modelo de predicción simple para cada partición. En la literatura existen varias implementaciones de estos métodos, entre las que se destacan: los Árboles de Clasificación y Regresión [23] (CART, por sus siglas en inglés), ID3 [70] y C4.5 [71] (el cual constituye una extensión de ID3). Además, de algunos estudios sobre el tema [64] [54] [77].

Estos métodos, según el tipo de la variable dependiente (variable que se desea predecir), se conocen en la literatura como: árboles de clasificación y árboles de regresión. Los árboles de clasificación son diseñados para variables dependientes que toman un número finitos de valores sin la existencia de un orden entre ellos, mientras que los árboles de regresión se utilizan para predecir variables que toman valores continuos o discretos con un orden definido. En este trabajo se utilizan los árboles de regresión, ya que se desea predecir el costo más probable que tienen las soluciones en una región de vecindad cualquiera de un VRP, que es una variable discreta ordenada. Para esto se utiliza una implementación del algoritmo CART.

En el problema de regresión, se tiene un conjunto de n elementos para una variable continua o discreta ordenada Y y p variables predictoras, $X_1, \ldots, X_p$ y el objetivo es poder predecir el valor de la variable Y para nuevos valores de X. En el contexto de este trabajo, el conjunto de elementos está formado por una muestra de las soluciones de una vecindad de un VRP, la variable a predecir es el costo de las mismas y las predictoras son los valores que toman las operaciones del criterio de vecindad para producir esa solución vecina.

El primer árbol de regresión publicado en la literatura se conoce como Detección Automática de Interacciones (AID, por sus siglas en inglés) [62]. Comenzando por el nodo raíz, AID particiona recursivamente los datos en cada nodo en dos nodos hijos. Para cualquier nodo t, sea $S(t)$ el conjunto de datos de entrenamiento en t y $\overline{y}_t$ la media de la muestra en t para la variable dependiente Y. Se puede definir la "impureza" del nodo t ($\phi(t)$) como la suma de la desviaciones cuadrado: $\phi(t) = \sum_{i \in S(t)} (y_i - \overline{y}_t)^2$. Este concepto refleja la presencia en un nodo de elementos que pertenecen realmente a clasificaciones distintas, luego la fórmula anterior tendrá valores altos cuando existan elementos de distintas clasificaciones y si esto no ocurre será cero. AID selecciona la partición que minimiza la suma de las impurezas en los dos nodos hijos. Este proceso de particionamiento termina cuando la reducción en la impureza para un nodo es menor que una cierta fracción de la impureza en el nodo raíz. El valor de la variable dependiente Y en cada hoja del árbol coincide con la media de los datos que pertenecen a ese nodo.

Los Árboles de Clasificación y Regresión (CART), son árboles binarios que se construyen a partir de particionar repetidamente el conjunto de datos en un nodo para formar dos nodos hijos en un proceso llamado "particionamiento recursivo". El mismo comienza por el nodo raíz del árbol,

que contiene todo el conjunto de los datos, y termina en las hojas cuando se cumple uno de los criterios de parada.

Este algoritmo, sigue el mismo enfoque goloso que AID para la selección de la partición que será efectuada en un nodo del árbol, ya que selecciona la que minimiza la suma de las impurezas en los dos nodos hijos. En CART se pueden utilizar dos medidas para la impureza de un nodo:

Error cuadrado: En este método las divisiones se seleccionan para minimizar la suma del error cuadrático entre los valores de la muestra que se encuentran en el nodo y la media de los mismos.

Desviación absoluta: Este método minimiza la desviación media absoluta de la mediana dentro de un nodo.

Algunos de los criterios que se utilizan para terminar el proceso de particionamiento en un nodo del árbol pueden ser: que el nodo se vuelva "puro" (o cuando su valor de impureza es 0), que el árbol alcance la máxima profundidad definida, si el número de datos en un nodo es menor que el mínimo permitido, o si la reducción en la impureza de un nodo no es menor que el valor especificado en el algoritmo después de aplicar la mejor división de los datos.

Sin embargo, también es posible no utilizar los criterios de parada y crecer el árbol completamente hasta que las hojas tengan una impureza mínima. El árbol resultante puede ser muy grande, por lo que se utilizan algunos enfoques para reducir este árbol a un tamaño óptimo, el cual puede ser determinado con el uso de otro conjunto independiente de datos o la validación cruzada [60], este proceso se conoce como "pruning".

En este trabajo se propone utilizar un árbol de regresión para estimar los valores de las operaciones del criterio de vecindad que resultan en las mejores soluciones vecinas, y a partir del agrupamiento de estas soluciones, atendiendo a dichos valores, poder identificar las regiones más prometedoras en una vecindad de un VRP.

Una vez definidos los elementos básicos de este trabajo, se puede pasar al siguiente capítulo, en el que se define una estrategia para calcular de forma eficiente la cardinalidad de una vecindad cualquiera del VRP.

Capítulo 2

Cálculo del número de soluciones vecinas

En este capítulo se presenta una estrategia para contar eficientemente el número de soluciones en una vecindad cualquiera del Problema de Enrutamiento de Vehículos. Esta cantidad se puede calcular a través de la propia definición de vecindad, aplicando exhaustivamente el criterio de vecindad sobre la solución actual, pero este enfoque es ineficiente pues es equivalente a explorar dicha vecindad. En el ámbito del presente trabajo, el cómputo de la cardinalidad de una vecindad del VRP se utiliza para poder decidir la estrategia de exploración más adecuada para la misma, y "eficiente" significa sin iterar sobre todas las soluciones vecinas, o sea, sin explorar la vecindad dos veces.

En la sección 2.1 se definen las soluciones y las operaciones necesarias para resolver el problema del conteo de vecinos, así como una clasificación de estas operaciones. En la sección 2.2 se presenta el algoritmo propuesto para el conteo de vecinos y en la sección 2.3 se define una estructura arbórea para almacenar toda la información necesaria de una vecindad del VRP, la cual se construye a partir del algoritmo presentado en la sección 2.2.

2.1. Solución y operaciones para el conteo de vecinos

Una vecindad es un concepto fundamental en un algoritmo de búsqueda local. La misma es el resultado de la aplicación exhaustiva del criterio de vecindad sobre la solución actual del algoritmo. La cardinalidad de una

vecindad se puede calcular haciendo uso de la definición anterior, lo cual sería equivalente a explorar dicha vecindad y por lo tanto, ineficiente.

En esta sección, se presentarán algunas modificaciones en la definición de la solución de un VRP, así como en las operaciones que conforman un criterio de vecindad con el objetivo de que estos conceptos reflejen las diferencias entre el problema de generar soluciones vecinas y el problema de contar las soluciones vecinas. Además, se definen tres tipos de clasificaciones para las operaciones que componen a un criterio de vecindad según la función de las mismas.

En su forma más simple, una solución del VRP se define como un conjunto de rutas, donde cada una de esas rutas está compuesta por una lista de clientes. Un ejemplo de solución para el VRP se puede encontrar en la figura 2.1.

```
R1: 1 2 3 4 5
R2: 6 7 8
R3: 9 10
```

Figura 2.1: Solución para un VRP con 10 clientes y 3 vehículos.

Para el propósito de contar la cantidad de soluciones en una vecindad del VRP, es irrelevante qué cliente se encuentra en una posición determinada de una ruta. Nótese la diferencia con la generación de las soluciones en una vecindad, donde sí es necesario identificar a cada cliente en las rutas. Por lo tanto, para calcular la cardinalidad de la vecindad solo es necesario conocer la cantidad de clientes en cada ruta. Esta simplificación de una solución del VRP, recibe el nombre de solución de conteo. En la figura 2.2 se puede ver un ejemplo de la misma para la solución de la figura 2.1.

```
R: 5 3 2
```

Figura 2.2: Solución de conteo para un VRP con 10 clientes y 3 vehículos.

En la figura 2.2 se puede apreciar una solución de conteo para el VRP, la cual se define como un conjunto de rutas, donde cada una se representa por la cantidad de clientes en las mismas. Este cambio en la representación de la solución hace necesario la modificación de los criterios de vecindad presentados en la sección 1.3 para una solución del VRP.

Un criterio de vecindad es una lista de operaciones de modificación que se aplican a una solución del VRP [63]. Al definirse una nueva solución para el conteo del número de vecinos en una vecindad del VRP, es necesario

modificar además las operaciones de vecindad definidas en la sección 1.3 para que las mismas actúen como operaciones de modificación válidas sobre la solución de conteo.

En este sentido, todas las operaciones excepto la selección de una ruta y la inversión del orden de los clientes en una subruta tienen la funcionalidad de incrementar o decrementar el número de clientes que se encuentran en una ruta. En el caso de la operación de selección de una ruta, esta mantiene su función original. Mientras que la operación de inversión del orden de los clientes en una subruta, no influye en el proceso de conteo de las soluciones vecinas pues la misma no tiene efecto alguno sobre la solución de conteo ni otras operaciones del criterio.

En la figura 2.3 se muestra el criterio de vecindad mover a un cliente de ruta, y en la figura 2.4 se presenta la solución resultado de aplicar dicho criterio con $r1 = 1$ y $r2 = 2$, a la solución de conteo de la figura 2.2. Se puede ver cómo al seleccionar un cliente de $r1$ `(ruta 1)` el número de clientes de la misma se decrementó en 1, y al insertar el mismo cliente en $r2$ `(ruta 2)` la cantidad de clientes en la misma se incrementó en 1.

```
select route r1
select client c1 from route r1
select route r2
insert client c1 into route r2
```

Figura 2.3: Mover a un cliente de ruta.

```
R: 4 4 2
```

Figura 2.4: Solución de conteo para la solución en la figura 2.1.

En las figuras 2.3 y 2.5, se pueden apreciar dos ejemplos de criterios de vecindad para la solución del VRP. Las operaciones que componen a los mismos tienen un conjunto de propiedades cuyos valores determinan el resultado de la aplicaión de dicha operación sobre la solución. Cada operación de vecindad tiene sus propiedades particulares. Por ejemplo: la operación de selección de una ruta tiene como propiedad el identificador de la ruta seleccionada; la operación de selección de un cliente almacena la ruta y la posición dentro de la misma correspondientes al cliente seleccionado; la selección de una subruta tiene como propiedades la ruta y la posición dentro de la misma correspondiente al primer cliente de la subruta seleccionada, así como su longitud; la inserción de una subruta debe tener en

cuenta la ruta y la posición dentro de la misma donde la subruta será insertada, así como la longitud de dicha subruta.

Estas operaciones se diferencian en la función que las mismas realizan en el proceso de modificación de la solución actual para la generación de una solución vecina. Según su función, las operaciones que se utilizan en el presente trabajo pueden ser divididas en tres clasificaciones: operaciones principales, operaciones modificadoras y operaciones pasivas. A continuación se presentan estas clasificaciones.

Las **operaciones principales**, son aquellas operaciones que definen el valor de ciertas propiedades comunes para varias operaciones del criterio. Por ejemplo, si se analiza el criterio de vecindad mostrado en la figura 2.3 se puede apreciar que la primera operación de selección de ruta define completamente el valor de la ruta $r1$, la cual además es una propiedad común con la operación de selección de cliente que le sigue en dicho criterio. De igual manera ocurre con la segunda operación de selección de ruta y la operación de inserción de cliente que le sucede. Por todo lo anterior, se puede decir que las selecciones de rutas son operaciones principales.

```
select route r1
select subroute s1 from route r1
select route r2
insert subroute s1 into route r2
```

Figura 2.5: Mover a una subruta de ruta.

De forma similar, al analizar el criterio de vecindad en la figura 2.5 y la solución de la figura 2.2 para los valores $r1 = 1$ y $r2 = 2$. Si se selecciona la subruta $s1$ tal que su longitud es igual a 1, entonces las solución resultante se encuentra definida en la figura 2.4. Sin embargo, si se selecciona la subruta $s1$ tal que su longitud es igual a 2, entonces la solución resultante estaría definida en la figura 2.6. En ambos casos, la única diferencia estuvo en la longitud de la subruta seleccionada lo que determinó la cantidad de clientes a incrementar por la operación de inserción de subruta en la ruta $r2$. Por lo tanto, la operaciones de selección de subrutas también son operaciones principales. Sin embargo, en esta operación no basta con definir la longitud de la subruta seleccionada, además es necesario decrementar dicha cantidad al número de clientes en la ruta en cuestión, lo que hace que la selección de subrutas pertenezca además a la clasificación siguiente.

Las **operaciones modificadoras**, son aquellas operaciones que se encargan de modificar la solución actual. En este tipo de operaciones, se pueden

```
R: 3 5 2
```

Figura 2.6: Solución de conteo para la solución en la figura 2.1.

encontrar las selecciones, inserciones e intercambios de clientes y subrutas. En el caso de las selecciones, uno o más clientes se eliminan de la ruta en cuestión. En el caso de las inserciones, uno o más clientes se añaden. Y finalmente, las operaciones de intercambio donde uno o más clientes se añaden a las rutas involucradas.

Las **operaciones pasivas**, son aquellas operaciones que no tienen efecto alguno sobre otras operaciones o la solución actual. En esta clasificación, se encuentra la operación de invertir el orden de los clientes de una subruta, pues la misma solo tiene efecto sobre una subruta. En esta categoría también se encuentra la operación cambiar vehículo de ruta, en un problema con flota heterogénea, porque no influye en la solución de conteo.

De las clasificaciones anteriores, resulta necesario destacar que las mismas no son excluyentes. Esto significa que una operación puede pertenecer a varias de estas clasificaciones, como es el caso de la operación seleccionar una subruta que pertenece tanto a las operaciones principales como a las modificadoras. Y es que en dicha operación se toman dos decisiones relacionadas, pero de naturaleza diferente: determinar las posibles longitudes de las subrutas y por cada una de estas, decrementar dicha cantidad al número de clientes en la ruta en cuestión. La primera hace que la selección de una ruta sea una operación principal, mientras que la segunda la convierte en una operación modificadora.

En esta sección se dieron los primeros pasos y definieron las ideas sobre la cual se basa la estrategia de conteo que se definirá en la siguiente.

2.2. Una estrategia para el conteo de vecinos

Una de las estapas más importantes de un algoritmo de búsqueda local es la exploración de la vecindad de la solución actual. Una vecindad del VRP está determinada por el criterio de vecindad y la solución inicial. Sin embargo, una vecindad del VRP posee otras características como: el número de soluciones en la misma, el número de soluciones que son mejores que la solución que generó la vecindad, el óptimo de la vecindad, la cantidad de óptimos en la vecindad, entre otras.

Un previo conocimiento de la cardinalidad de la vecindad que se desea explorar permite la selección de una estrategia de exploración [40] que se ajuste en mayor medida a la vecindad en cuestión. Por lo tanto, en esta sección se presenta una estrategia para contar eficientemente el número exacto de soluciones en una vecindad cualquiera del VRP. En este contexto eficiente significa sin iterar sobre todas las soluciones vecinas.

Para comprender los principios básicos de esta estrategia habría que conocer primeramente cómo se utiliza el criterio de vecindad para generar las soluciones vecinas de una solución dada. En el marco del presente trabajo, una solución vecina se encuentra representada por una lista con las operaciones de vecindad del criterio, en el orden en el que aparecen en el mismo, donde cada uno de los valores que pueden tomar cada operación se encuentran fijados.

Lo anterior significa que existe una correspondencia tal que a cada asignación válida en los valores de las operaciones de un criterio se le asocia una solución vecina, la cual es el resultado de aplicar las operaciones de modificación con dichos valores sobre la solución dada. Por ejemplo, si se aplica el criterio de vecindad de la figura 2.3 a la solución en la figura 2.1 para la siguiente asignación en los valores de las operaciones la solución vecina se encuentra representada en la figura 2.7.

- $r_1 = 1$: La ruta r_1 de la cual el cliente c_1 será extraído es la primera ruta de la solución dada.
- $c_1.selection\_pos = 1$: El cliente c_1 es el primer cliente de la ruta r_1.
- $r_2 = 2$: La ruta r_1 en la cual el cliente c_2 será insertado es la segunda ruta de la solución dada.
- $c_1.insertion\_pos = 1$: El cliente c_1 será insertado como el primer cliente de la ruta r_2.

```
R1: 2 3 4 5
R2: 1 6 7 8
R3: 9 10
```

Figura 2.7: Solución para un VRP con 10 clientes y 3 vehículos.

En este trabajo, una asignación de los valores de una operación de vecindad será llamado una instanciación de la misma, y una lista de instanciaciones de todas las operaciones de un criterio de vecindad constituye la

instanciación de un criterio. Como ya se mencionó anteriormente, a cada instanciación válida de un criterio le corresponde una solución vecina a la solución dada sobre la cual se aplica dicho criterio de vecindad. En este sentido el algoritmo definido en esta sección se encarga de contar el total de posibles instanciaciones de un criterio de vecindad para una solución del VRP dada.

Al generar las distintas instanciaciones de un criterio de vecindad para una solución dada se cumple que muchas de estas comparten una secuencia común de operaciones instanciadas. La estrategia propuesta se basa en agrupar aquellos criterios instanciados para los cuales dicha secuencia común comience en la primera operación de los mismos, pues de esta forma el resto de las operaciones de tales criterios instanciados no se ven afectadas, y contar para cada una de estas el número de posibles secuencias de operaciones instanciadas que unidas con la secuencia común forman un criterio instanciado.

Para contar la cantidad de posibles secuencias de operaciones instanciadas distintas que existen a partir de una solución del VRP y una lista de operaciones de vecindad, cada operación debe ser capaz de definir una forma de calcular y combinar el conjunto de sus posibles instanciaciones con las del resto de operaciones de la lista. La forma en la que se realiza dicho cálculo y combinación varía según la operación en cuestión, pero existe una estructura o pasos generales los cuales son comunes a todas las operaciones, como lo son:

1. Definir el conjunto de posibles instanciaciones para esta operación.
2. Actualizar el estado de la solución.
3. Combinar las posibles instanciaciones para esta solución con el total de secuencias de operaciones instanciadas que se pueden formar con el resto de las operaciones.

Antes de definir las funciones de conteo para las operaciones de vecindad, es necesario mencionar que todas las instanciaciones de las operaciones modificadoras alteran a la solución de conteo en la misma forma (en el caso de la selección de una subruta, se refiere a la instanciación de la propiedad que representa la posición del primer cliente de la misma y no de su longitud), por lo que solo es necesario computar el total de secuencias de instanciaciones del resto de las operaciones para una de ellas. Esto significa que si se extrae un cliente de la ruta 1 de la solución no importa la

posición de la cual el mismo fue extraído, la única modificación será el decrementar en 1 la cantidad de clientes para dicha ruta. Esta es la verdadera simplificación por la cual la solución de conteo y las operaciones para el conteo fueron definidas en la sección anterior y constituye la base para la eficiencia en el algoritmo propuesto.

En la figura 2.8 se presenta la función de conteo para la operación de selección de un cliente. Esta es una operación modificadora, por lo que es solamente necesario computar el total de secuencias de operaciones instanciadas que se pueden formar si se extrae un cliente en dicha ruta de la solución, y se combina multiplicándolo con el número de posibilidades de instanciación. Esta función tiene un orden constante.

```
COUNT_NEIGHBORS(op,sol,other_ops)
1 posibilities = op.route.len
2 sol.routes[op.route.id] -= 1
3 next_op = other_ops[0]
4 rest_ops = other_ops[1:]
5 count = COUNT_NEIGHBORS(next_op,sol,rest_ops)
6 sol.routes[op.route.id] += 1
7 return possibilities * count
```

Figura 2.8: Función para el conteo en la operación de selección de un cliente.

En la figura 2.9 se presenta la función de conteo para la operación de inserción de un cliente. Esta función es prácticamente idéntica a la anterior, la única diferencia es que las posibles instanciaciones se encuentran determinadas por el número de clientes en la ruta más 1, ya que se puede insertar delante de cada cliente y detrás del último.

En la figura 2.10 se presenta la función de conteo para la operación de intercambio de clientes. Existe solo una instanciación posible para esta operación, luego esta computa el total de secuencias de operaciones instanciadas si se inserta un cliente en cada una de las rutas involucradas. Esta función tiene un orden constante.

En la figura 2.11 se define la función para el conteo en la operación de selección de una ruta. Se cumple que cada una de las distintas rutas a seleccionar de la solución pueden afectar de forma diferente al resto de las operaciones, y por lo tanto hay que contar el total de secuencias de opera-

```
COUNT_NEIGHBORS(op,sol,other_ops)
1 posibilities = op.route.len + 1
2 sol.routes[op.route.id] += 1
3 next_op = other_ops[0]
4 rest_ops = other_ops[1:]
5 count = COUNT_NEIGHBORS(next_op,sol,rest_ops)
6 sol.routes[op.route.id] -= 1
7 return possibilities * count
```

Figura 2.9: Función para el conteo en la operación de inserción de un cliente.

```
COUNT_NEIGHBORS(op,sol,other_ops)
1 sol.routes[op.route1.id] += 1
2 sol.routes[op.route1.id] += 1
3 next_op = other_ops[0]
4 rest-ops = other-ops[1:]
5 count = COUNT_NEIGHBORS(next_op,sol,rest_ops)
6 sol.routes[op.route1.id] -= 1
7 sol.routes[op.route2.id] -= 1
8 return count
```

Figura 2.10: Función para el conteo en la operación de intercambio de clientes.

ciones instanciadas que se pueden formar por cada posible ruta seleccionada de la solución del VRP. Estos totales se combinan mediante la suma de los mismos, y se puede decir que la complejidad temporal de esta función depende linealmente de la cantidad de rutas en la solución.

En la figura 2.12 se define la función de conteo para la operación de selección de una subruta. Se cumple que cada longitud de subruta seleccionada puede afectar al resto de las operaciones de forma diferente. Luego, es necesario computar el total de secuencias de operaciones instanciadas que se pueden formar por cada longitud de subruta seleccionada. Además, esta es una operación modificadora con respecto a la extracción de los clientes que componen la subruta, por lo que estos totales se combinan multiplicando cada uno de ellos con el número de posibles posiciones donde puede comenzar la subruta con el tamaño fijado y sumando estos productos. Esta

```
COUNT_NEIGHBORS(op,sol,other_ops)
1 count = 0
2 for op.route.id from 1 to sol.routes.len:
3   next_op = other_ops[0]
4   rest_ops = other_ops[1:]
5   count += COUNT_NEIGHBORS(next_op,sol,rest_ops)
6 return count
```

Figura 2.11: Función para el conteo en la operación de selección de ruta.

función depende linealmente del número de clientes en la ruta en cuestión, pues la longitud de la subruta se encuentra acotado superiormente por dicho número.

```
COUNT_NEIGHBORS(op,sol,other_ops)
1 total_count = 0
2 for op.subr.len from 1
                  to sol.routes[op.route.id]-1:
3   next_op = other_ops[0]
4   rest_ops = other_ops[1:]
5   possibilities = sol.routes[op.route.id] -
                    op.subr.len + 1
6   sol.routes[op.route.id] -= op.subr.len
7   count = COUNT_NEIGHBORS(next_op,sol,rest_ops)
8   total_count += possibilitites * count
9   sol.routes[op.route.id] += op.subr.len
10 return total_count
```

Figura 2.12: Función para el conteo en la operación de selección de una subruta.

En la figura 2.13 se define la función de conteo para la operación de inserción de una subruta. Esta función es prácticamente idéntica a la definida anteriormente para la inserción de un cliente, la única diferencia es que la cantidad de clientes a insertar en la ruta está determinada por la longitud de la subruta.

En la figura 2.14 se define la función de conteo para la operación de intercambio de dos subrutas. Esta función es idéntica a la definida para el

```
COUNT_NEIGHBORS(op,sol,other_ops)
1 posibilities = op.route.len + 1
2 sol.routes[op.route.id] += op.subr.len
3 next_op = other_ops[0]
4 rest_ops = other_ops[1:]
5 count = COUNT_NEIGHBORS(next_op,sol,rest_ops)
6 sol.routes[op.route.id] -= op.subr.len
7 return possibilities * count
```

Figura 2.13: Función para el conteo en la operación de inserción de una subruta.

intercambio de dos clientes, la diferencia radica en que la cantidad de clientes a insertar en las rutas involucradas está determinada por la longitud de las subrutas.

```
COUNT_NEIGHBORS(op,sol,other_ops)
1 sol.routes[op.route1.id] += op.subr1.len
2 sol.routes[op.route1.id] += op.subr2.len
3 next_op = other_ops[0]
4 rest_ops = other_ops[1:]
5 count = COUNT_NEIGHBORS(next_op,sol,rest_ops)
6 sol.routes[op.route1.id] -= op.subr2.len
7 sol.routes[op.route2.id] -= op.subr1.len
8 return count
```

Figura 2.14: Función para el conteo en la operación de intercambio de subrutas.

En este documento no se definirá la función de conteo para la operación que invierte el orden de los clientes dentro de una subruta, pues la misma no afecta a la solución o alguna otra operación. Esto significa que para el objetivo de contar la cardinalidad de una vecindad del VRP es posible eliminar dicha operación del criterio de vecindad, si es que está presente.

Aprovechando la definición recursiva del algoritmo anterior, se puede definir una estructura arbórea que almacene y refleje la ejecución de dicho algoritmo. Esta estructura recibe el nombre de árbol de vecindad y se definirá con más detalle en la próxima sección.

2.3. Árbol de Vecindad

Al computar la cardinalidad de una vecindad del VRP, se utiliza una estrategia recursiva que consiste en contar para una operación el número de secuencias de operaciones instanciadas que se pueden formar con el resto de las operaciones del criterio, así como combinar las mismas con las posibles instanciaciones de la operación actual aprovechando que para las operaciones modificadoras estas secuencias son comunes a todas las posibles instanciaciones de las mismas.

Es posible utilizar este algoritmo, para almacenar toda la información necesaria para cualquier procesamiento posterior sobre dicha vecindad en una estructura arbórea que será llamada árbol de vecindad y que constituye una representación de la vecindad en cuestión. En esta sección se definirá formalmente dicho árbol de vecindad, así como los nodos que lo forman.

Como cualquier estructura arbórea, el árbol de vecindad está compuesto por nodos. En su forma clásica, estos tienen la función de almacenar cierta información y de mantener la estructura conectada. Además, cada nodo de este árbol representa una operación de vecindad. En el presente trabajo existen 8 tipos de nodos que forman este árbol, cada uno de ellos corresponde con una operación de vecindad. Si se desea extender esta idea a otros problemas en los que existan otras operaciones para la vecindad, solo habría que agregar los nuevos nodos.

A continuación se definirá para cada nodo la operación de vecindad que representa, la información que almacena y la función del mismo. Cada uno de estos nodos almacena el total de soluciones que hay en el subárbol del cual es raíz y una referencia a su padre, por lo que esta información no se mencionará a continuación:

R-Nodo: Este nodo representa una operación de selección de ruta. El mismo almacena una lista con una referencia a sus hijos por cada posible ruta seleccionada de la solución. Este es el único nodo que puede funcionar como la raíz de este árbol, además puede ser un nodo interior del árbol pero no una hoja, pues todos los criterios de vecindad deben comenzar con una selección de ruta y no pueden terminar con dicha operación [63].

A-Nodo: Este nodo representa una operación de selección de cliente. El mismo almacena una referencia a su hijo en el árbol, el número de la ruta de la cual se seleccionará el cliente y la cantidad de clientes

presentes en dicha ruta. Este nodo sólo puede funcionar como nodo interior del árbol de vecindad, porque la selección de un cliente no puede ser la última operación de un criterio de vecindad [63].

B-Nodo: Este nodo representa una operación de inserción de cliente. El mismo almacena una referencia a su hijo en el árbol, el número de la ruta en la cual el cliente será insertado, el número de posiciones en las que el cliente puede ser insertado en dicha ruta y el identificador de la operación de selección de cliente que corresponde con el cliente que será insertado. Este nodo puede funcionar tanto como un nodo interior, o como una hoja del árbol de vecindad.

C-Nodo: Este nodo representa una operación de intercambio de clientes. El mismo almacena una referencia a su hijo en el árbol, así como los dos identificadores de las operaciones de selección de cliente que corresponden con los clientes a ser intercambiados. Este nodo puede funcionar tanto como un nodo interior, o como una hoja del árbol de vecindad.

E-Nodo: Este nodo representa una operación de selección de subruta. El mismo almacena una lista con una referencia a sus hijos por cada posible longitud de subruta y el número de la ruta de la cual la subruta será seleccionada. Este nodo sólo puede funcionar como un nodo interior en el árbol de vecindad, porque la selección de una subruta no puede ser la última operación de un criterio de vecindad [63].

F-Nodo: Este nodo representa una operación de inserción de subruta. El mismo almacena una referencia a su hijo en el árbol, el número de la ruta en la cual la subruta será insertada, la cantidad de posiciones en las que la subruta puede ser insertada en dicha subruta y el identificador de la operación de selección de subruta que corresponde con la subruta a ser insertada. Este nodo puede funcionar tanto como un nodo interior, o como una hoja del árbol de vecindad.

G-Nodo: Este nodo representa una operación de intercambio de subrutas. El mismo, almacena una referencia a su hijo en el árbol, así como los dos identificadores de las operaciones de selección de subruta que corresponden con las subrutas a ser intercambiadas. Este nodo, puede funcionar tanto como un nodo interior, o como una hoja del árbol de vecindad.

H-Nodo: Este nodo representa una operación de inversión en el orden de los clientes en una ruta. El mismo almacena una referencia a su hijo en el árbol, así como el identificador da la operación de selección de subruta en la cual el orden de los clientes será invertido. Este nodo puede funcionar tanto como un nodo interior, o como una hoja del árbol de vecindad.

A partir de un criterio de vecindad y una solución para el VRP, se define una única vecindad. El criterio de vecindad de la figura 2.15 representa el movimiento de un cliente dentro de su propia ruta, si se aplica este criterio sobre la solución de la figura 2.2 el árbol que representa dicha vecindad se muestra en la figura 2.16.

```
select route r1
select client c1 from route r1
insert client c1 into route r1
```

Figura 2.15: Mover a un cliente dentro de su ruta.

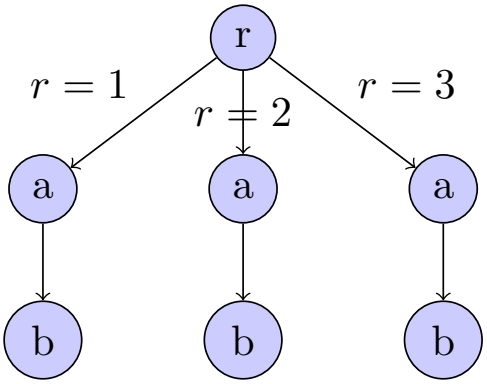

Figura 2.16: Árbol de vecindad

Este árbol de vecindad, es un reflejo exacto de la ejecución del algoritmo propuesto en la sección anterior para contar eficientemente el número de soluciones en una vecindad del VRP. Cada nodo del árbol, representa una operación de vecindad y almacena toda la información necesaria para instanciar dicha operación y de esta forma, a partir del proceso que computa la cardinalidad de la vecindad, se pueden pasar a generar soluciones específicas de la misma.

Como se puede apreciar en la figura 2.16, la altura de este árbol coincide con el número de operaciones en el criterio de vecindad. Debido a que cada nivel del mismo representa una operación del criterio con valores de instanciación diferentes, o al menos relacionados con operaciones anteriores en el criterio de vecindad con valores de instanciación diferentes. El número de hojas de este árbol, es un valor más difícil de computar ya que el mismo depende del criterio de vecindad y de la solución en cuestión. Además, este número de hojas coincide con la cantidad de ramas del árbol de vecindad.

En el árbol de vecindad mostrado en la figura 2.16 se puede apreciar la existencia de 3 ramas. La primera de ellas se encarga de contar y generar las posibles instanciaciones de criterios de vecindad donde el cliente se selecciona e inserta en la ruta número 1 de la solución, la segunda rama se encarga de las instanciaciones para la ruta número 2 de la solución y la tercera rama para la ruta número 3 de la solución. Al existir una correspondencia entre una instanciación de un criterio de vecindad para una solución dada y una solución vecina, se puede decir que cada rama del árbol de vecindad representa un conjunto de soluciones en la vecindad, o sea el conjunto de ramas de este árbol constituyen una partición de la vecindad que dicho árbol representa, idea sobre la cual se profundizará en el próximo capítulo.

En esta sección se presentó el árbol de vecindad, así como algunas de las propiedades y elementos del mismo. En este árbol se encuentra almacenada toda la información necesaria para realizar procesamientos más complejos sobre la vecindad que este representa, como: indizar en la misma, generar las soluciones vecinas, explorar y particionar esta vecindad, entre otras aplicaciones que se presentarán en este trabajo.

Capítulo 3

Otros análisis sobre una Vecindad del VRP

En este capítulo se presentan conceptos fundamentales para el diseño y el funcionamiento de la herramienta propuesta. En la sección 3.1 se introduce una estrategia para definir un orden entre la soluciones de una vecindad cualquiera de un Problema de Enrutamiento de Vehículos. En la sección 3.2 se define el concepto de región de una vecindad de un VRP y una forma de particionar a dicha vecindad en regiones. Por último, en la sección 3.3 se presenta otra función para definir un orden en la vecindad la cual resulta más conveniente para los análisis que involucren el trabajo con las regiones.

3.1. Una estrategia de indexación clásica

Una vecindad del Problema de Enrutamiento de Vehículos, constituye un conjunto de soluciones del mismo que se encuentran relacionadas con otra solución dada mediante un criterio de vecindad. La existencia de un orden bien definido entre las soluciones de una vecindad del VRP resulta conveniente para el análisis de la misma, pues dicho orden permite la simplificación de la representación de las soluciones vecinas asociando las mismas con el índice que le corresponde en el ordenamiento ascendente de las soluciones de la vecindad.

En esta sección, se presenta un orden para las soluciones de una vecindad del VRP. Dicho orden se define mediante una biyección entre el con-

junto de soluciones en la vecindad y el conjunto de los números naturales positivos. A dicho orden se le llamará función de indexación.

Como ya fue mencionado en la sección 2.2, a cada solución de una vecindad de un VRP le corresponde una instanciación del criterio que generó dicha vecindad. Este criterio está compuesto por un conjunto de operaciones, también instanciadas, o sea, cada una de las operaciones que lo forman tienen valores asignados para todas sus propiedades. Estas propiedades de las operaciones de vecindad fueron definidas en las sección 2.1, y cada una de estas operaciones se caracteriza por una de sus propiedades (o dos, en el caso de la selección de una subruta) para la definición de la función de indexación. A continuación se muestra por cada operación la propiedad (o valor) que la caracterizan:

Select route: Esta operación se caracteriza por el número de la ruta seleccionada.

Select client: Esta operación se caracteriza por la posición de la cual se selecciona el cliente.

Insert client: Esta operación se caracteriza por la posición en la cual se inserta el cliente.

Swap clients: Esta operación no participa en la definición de la función de orden, pues la misma se encuentra completamente determinada por las selecciones de los clientes involucrados.

Select subroute: Esta operación se caracteriza por dos valores, la longitud de la subruta y la posición del primer cliente en la misma. Para el propósito del orden, la longitud de la subruta tiene propiedad sobre el valor de la posición del primer cliente en la misma.

Insert subroute: Esta operación se caracteriza por la posición en la cual se inserta la subruta.

Swap subroutes: Esta operación no participa en la definición de la función de orden, pues la misma se encuentra completamente determinada por las selecciones de las subrutas.

Reverse subroute: Esta operación no participa en la definición de la función de orden, pues la misma no tiene relación alguna con el número de soluciones en la vecindad.

```
select route r1
select client c1 from route r1
select route r2
insert client c1 into route r2
```

Figura 3.1: Mover a un cliente de ruta.

```
R1: 1 2 3 4 5
R2: 6 7 8
R3: 9 10
```

Figura 3.2: Solución para un VRP con 10 clientes y 3 vehículos.

En este sentido el primer vecino en la vecindad se define como aquella solución para la cual el criterio instanciado que la representa tiene el menor valor posible de todas las operaciones del criterio. Por ejemplo, si se usa el criterio de vecindad en la figura 3.1, el cual representa el movimiento de un cliente de ruta, para la solución en la figura 3.2, el primer vecino en la vecindad se encuentra representado por el criterio anterior instanciado para los valores que aparecen en la figura 3.4. Esta solución se muestra en la figura 3.3.

```
R1: 1 2 3 4 5
R2: 6 7 8
R3: 9 10
```

Figura 3.3: Primera solución vecina.

$\{R_1 : 1, A : 1, R_2 : 1, B : 1\}$

Figura 3.4: Instanciación del criterio en la figura 3.1 para el primer vecino.

Como se puede apreciar, la solución en la figura 3.3 coincide con la solución a partir de la cual se generó (la solución en la figura 3.2). Esto se debe a que esta solución es el resultado de seleccionar un cliente en la primera posición de la ruta número 1 de la solución e insertar dicho cliente en la misma posición de tal ruta.

La segunda solución en este orden de la vecindad, es aquella a la cual le corresponde el criterio instanciado resultado de incrementar en uno el valor de la última operación de la instanciación anterior. Para la vecindad anterior, la instanciación de dicho criterio se muestra en la figura 3.5 y la solución resultado de la aplicación de este sobre la solución actual se presenta

en la figura 3.6.

$\{R_1 : 1, A : 1, R_2 : 1, B : 2\}$

Figura 3.5: Instanciación del criterio en la figura 3.1 para el segundo vecino.

```
R1: 2 1 3 4 5
R2: 6 7 8
R3: 9 10
```

Figura 3.6: Segunda solución vecina.

Siguiendo dicho patrón, la próxima solución en este orden estará representada por el criterio instanciado resultado de incrementar en uno el valor de la última operación de la instanciación del criterio correspondiente a la solución anterior. Cuando dicha operación toma su máximo valor, vuelve a tomar su valor inicial, y la operación anterior a esa se incrementa.

Esto se evidencia para la solución representada por los valores de las operaciones en la figura 3.7, la misma es la quinta solución en la vecindad y para calcular el siguiente vecino ocurre que el valor de la última operación alcanzó su valor máximo, por lo que su valor pasará a ser 1, y el número de la ruta en la segunda operación de selección de subruta se incrementará en uno. Los valores de las operaciones para la sexta solución en la vecindad se muestran en la figura 3.8.

$\{R_1 : 1, A : 1, R_2 : 1, B : 5\}$

Figura 3.7: Instanciación del criterio en la figura 3.1 para el quinto vecino.

$\{R_1 : 1, A : 1, R_2 : 2, B : 1\}$

Figura 3.8: Instanciación del criterio en la figura 3.1 para el sexto vecino.

Es necesario aclarar que los valores de las operaciones intermedias del criterio también alcanzarán su máximo valor, de igual forma que se definió para la última operación del criterio, esta operación vuelve a tomar su valor inicial, y la operación anterior a esa incrementa su valor.

Un ejemplo de esto lo constituye la solución que ocupa el índice 12 en la vecindad anterior, representada por los valores de las operaciones en la figura 3.9. Se cumple que los valores de las dos últimas operaciones han alcanzado su máximo valor, por lo cual el siguiente vecino estará representado por los valores instanciados de la figura 3.10.

$\{R_1 : 1, A : 1, R_2 : 3, B : 3\}$

Figura 3.9: Instanciación del criterio en la figura 3.1 para el duodécimo vecino.

$\{R_1 : 1, A : 2, R_2 : 1, B : 1\}$

Figura 3.10: Instanciación del criterio en la figura 3.1 para el décimo tercer vecino.

Este proceso se repite hasta que todas las operaciones tengan su último valor posible, y ese sería el último vecino. En el caso del ejemplo, sería la solución en la vecindad asociada a los valores de las operaciones que se muestran en la figura 3.11.

$\{R_1 : 3, A : 2, R_2 : 3, B : 2\}$

Figura 3.11: Instanciación del criterio en la figura 3.1 para el último vecino.

En esta sección se definió una forma de definir un orden entre las soluciones de una vecindad de un VRP, en la siguiente se presenta una manera de particionar una vecindad en grupos de soluciones con ciertas características similares llamados “regiones”.

3.2. Particionando una Vecindad del VRP

En el análisis de una vecindad resulta conveniente agrupar aquellas soluciones que comparten ciertas características similares e intentar identificar algunas de las propiedades que pueden cumplir dichos grupos. Por ejemplo, en el proceso de exploración de una vecindad puede resultar interesante identificar las características de los grupos que contienen las mejores soluciones dentro de la vecindad, para focalizar la exploración en estas soluciones. A tales grupos de soluciones dentro de la vecindad se les llamará regiones y en esta sección se presenta una forma de particionar una vecindad cualquiera del VRP en las mismas.

En la figura 3.12 se muestra un criterio de vecindad que representa el intercambio de dos clientes, mientras en la figura 3.13 se define una solución para el VRP en su forma más básica.

Al aplicar el criterio de la figura 3.12 sobre la solución en la figura 3.13,

```
select route r1
select client c1 from route r1
select route r2
select client c2 from route r2
swap clientes c1 y c2
```

Figura 3.12: Intercambio de dos clientes.

```
R1: 1 2 3 4 5
R2: 6 7 8
R3: 9 10
```

Figura 3.13: Solución actual.

si se selecciona el primer cliente de la primera ruta y el primer cliente de la segunda ruta se obtiene la solución en la figura 3.14.

```
R1: 6 2 3 4 5
R2: 1 7 8
R3: 9 10
```

Figura 3.14: Solución vecina 1.

Si se selecciona el primer cliente de la primera ruta y el segundo cliente de la segunda ruta se obtiene la solución en la figura 3.15.

```
R1: 7 2 3 4 5
R2: 6 1 8
R3: 9 10
```

Figura 3.15: Solución vecina 2.

Finalmente, si se selecciona el primer cliente de la primera ruta y el primer cliente de la tercera ruta se obtiene la solución en la figura 3.16.

Se puede notar que para las soluciones en las figuras 3.14 y 3.15, el primer cliente se seleccionó de la ruta 1 y el segundo cliente de la ruta 2. No así para la solución en la figura 3.16 donde a pesar de seleccionar el primer cliente de la ruta 1, el segundo cliente se seleccionó de la ruta 3.

En el capítulo anterior se mencionó que cada solución en la vecindad puede ser representada por el conjunto de valores de las operaciones del criterio de vecindad que la generaron. En este sentido se puede particionar una vecindad del VRP en clases de equivalencia, donde una clase de equi-

```
R1: 9 2 3 4 5
R2: 6 7 8
R3: 1 10
```

Figura 3.16: Solución vecina 3.

valencia es el conjunto de soluciones de la vecindad que poseen valores iguales para las operaciones principales del criterio.

Luego, como la solución en la figura 3.16 difiere en el valor de la segunda operación de selección de ruta con respecto a las soluciones 3.14 y 3.15, se cumple que estas últimas se encuentran en una misma región a la cual la primera no pertenece.

En la vecindad generada por el criterio y la solución anterior hay un total de 9 regiones, ya que la solución tiene un total de 3 rutas y en el criterio existen 2 operaciones principales, en este caso de selección de rutas. El número de regiones en una vecindad del VRP depende del criterio de vecindad y de la solución actual en cuestión.

En este trabajo se definieron dos operaciones principales en la sección 2.1, estas operaciones son las selecciones de rutas y subrutas. La definición brindada anteriormente sobre una región de una vecindad presenta una limitación para el trabajo con las selecciones de subrutas, y esta es la existencia para algunas vecindades de un VRP de una alta cantidad de regiones en las mismas, haciendo igualmente costoso el análisis de las regiones. En el contexto de este trabajo el número de regiones de una vecindad se compara con la raíz cuadrada de la cardinalidad de la misma, considerándose esta raíz cuadrada como la cantidad idónea de grupos en los que debe ser particionada una vecindad de un VRP.

Por ejemplo, si se considera el criterio de vecindad de la figura 3.17, el cual representa el cambio de ruta de una subruta, para la solución de la figura 3.13, la vecindad generada tiene un total de 263 soluciones y 30 regiones. El número de regiones para esta vecindad es casi el doble de la raíz cuadrada de la cardinalidad de la misma, por lo que se puede decir que es muy alto.

Este alto número de regiones dentro de una vecindad de un VRP, se debe a la existencia de una región por cada posible longitud de subruta seleccionada. En ese sentido se propone la modificación de la definición de región anterior, y en lugar de dos soluciones pertenecer a regiones diferentes si las longitudes de las subrutas seleccionadas en las mismas son

```
select route r1
select subroute s1 from route r1
select route r2
insert subroute s2 into r2
```

Figura 3.17: Cambiar de ruta una subruta.

distintas, se definirá un rango de valores para las longitudes de las subrutas. La idea es definir una clasificación para las longitudes de las subrutas, por ejemplo: pueden haber subrutas de pequeña, mediana y larga longitud.

Dada la clasificación que considera como pequeñas aquellas subrutas con longitud 1 o 2, medianas las que cuenten con un número de clientes igual a 3 o 4 y grandes las de longitud 5. Para la vecindad definida anteriormente, por el criterio de la figura 3.17 y la solución de la figura 3.13, se cumple que las soluciones resultado de seleccionar una subruta de longitud 1 de la ruta 1 e insertar dicha subruta en la ruta 2 y de seleccionar e insertar una subruta para las mismas rutas que la anterior pero con una longitud igual a 2, pertenecen a la misma región. Para esta nueva definición el número de regiones en la vecindad anterior es igual a 18.

En esta sección se definió el concepto de región de una vecindad del VRP, a partir del cual es posible generar una partición de las mismas. En la próxima sección se presentará otra función de indexación, la cual cumple ciertas propiedades que facilitan el análisis de las regiones de una vecindad del VRP.

3.3. Una estrategia de indexación adaptada

Una región de una vecindad de un VRP está representada por el conjunto de índices correspondientes a las soluciones que pertenecen a la misma, los cuales se encuentran determinados por la función de indexación utilizada. En este sentido, es posible definir el proceso de exploración sobre una vecindad o región de la misma, como una iteración sobre las soluciones asociadas a los índices que representan a tal conjunto de soluciones. Para este propósito, es deseable que el conjunto de índices asociado a una región en la vecindad constituya una secuencia continua de números naturales positivos, lo cual hace posible guardar eficientemente el estado de la exploración de una vecindad o región de la misma, además de simplificar la definición de los generadores presentados en la herramienta propuesta

como se mostrará en el próximo capítulo.

La función de indexación presentada en la sección 3.1, no tiene en cuenta la existencia de una forma de particionar una vecindad de un VRP en regiones, por lo que en general la propiedad mencionada anteriormente no se cumple. Si se analiza el ejemplo desarrollado en la sección 3.1, se puede notar que los valores de las propiedades para el primer vecino se muestran en la figura 3.4, mientras que la figura 3.10 muestra los valores de las propiedades para la décimo tercera solución en la vecindad según el orden definido por la función de indexación clásica. Se cumple que ambas soluciones pertenecen a una misma región en la vecindad, pues comparten los mismos valores para las operaciones principales, en este caso las selecciones de rutas.

Sin embargo, en la figura 3.8 se presentan los valores de las operaciones para la sexta solución en el orden de la vecindad. Esta operación difiere de las otras dos mencionadas anteriormente en el valor de la segunda operación de selección de ruta, por lo que pertenece a otra región distinta a la que se encuentran estas otras operaciones. Además, dicha solución se encuentra en medio de estas dos soluciones en el orden definido, luego existen soluciones de una misma región para las cuales hay vecinos de otras regiones entre ellas. Por lo que se puede decir que el conjunto de índices que representa a una región de una vecindad de un VRP para la función de indexación definida en la sección 3.1 no es una secuencia continua de números naturales positivos.

Por tal razón, en esta sección se define una adaptación a la función de indexación presentada anteriormente, la cual indexa de forma continua las soluciones de una misma región en la vecindad, facilitando el análisis de estas regiones.

En la sección 3.1 al definir la función de indexación clásica, se da igual tratamiento a todas las operaciones del criterio. Esto significa concretamente que cuando el valor de una operación (sin importar su clasificación) alcanza su máximo valor, vuelve a tomar su valor inicial, y la operación anterior a esa (nuevamente, sin importar su clasificación) incrementa su valor. Esto constituye la causa por la cual las soluciones de una región de una vecindad de un VRP no son indexadas de forma continua.

Al igual que para el orden clásico, el primer vecino en la vecindad para la función de indexación adaptada es aquella solución representada por la instanciación del criterio para los valores mínimos de las operaciones. Por lo que para la vecindad definida por el criterio de la figura 3.1 y la solución

mostrada en la figura 3.16, el primer vecino en la vecindad también se encuentra definido por los valores de las operaciones mostrados en la figura 3.4. Además, la segunda solución en la vecindad también coincide con la del orden definido en secciones anteriores, se forma incrementando el valor de la última operación del criterio y está representada por los valores de las operaciones que se muestran en la figura 3.5.

Sin embargo, la diferencia con respecto a la definición del orden clásico está precisamente en el tratamiento que reciben las operaciones según su clasificación para calcular la próxima solución en la vecindad. En este sentido, el vecino que le sucede en este nuevo orden a la solución actual se forma incrementando el valor de la última operación no principal, cuando una de estas operaciones toma su máximo valor, vuelve a tomar su valor inicial, y la solución no principal anterior a esa incrementa su valor.

Esto se puede notar al computar la solución que le sigue al vecino representado por los valores de las operaciones en la figura 3.7. Se cumple que la última operación tiene su valor máximo, por lo que este toma su valor inicial, y se procede a incrementar el valor de la operación no principal anterior a esa. La operación anterior es una operación de selección de ruta, por lo que su valor no se modifica y se incrementa en uno el valor de la operación anterior a esta, la cual es una operación de selección de clientes. El conjunto de valores de las operaciones que corresponden al nuevo vecino se muestran en la figura 3.10.

En el caso de que todas las operaciones no principales tomaran su máximo valor, estas vuelven a tomar su valor inicial, y se comienza a incrementar el valor por la última operación principal. Esto se evidencia al formar la solución que le sucede al vecino asociado a los valores de las operaciones mostrados en la figura 3.18, por lo que su sucesor se representa por los valores de las operaciones en la figura 3.8. Es necesario mencionar que cada modificación al valor de una operación principal representa una región diferente en la vecindad, por lo que estar en presencia del caso anterior significa comenzar a indexar soluciones de una nueva región de la vecindad.

$\{R_1 : 1, A : 5, R_2 : 1, B : 5\}$

Figura 3.18: Instanciación del criterio en la figura 3.1 para el vigésimo quinto vecino.

Al igual que en la función clásica, este proceso se repite hasta que todas las operaciones tengan su último valor posible, el cual sería el último ve-

cino, para este ejemplo representado por los valores de las operaciones en la figura 3.11.

En esta sección se presentó una función de indexación capaz de ordenar de forma continua las soluciones de las regiones de una vecindad de un VRP. Esta función, así como el orden clásico definido en la sección 3.1 y la forma de parcicionar una vecindad en regiones presentada en la sección anterior, constituyen elementos fundamentales de los generadores definidos en la herramienta propuesta para la exploración de una vecindad de un VRP, y los cuales serán presentados en el siguiente capítulo.

Capítulo 4

Uso de generadores para explorar una Vecindad del VRP

En el proceso de exploración de una vecindad se parte de una solución inicial y una estructura de entorno. Este proceso consiste en iterar sobre las soluciones vecinas que se generan de acuerdo a cierto criterio (llamado estrategia de exploración), analizar qué tan buena es cada una y seleccionar una de ellas a través de algún otro criterio (llamado estrategia de selección).

En el presente capítulo, se verán algunos ejemplos de funciones que generan las soluciones de una vecindad siguiendo una estrategia de exploración particular. En la sección 4.1 se definen los generadores para las estrategias de exploración más comunes en la literatura. Mientras que en la sección 4.2 se presentan nuevas estrategias para generar las soluciones de una vecindad del VRP y por último en la sección 4.3 se presentan algunas de las características de estos generadores.

4.1. Estrategias clásicas de exploración

En esta sección se definen los generadores para las estrategias de exploración más conocidas en la literatura: la exploración exhaustiva y aleatoria.

4.1.1. Exploración exhaustiva

La exploración exhaustiva consiste en el análisis de todas las soluciones de una vecindad en un orden determinado. En el presente trabajo dicho orden está definido por la función de indexación que se utiliza para la vecindad en cuestión, como ya se explicó en el capítulo 3. A continuación se presentará una función para generar una a una las soluciones de una vecindad del VRP siguiendo una estrategia de exploración exhaustiva.

La función definida en la figura 4.1, representa una de las formas más sencillas de definir una exploración exhaustiva sobre una vecindad. El primer paso de esta consiste en computar la cantidad de soluciones en la vecindad a explorar utilizando el algoritmo presentado en la sección 2.2. Una vez la cardinaldiad se conoce, es posible realizar una iteración sobre el conjunto de los números naturales comenzando por el 1 y terminando en la cantidad de soluciones vecinas. Por cada uno de estos números, se utiliza una de las funciones de indexación definidas en el capítulo 3 para generar la solución vecina que está asociada con dicho índice.

```
EXHAUSTIVE_EXPLORATION(neigh):
1 card_neigh = COUNT_NEIGHBORS(neigh)
2 solutions = []
3 for i from 1 to card_neigh:
4   cur_sol = NTH_NEIGHBOR(neigh,i)
5   solutions = solutions + {cur_sol}
6 return solutions
```

Figura 4.1: Función que representa una estrategia de exploración exhaustiva.

Esta función constituye una simplificación del generador exhaustivo presente en la herramienta propuesta[1]. Al igual que la función anterior, el primer paso de este generador está determinado por el cálculo del número de soluciones de la vecindad que se desea explorar. Iniciando por el índice 1, en cada llamado al mismo se devuelve la solución que le corresponde al índice actual y se incrementa ese índice. En el caso que el índice actual sea mayor que la cardinalidad de la vecindad, se devuelve el valor “nil”.

[1]La diferencia fundamental es que el algoritmo implementado no calcula todas las soluciones, sino que en cada llamado solo calcula la solución siguiente a la última calculada.

En el presente trabajo, a este tipo de funciones se les llamarán generadores, ya que en cada llamado las mismas generan la próxima solución en la vecindad siguiendo una estrategia de exploración específica, o "nil", si todas las soluciones de la vecindad ya han sido analizadas.

Este generador exhaustivo constituye una mejora respecto a la función definida en la figura 4.1, pues este es capaz de realizar una exploración exhaustiva tanto de una vecindad como de una región de la misma, de forma perezosa. Esta funcionalidad permite una exploración de una vecindad del VRP focalizada en ciertas regiones de la misma, lo cual será una idea fundamental para la definición de otras variantes de exploración en las siguientes secciones.

El elevado número de vecinos que puede contener una vecindad del VRP, unido a la búsqueda de la eficiencia en el proceso de exploración de la misma, hacen que en algunas ocasiones resulte impracticable analizar todas las soluciones en la vecindad. Como una alternativa al problema anterior surge la estrategia de exploración aleatoria, la cual será definida en la siguiente sección.

4.1.2. Exploración aleatoria

La exploración aleatoria consiste en el análisis de un subconjunto aleatorio de todas las posibles soluciones en una vecindad. La cantidad de elementos en dicho subconjunto es dado y el orden de los mismos está determinado por un ordenamiento aleatorio de los elementos de la vecindad.

En la figura 4.2 se presenta una forma simple de definir una exploración aleatoria sobre una vecindad. Al igual que la función presentada en la sección anterior para la estrategia exhaustiva, el primer paso en esta función es propiamente el cálculo del número de soluciones vecinas. Este número se usa para crear una permutación de números naturales en el intervalo $[1; card\_neigh]$, la cual constituye el orden según el cual las soluciones de la vecindad serán generadas. Finalmente, se itera sobre dicha permutación y se utiliza una de las funciones de indexación definidas en el capítulo 3 para generar la solución vecina que le corresponde a dicho índice.

El generador aleatorio definido en la herramienta propuesta, genera una nueva solución en la vecindad de forma aleatoria en cada llamado, o "nil", si todas las soluciones de la misma ya han sido analizadas. Para esto se basa en la generación de una variación aleatoria del conjunto de números naturales representado por el intervalo $[1; card\_neigh]$.

```
UNIFORM_EXPLORATION(neigh,cant):
1 card_neigh = COUNT_NEIGHBORS(neigh)
2 permutation = PERMUTACION_ALEATORIA(1,card_neigh)
3 solutions = []
4 for i in permutation:
5   if cant == 0:
6     return solutions
5   cur_sol = NTH_NEIGHBOR(neigh,i)
6   solutions = solutions + {cur_sol}
7   cant -= 1
8 return solutions
```

Figura 4.2: Función que representa una estrategia de exploración aleatoria.

La función definida en la figura 4.2 representa una simplificación de este generador aleatorio, pues el mismo genera las soluciones de la vecindad una a una de forma perezosa, por lo que no hay necesidad de almacenar el subconjunto seleccionado de la vecindad. Además, no es necesario decidir el número de soluciones que deben formar parte de dicho subconjunto de antemano, puesto que se estará trabajando con un generador y siempre que se necesite otra solución esta puede generarla, a no ser que todos los vecinos ya hayan sido analizados.

El generador aleatorio de la herramienta propuesta, al igual que para la estrategia exhaustiva, es capaz de realizar el proceso de exploración tanto en una vecindad como en una región de la misma. En la siguiente sección se definirán dos nuevas estrategias de exploración las cuales se enfocan en un análisis por regiones de una vecindad del VRP.

4.2. Nuevas estrategias de exploración

En esta sección se presentan nuevas estrategias de exploración sobre una vecindad del VRP. Las mismas se basan en la existencia de una partición de la vecindad en regiones, conceptos definidos en el capítulo 3. Además, estas estrategias funcionan sobre la base de las clásicas definidas anteriormente y solo pueden utilizar como función de indexación la presentada en la sección 3.3. Las nuevas estrategias reciben el nombre de exploración combinatoria y secuencial, y se presentan a continuación.

4.2.1. Exploración Combinatoria

La exploración combinatoria de una vecindad del VRP consiste en explorar las soluciones de la misma, de tal forma que se cumpla que entre dos soluciones que pertenezcan a una misma región de la vecindad exista exactamente una solución por cada una de las regiones restantes que no se han analizado completamente.

En este trabajo además de definir el conjunto de regiones en las cuales se particiona una vecindad del VRP, también se precisa un orden entre las mismas. Por lo que la región de la próxima solución generada, será aquella que le sigue en dicho orden a la región de la solución anterior y aún cuenta con soluciones por ser analizadas. En el caso de alcanzar la última región en el orden, es necesario comenzar nuevamente por la primera región hasta que todas las regiones de la vecindad hayan sido completamente analizadas.

Además, las soluciones de cada región se generan de acuerdo a una cierta estrategia de exploración, las cuales en este trabajo pueden ser una estrategia exhaustiva o aleatoria.

Esta estrategia tiene el objetivo de explorar una vecindad del VRP generando una muestra lo más representativa posible de la misma. En el contexto del presente trabajo, esto significa que el número de soluciones generadas de cada región sea cercano. Por lo tanto, esta estrategia mantiene la invariante de que el número de soluciones de cada región de la vecindad que se han generado hasta el momento solo difiere a lo sumo en 1, excepto en el caso de aquellas regiones que ya han sido exploradas completamente.

En la figura 4.3 se presenta la definición de una función que genera las soluciones de una vecindad del VRP siguiendo esta estrategia. Esta función se encarga de primeramente crear un generador aleatorio por cada una de las regiones de la vecindad, pero como ya se mencionó anteriormente cualquiera de las estrategias clásicas presentadas en la sección anterior pueden utilizarse ya que las mismas son capaces de trabajar sobre regiones también. Posteriormente, se tiene un ciclo que solo termina de ejecutarse una vez la vecindad ha sido explorada completamente, en cada iteración del mismo se recorren los generadores en el orden que tienen las regiones en la vecindad, y por cada una de estas que no haya sido explorada en su totalidad se genera una solución de la misma.

La función mostrada en la figura 4.3 representa una posible implementación de una exploración combinatoria sobre una vecindad y una simplifi-

```
COMBINATORIAL_EXPLORATION(neigh):
1  generators = []
2  for reg in neigh.regions:
3    cur_gen = UNIFORM_EXPLORATION(reg)
4    generators = generators + {cur_gen}
5  solutions = []
6  while True:
7    if IS_EXHAUSTED(neigh):
8      break
9   for gen in generators:
10     cur_sol = gen()
11     if cur_sol != "nil":
12       solutions = solutions + {cur_sol}
13  return solutions
```

Figura 4.3: Función que representa una estrategia de exploración combinatoria.

cación del generador combinatorio presentado en la herramienta propuesta. Este generador, permite la exploración perezosa de una vecindad de un VRP mediante la generación una a una de las soluciones de la misma, además de la posibilidad de seleccionar la estrategia de exploración clásica que será utilizada en cada una de las regiones de la vecindad.

En esta sección se presentó una estrategia para generar las soluciones de una vecindad del VRP de una forma "representativa", en la siguiente se presentará otra estrategia que a partir de la definición de un orden entre las regiones de la vecindad genera las soluciones de la misma explorando completamente dichas regiones en tal orden, esta recibe el nombre de exploración secuencial.

4.2.2. Exploración secuencial

La exploración secuencial de una vecindad consiste en la generación de las soluciones de la misma siguiendo un orden determinado para las regiones. O sea, todas las soluciones de una región deben ser analizadas antes de continuar el proceso de exploración en la siguiente región de dicho orden. Este orden no tiene por qué contener a todas las regiones de la vecindad, por lo que se define como un subconjunto ordenado de las regiones de la

misma.

Una vez conocidas las regiones a explorar, así como el orden de estas, es necesario definir la estrategia de exploración según la cual se generarán las soluciones de las mismas. Para este propósito, al igual que la estrategia combinatoria mostrada en la sección anterior se pueden utilizar cualquiera de las estrategias clássicas definidas en la sección 4.1.

En la figura 4.4 se define una función que genera las soluciones de una vecindad del VRP siguiendo una estrategia secuencial. El primer paso de esta función consiste en la creación de una lista con los generadores aleatorios de cada una de las regiones que serán exploradas, cumpliéndose el orden definido en la lista de entrada. Posteriormente, se itera por cada uno de estos generadores, y en cada uno de ellos se analizan una a una todas las soluciones de la región que dicho generador representa.

```
SECUENCIAL_EXPLORATION(neigh,reg_lst):
1   generators = []
2   for reg in reg_lst:
3     cur_gen = UNIFORM_EXPLORATION(reg)
4     generators = generators + {cur_gen}
5   solutions = []
6   for gen in generators:
7     while True:
8       cur_sol = gen()
9       if cur_sol == "nil":
10        break
11      solutions = solutions + {cur_sol}
12   return solutions
```

Figura 4.4: Función que representa una estrategia de exploración secuencial.

La función definida en la figura 4.4 constituye una posible implementación de una exploración secuencial sobre una vecindad y una simplificación del generador secuencial definido en la herramienta propuesta. Este generador, al igual que el combinatorio, permite analizar de forma perezosa una vecindad de un VRP, así como la selección de la estrategia de exploración clásica que será utilizada para el análisis en las regiones.

Esta estrategia tiene el objetivo de explorar las regiones de una vecindad

del VRP de acuerdo a su prioridad, de forma tal que todas las soluciones que pertecen a una región de alta prioridad deben ser analizadas antes que aquellas soluciones de regiones con una prioridad menor. En este trabajo el nivel de prioridad de las regiones se define mediante un análisis estadístico (sección 1.6) de una muestra de la vecindad. Para no afectar la validez de este análisis por causa de una distribución poco uniforme de las soluciones de la muestra en las regiones, es deseable que la misma sea tan representativa como sea posible.

En esta sección se presentó una estrategia de exploración para realizar un análisis focalizado de las regiones de una vecindad del VRP, en la siguiente se mostrarán algunas de las características y propiedades generales de los generadores que representan estas estrategias en la herramienta propuesta.

4.3. Características de las funciones generadoras

Las funciones mostradas en las secciones anteriores constituyen una simplificación de los generadores presentados en la herramienta propuesta. Estos generadores siguen una interfaz bien definida en cuanto a la generación de las soluciones en la vecindad (o región) que se está analizando, pues en cada paso se genera una nueva solución de la vecindad (o región), o "nil", en el caso de que la vecindad (o región) haya sido explorada completamente.

Estos generadores representan formas de cómo explorar un conjunto de soluciones. En este trabajo estos conjuntos pueden ser tanto una vecindad del VRP, como una región de la misma. Todos los generadores definidos en este documento son capaces de explorar una vecindad, mientras que los denominados clásicos pueden además realizar el proceso de exploración sobre una región de la misma, no siendo así para las nuevas estrategias introducidas en la sección 4.2 las cuales necesitan de la existencia de una partición en el conjunto de soluciones que está analizando.

Para estos generadores, un conjunto de soluciones (ya sea una vecindad o una región de la misma) está identificada por un intervalo continuo de números naturales. Cada uno de estos números (o índices) está asociado con una solución del conjunto mediante una función de indexación. Por ejemplo, para el caso de una vecindad este intervalo estará formado por los números del 1 a la cantidad de vecinos que hay en la misma, y ambas de

las funciones de indexación que fueron definidas en el capítulo 3 pueden utilizarse. Sin embargo, para el caso de una región es necesario que la función de indexación cumpla la propiedad de asignar índices consecutivos a todas las soluciones de las región. La función de indexación introducida en la sección 3.1 no cumple esta propiedad, por esta razón para explorar las regiones de una vecindad del VRP es necesario utilizar la función de indexación presentada en la sección 3.3.

Esta forma de representar un conjunto de soluciones vecinas constituye una simplificación del espacio que se está explorando al de los números naturales, lo que posibilita guardar eficientemente el estado del proceso de exploración en estos conjuntos.

Para almacenar el estado del proceso de exploración, los conceptos de vecindad y región definen dos propiedades, la primera guarda la cantidad de soluciones que han sido analizadas del propio conjunto, mientras que la segunda es una función de asociación. Esta función almacena el nuevo orden aleatorio definido para las soluciones de la vecindad, asociando para cada índice del conjunto de soluciones (definido por la función de indexación que se está utilizando) el correspondiente en el nuevo ordenamiento del conjunto.

Los generadores presentados en este trabajo siguen una estrategia común para modificar estas propiedades que representan el estado del proceso de exploración en una vecindad del VRP o región de la misma. En realidad, solo los generadores clásicos modifican estas propiedades directamente, mientras que las nuevas estrategias definidas en la sección anterior las modifican indirectamente mediante el uso interno de las estrategias clásicas de exploración para analizar las regiones de la vecindad.

La estrategia exhaustiva solo incrementa la cantidad de soluciones analizadas del conjunto de soluciones que se está explorando, puesto que el orden en que se generan estas soluciones en dicha estrategia se encuentra completamente determinado por la función de indexación utilizada. Sin embargo, en la estrategia aleatoria además de incrementar el número de soluciones analizadas es necesario la modificación de la función de asociación.

El generador aleatorio utiliza esta función de asociación para crear una variación aleatoria (una modificación del algoritmo para crear una permutación aleatoria de los números $1 \ldots n$ [73]) del intervalo de índices que le corresponde al conjunto de soluciones que se está analizando. Si se define como $[a;b]$ el intervalo de índices correspondiente al conjunto de soluciones

que se desea explorar, k la cantidad de soluciones que han sido analizadas hasta el momento, y se asume que en un comienzo la función de asociación coincide con la identidad (sea ψ la función de asociación, se cumple $\forall x \in [a;b] : \psi(x) = x$). En cada llamado al generador ocurre lo siguiente:

1. Si $k = (b - a) + 1$, entonces el generador retorna "nil".
2. Se genera un número aleatorio en el intervalo $[(a + k);b]$, el cual se denotará x.
3. Si $x = (a + k)$, entonces se retorna $\psi(x)$.
4. Si $x \neq (a + k)$, entonces se retorna $\psi(x)$ y además es necesario actualizar el valor asignado a x en la función de asociación $\psi(x) = \psi(a + k)$.

Debido a esta propiedad de los generadores que les permite modificar convenientemente el estado del proceso de exploración sobre el conjunto de soluciones que se está analizando, es posible combinar el uso de estos generadores para explorar una vecindad o región de la misma sin repetir el análisis de soluciones vecinas. Por ejemplo, al analizar una vecindad del VRP es posible explorar una parte de las soluciones vecinas utilizando una estrategia de exploracón combinatoria y el resto de las soluciones de la vecindad mediante una estrategia de exploración secuencial. Un ejemplo de esta combinación de generadores se mostrará en la sección siguiente cuando se presente una heurística para explorar una vecindad del VRP.

Esta combinación de generadores posibilita la definición de estrategias de exploración más complejas a partir de otras más sencillas de una forma simple, lo cual constituye una de las grandes ventajas del uso de los mismos. Estos generadores representan una forma perezosa de generar las soluciones en una vecindad y proveen al proceso de exploración de la misma de cierta eficiencia. En la siguiente sección se introduce una heurística para realizar una exploración en dos fases de una vecindad del VRP la cual se basa en el uso y la combinación de estos generadores.

4.4. Exploración en dos fases de una vecindad de un VRP

Investigaciones recientes [27] han demostrado que la ocurrencia sumultánea entre la exploración y explotación de un espacio de búsqueda puede

limitar significativamente la efectividad de la búsqueda heurística en dicho espacio. Esto ha llevado al diseño de algoritmos híbridos que separan estas dos tareas para aliviar dicha limitación [16].

En esta sección se presenta una heurística para explorar una vecindad de un VRP realizando el análisis de la misma en dos etapas. Una primera, llamada de exploración, donde se intenta construir una muestra representativa de la vecindad en cuestión, y otra etapa posterior de explotación (o de intensificación), en la cual se exploran las regiones más prometedoras de la vecindad que fueron identificadas mediante el uso de ténicas estadísticas sobre la muestra generada en la fase anterior.

Este algoritmo se basa en las ideas de los híbridos de solo-exploración solo-explotación presentadas en la sección 1.5, y se caracteriza por realizar un muestreo de la vecindad en cuestión (que consume muy poco tiempo), estimar las "mejores" regiones de la vecindad utilizando las técnicas estadísticas introducidas en la sección 1.6, e intensificar la búsqueda en estas regiones.

En la primera etapa de este algoritmo, o fase de exploración, se utiliza una estrategia de exploración combinatoria sobre la vecindad analizada. Esto permite, debido a las propiedades de esta estrategia, la creación de una muestra "representativa" de la vecindad donde todas las regiones de la misma son analizadas en igual medida. Por lo tanto, para la implementación de esta primera etapa del algoritmo se usa un generador combinatorio (sección 4.2.1).

En este algoritmo, la transición entre las dos etapas se define de acuerdo al número de soluciones analizadas en la vecindad. Para esto se define la cantidad total de soluciones que serán analizadas por el algoritmo y de ese total se asigna una proporción para explorar la vecindad y otra para intensificar la búsqueda en las mejores regiones de la misma. En la herramienta propuesta, a pesar de ser adaptable, el número total de soluciones a analizar se define como un orden cuadrado de la cantidad de clientes en el VRP involucrado, pero este número puede variar de problema en problema.

Por lo dicho anteriormente, en el algoritmo de dos fases presentado, la transición a la etapa de explotación (o intensificación en las mejores regiones) tendrá lugar una vez el generador combinatorio haya devuelto una cantidad de vecinos igual a la proporción del número total de soluciones a analizar que le fue asignada en la inicialización de este algoritmo. Pero antes de comenzar la etapa de explotación, es necesario identificar las regiones más prometedoras en la vecindad, para lo que se utilizan las técnicas

estadísticas presentadas en la sección 1.6.

Una vez definido un orden para las regiones de una vecindad de un VRP de más a menos prometedoras, comienza la fase de explotación. En la segunda etapa de este algoritmo se utiliza un estrategia de exploración secuencial sobre la vecindad analizada, implementada como un generador secuencial que sigue el orden previamente definido para explorar las regiones de la vecindad que no hayan sido analizadas completamente.

El empleo de una estrategia de exploración secuencial combinado con la definición de un orden para las regiones de la vecindad de más a menos prometedoras, permite realizar un análisis de la vecindad focalizado en las mejores regiones de la misma, o sea intensificando la búsqueda en estas regiones. Por lo que se puede decir que este algoritmo intenta aumentar el número de soluciones mejores que la actual analizadas en el proceso de exploración de una vecindad de un VRP. Esto representa una vía para mejorar los resultados obtenidos en dicho proceso.

El criterio de parada de este algoritmo, o de terminación de la segunda fase del mismo, se define a partir del número de soluciones analizadas al igual que en la primera etapa del mismo, y coincide con la resta entre el total de soluciones que deben ser exploradas y las analizadas en la etapa de exploración.

En esta heurística existen un conjunto de parámetros cuyos valores pueden ser configurados en la inicialización del mismo. Por ejemplo, la constante que determina el exponente al que el número de clientes en el VRP involucrado es elevado, así como la constante que es multiplicada a la potencia anterior y juntos determinan el total de soluciones que serán analizadas por el algoritmo. Además, la proporción del total de soluciones que se analizarán en las distintas fases del algoritmo. También, la técnica estadística que será usada, y las estrtegias de exploración en las regiones para las fases de exploración y explotación.

En el presente capítulo se definieron varios generadores que representan estrategias para explorar una vecindad cualquiera del Problema de Enrutamiento de Vehículos, así como las características de los mismos y una heurística de dos fases para explorar una vecindad de un VRP que utiliza estos generadores. En el siguiente se presentan algunos de los resultados obtenidos para la implementación del algoritmo propuesto en este capítulo para las diferentes técnicas estadísticas implementadas.

Capítulo 5

Resultados

En este capítulo se presentan algunos de los resultados obtenidos para la exploración de una vecindad de un VRP utilizando el algoritmo definido en el anterior. Para esto se utilizarán tres instancias del Problema de Enrutamiento de Vehículos con restricciones Capacidad (CVRP, por sus siglas en inglés) conocidas en la literatura [10] como: **a-n32-k5**, **a-n65-k9** y **a-n80-k10**.

Una vecindad de un VRP se define a partir de una solución de este y un criterio de vecindad. En este trabajo, se definieron 10 criterios de vecindad y para cada par problema-criterio se construyeron 10 soluciones aleatorias del VRP en cuestión. Por lo tanto, los datos presentados en este capítulo muestran los resultados obtenidos por el algoritmo propuesto en la sección 4.4 haciendo una comparación con una exploración aleatoria para 300 vecindades del VRP.

En la sección 5.1 se definen los criterios de vecindad usados para generar las vecindades utilizadas en esta experimentación. En la sección 5.2 se muestra por qué no pudo utilizarse el análisis de varianza como técnica estadística, ya que en ninguno de los 300 casos comprobados se cumplen las hipótesis. En la sección 5.3 se utiliza como técnica estadística el cálculo de las medias de los grupos, mientras que en la sección 5.4 se utilizan los árboles de regresión.

5.1. Algunos criterios de vecindad para el VRP

En esta sección se muestran los criterios de vecindad utilizados para definir las vecindades del VRP sobre las cuales se computan los resultados para este trabajo.

En las figuras 5.1 y 5.2 se muestran los criterios que representan el intercambio de clientes y subrutas, estas últimas pudiendo tener invertido el orden de sus clientes o no. Los criterios definidos en las figuras 5.3, 5.4, 5.5 y 5.6 responden al movimiento de una subruta dentro de su ruta original o hacia otra ruta de la solución involucrada, con la posibilidad de invertir el orden de los clientes en dicha subruta. Por último, los criterios presentados en las figuras 5.7, 5.8, 5.9 y 5.10 representan el intercambio de dos subrutas, en las que el orden de los clientes de las mismas puede estar invertido o no.

Una vez definidos estos criterios, en la siguiente sección se muestra cómo no se cumplen los supuestos 2 y 3 del análisis de varianza para las vecindades generadas a partir de la aplicación de estos sobre las distintas soluciones construidas para los VRP utilizados.

```
select route r1
select client c1 from route r1
select route r2
select subroute z1 from route r2
insert client c1 into route r2
insert subroute z1 into route r1
```

Figura 5.1: Intercambio de un cliente y una subruta.

```
select route r1
select client c1 from route r1
select route r2
select subroute z1 from route r2
reverse subroute z1
insert client c1 into route r2
insert subroute z1 into route r1
```

Figura 5.2: Intercambio de un cliente y una subruta con el orden de los clientes invertido.

```
select route r1
select subroute z1 from route r1
insert subroute z1 into route r1
```

Figura 5.3: Mover una subruta dentro de su ruta.

```
select route r1
select subroute z1 from route r1
select route r2
insert subroute z1 into route r2
```

Figura 5.4: Mover una subruta hacia otra ruta.

5.2. Exploración en dos fases con análisis de varianza

El análisis de varianza, como ya fue mencionado en la sección 1.6, requiere del cumplimiento de tres supuestos por parte de los grupos involucrados para su utilización, que son: la independencia entre las muestras, que estas pertenezcan a poblaciones con distribución normal y la igualdad de varianza entre estas poblaciones.

En el presente trabajo, debido a la forma en que se generan los grupos, estos son independientes por lo que el primero de los supuestos se cumple. Sin embargo, al realizar una experimentación sobre las vecindades definidas en este capítulo se llega a la conclusión que los grupos para es-

```
select route r1
select subroute z1 from route r1
reverse subroute z1
insert subroute z1 into route r1
```

Figura 5.5: Mover una subruta con el orden de los clientes invertido dentro de su ruta.

```
select route r1
select subroute z1 from route r1
select route r2
reverse subroute z1
insert subroute z1 into route r2
```

Figura 5.6: Mover una subruta con el orden de los clientes invertido hacia otra ruta.

```
select route r1
select subroute z1 from route r1
select route r2
select subroute z2 from route r2
swap subroutes z1 y z2
```

Figura 5.7: Intercambio de dos subrutas.

```
select route r1
select subroute z1 from route r1
select route r2
select subroute z2 from route r2
reverse subroute z1
swap subroutes z1 y z2
```

Figura 5.8: Intercambio de una subruta con el orden de sus clientes invertido con otra subruta.

tas vecindades no pertenecen a poblaciones con distribución normal y no existe una igualdad entre las varianzas de estas distribuciones, ya que en ninguna de las 300 vecindades analizadas se cumplieron estos supuestos.

Aunque el análisis de varianza es robusto para ciertos casos en los cuales se incumple el supuesto de normalidad [56], en la literatura se muestra que esta técnica es sensible a violaciones en la asumpción de homocedasticidad en las poblaciones [59, 92], y como en este caso, no se cumple, se decidió no utilizar esta técnica.

En las siguientes secciones se mostrarán los resultados para la utilización de la estadística descriptiva y los árboles de decisión (árboles de regresión), como técnicas estadísticas para estimar las mejores regiones de la vecindad, dentro del algoritmo presentado en el capítulo anterior.

```
select route r1
select subroute z1 from route r1
select route r2
select subroute z2 from route r2
reverse subroute z2
swap subroutes z1 y z2
```

Figura 5.9: Intercambio de una subruta con otra con el orden de sus clientes invertido.

```
select route r1
select subroute z1 from route r1
select route r2
select subroute z2 from route r2
reverse subroute z1
reverse subroute z2
swap subroutes z1 y z2
```

Figura 5.10: Intercambio de dos subrutas con el orden de sus clientes invertido.

5.3. Exploración en dos fases con cálculo de medias

En esta sección se presentan los resultados para la exploración de las vecindades definidas en este capítulo, utilizando el algoritmo de dos fases con el cálculo de medias como técnica estadística. Para esto se toma como referencia una exploración aleatoria de estas vecindades.

La utilización del cálculo de medias para identificar las regiones más prometedoras de la vecindad se realiza mediante el cómputo de dicha propiedad para cada una de las muestras de las regiones. Posteriormente, estas muestras serán ordenadas de menor a mayor media, siendo las de menor media las que se encuentran en las mejores regiones pues esto implica un menor costo de evaluación.

Es necesario aclarar que las vecindades analizadas se caracterizan por tener una cantidad pequeña de soluciones mejores que la actual con respecto a la cardinalidad de la mismas, pues en estas vecindades el número de mejoras es siempre menor del 15% de su cardinalidad, y en algunas, menor que el 1%. Además, el número de soluciones analizadas en cada una de estas vecindades está relacionado con la cantidad de clientes en el VRP involucrado, este número es exactamente cinco veces el cuadrado de la cantidad de clientes en el VRP en cuestión. La mitad de este total de soluciones serán analizadas en la etapa de exploración y las restantes, en la fase de explotación.

En la tabla 5.1 se muestra una comparación de los resultados del algoritmo de dos fases con cálculos de medias y la exploración aleatoria. Estos datos son el producto de ejecutar estos algoritmos sobre las 300 vecindades, realizando 30 iteraciones sobre cada una de estas. La primera fila de esta tabla representa el número de vecindades en las que estos métodos alcanzan

	Exp.Uniforme	Dos Fases con Medias
Alcanza el óptimo	167	217
Alcanza mayor número de mejoras	0	192

Cuadro 5.1: Número de vecindades para las que dos fases con cálculo de medias y exploración aleatoria alcanzan el óptimo, y la cantidad de estas en las que el primero supera al otro en cuanto al número de mejoras encontradas y viceversa.

el óptimo y la segunda, la cantidad de vecindades para cada algoritmo en las que este supera el otro con respecto al número de mejoras alcanzadas. Como se puede apreciar el algoritmo propuesto obtiene mejores resultados que la exploración aleatoria, encontrando el óptimo en un número mayor de vecindades y una mayor cantidad de mejoras que la estrategia aleatoria en 192 de las 300 vecindades, mientras que la búsqueda aleatoria no pudo superar al propuesto en este ámbito en ninguna de las vecindades analizadas. En las 108 vecindades restantes ambos métodos encontraron la misma cantidad de mejoras.

	% de mejoras alcanzadas	% mínimo	% máximo
Exp. Uniforme	51.7	0	100
Dos Fases con Medias	76.3	6.8	100

Cuadro 5.2: Promedio del porciento de mejoras encontradas en las 300 vecindades por ambos algoritmos. Además, el porciento mínimo y máximo alcanzado para alguna de estas vecindades.

Al explorar estas vecindades, en lugar de almacenar el número de mejoras que estos algoritmos son capaces de encontrar en las mismas, se guarda el porciento que dicho número representa del total de las mejoras existentes en la vecindad en cuestión. Por ejemplo, si al explorar una vecindad con un número de soluciones mejores que la actual igual a 220 la cantidad de mejoras encontradas es 20, entonces se guarda el porciento que este número representa del total, o sea, aproximadamente el 9,1 %. Este proceso se repite para las 300 vecindades y estos porcientos se promedian para producir los valores que se muestran en la primera columna de la tabla 5.2.

En la tabla 5.2 se muestra el promedio de estos porcientos para las 300 vecindades del VRP definidas en este capítulo. En esta se evidencia la su-

perioridad del algoritmo propuesto en comparación con una exploración aleatoria para encontrar las mejores soluciones de la vecindad, siendo capaz de analizar una cantidad de mejoras en una vecindad del VRP que sobrepasa al número de la aleatoria en casi un 25%. También se muestran en esta tabla el porciento de mejoras mínimo y máximo que ambos algoritmos encontraron para las vecindades ya mencionadas. Ambos métodos logran encontrar el 100% de las mejoras en algunas vecindades, sin embargo, mientras la búsqueda aleatoria no es capaz de encontrar ninguna mejora en alguna de las vecindades, el algoritmo propuesto encuentra más del 6% del total de mejoras para las vecindades exploradas.

5.4. Exploración en dos fases con árboles de regresión

En esta sección se presentan los resultados para el algoritmo de dos fases introducido en el capítulo anterior utilizando un árbol de regresión como técnica estadística para la exploración de las vecindades definidas anteriormente.

En la tabla 5.3 se muestra una comparación de este algoritmo con una estrategia de exploración aleatoria para las 300 vecindades analizadas. Se puede apreciar que el algoritmo en cuestión es superior a dicha estrategia aleatoria, puesto que es capaz de encontrar el óptimo en un número mayor de vecindades, y de igual forma que para la variante con cálculo de medias, este método alcanza una cantidad mayor de mejoras en 192 de las 300 vecindades exploradas. Se cumple que para las 108 vecindades restantes ambos métodos encuentran el mismo número de mejoras, pues 107 de esas vecindades se analizaron completamente.

	Exp.Uniforme	Dos Fases con Árbol
Alcanza el óptimo	167	205
Alcanza mayor número de mejoras	0	192

Cuadro 5.3: Número de vecindades para las que dos fases con un árbol de regresión y exploración aleatoria alcanzan el óptimo, y la cantidad de estas en las que el primero supera al otro en cuanto al número de mejoras encontradas y viceversa.

El algoritmo de dos fases con árbol de regresión funciona ligeramente mejor que la variante con el cálculo de medias en cuanto al promedio del

porciento de mejoras encontradas en las vecindades exploradas. Sin embargo, encuentra el óptimo en un número menor de vecindades. Esto se evidencia en los datos presentados en la tabla 5.4, a partir de esta se puede decir además que este método es muy superior a la exploración aleatoria pues encuentra en promedio una cantidad de soluciones mejores que la actual para una vecindad que supera en más de un 25% el número alcanzado por el aleatorio.

	% de mejoras alcanzadas	% mínimo	% máximo
Exp. Uniforme	51.7	0	100
Dos Fases con Árbol	77.3	6.7	100

Cuadro 5.4: Promedio del porciento de mejoras encontradas en las 300 vecindades por ambos algoritmos. Además, el porciento mínimo y máximo alcanzado para alguna de estas vecindades.

Estos resultados indican que el algoritmo de dos fases es significativamente superior a la estrategia de exploración aleatoria para una vecindad de un VRP. Además, parece ser que la variante con un árbol de regresión como técnica estadística se comporta ligeramente mejor que la que usa el cálculo de medias, pues el primero logra encontrar en promedio un número mayor de mejoras en las vecindades analizadas, a pesar de alcanzar el óptimo en una cantidad menor de estas. Sin embargo, el tiempo de ejecución con el árbol de regresión fue mucho mayor, por lo que se recomienda la implementación del algoritmo de dos fases con el cálculo de medias.

Conclusiones

En este trabajo se propone una herramienta que permite analizar una vecindad de un Problema de Enrutamiento de Vehículos, utilizando varias estrategias de exploración. Dicha herramienta permite computar eficientemente ciertas características de una vecindad, como el número de soluciones que hay en la misma, y almacenar de una forma eficiente toda la información necesaria de dicha vecindad para realizar cualquier tipo de análisis sobre esta a través de una estructura arbórea llamada árbol de vecindad.

Además, en este trabajo se presentó una forma de definir una biyección entre el conjunto de soluciones de una vecindad de un VRP y el de los números naturales, lo que indirectamente define un orden entre las soluciones vecinas y permite guardar de una forma eficiente el estado del proceso de exploración de la vecindad en cuestión. También se presenta cómo particionar una vecindad de un VRP en regiones, o grupos de soluciones que comparten ciertas características.

Los generadores introducidos en este trabajo constituyen una idea novedosa y conveniente para representar las estrategias de exploración sobre un conjunto de soluciones. La propiedad de los mismos de modificar convenientemente el estado del proceso de exploración sobre la vecindad o región involucrada, permite la combinación de estos generadores para explorar una vecindad de un VRP o región de esta sin repetir el análisis de soluciones vecinas. Esto significa que es posible definir estrategias de exploración más complejas sobre una vecindad a partir de la combinación de estos generadores.

El algoritmo de dos fases para la exploración de una vecindad propuesto en este trabajo constituye una de esas estrategias de exploración más complejas, el mismo se define a partir de la composición de un generador combinatorio y otro secuencial. Luego de los experimentos realizados los resultados para el algoritmo de dos fases propuesto son superiores a una

exploración uniforme de la vecindad, cuando se utilizan como técnicas estadísticas el cálculo de medias y los árboles de regresión. A pesar de que la exploración uniforme es una buena opción para alcanzar el óptimo de una vecindad, el algoritmo propuesto encuentra una cantidad considerablemente mayor de mejoras en la vecindad. Por último, el algoritmo de dos fases propuesto funciona ligeramente mejor cuando usa como técnica estadística el árbol de regresión, aunque esta consume un poco más de tiempo que el cálculo de medias.

Recomendaciones

Como trabajo futuro se recomienda el diseño de otros generadores que representen nuevas estrategias de exploración sobre una vecindad, como una forma de ampliar las variantes ya existentes en la herramienta propuesta y la posibilidad de definir otras más complejas a partir de la combinación de las ya definidas.

Además, se propone el uso de técnicas estadísticas más complejas para estimar las regiones más prometedoras de la vecindad como los modelos lineales generalizados. También, se recomienda la utilización de algoritmos de aprendizaje de máquinas como los clasificadores para decidir el punto de transición entre la etapa de exploración y explotación en el algoritmo de dos fases propuesto. Esta transición en este algoritmo se realiza en consideración al número de soluciones analizadas en cada fase del mismo.

Los próximos trabajos deben estar dirigidos en estas direcciones.

Bibliografía

[1] E. Aarts, J. H. M. Korst, and P. J. M. Van Laarhoven. Simulated aneaing. In E. Aarts and J. Lenstra, editors, *Local search in combinatorial optimization*, pages 91–120. John Wiley & Sons, Chichester, 1997. (Citado en la página 7).

[2] Agostinho Agra, Marielle Christiansen, Rosa Figueiredo, Lars Magnus Hvattum, Michael Poss, and Cristina Requejo. The robust vehicle routing problem with time windows. *Computers & operations research*, 40(3):856–866, 2013. (Citado en la página 5).

[3] Marwa Amous, Said Toumi, Bassem Jarboui, and Mansour Eddaly. A variable neighborhood search algorithm for the capacitated vehicle routing problem. *Electronic Notes in Discrete Mathematics*, 58:231–238, 2017. (Citado en la página 7).

[4] Claudia Archetti, Luca Bertazzi, Gilbert Laporte, and Maria Grazia Speranza. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. *Transportation science*, 41(3):382–391, 2007. (Citado en la página 2).

[5] Claudia Archetti, M Grazia Speranza, and Daniele Vigo. Chapter 10: Vehicle routing problems with profits. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 273–297. SIAM, 2014. (Citado en la página 2).

[6] Claudia Archetti and Maria Grazia Speranza. Vehicle routing problems with split deliveries. *International transactions in operational research*, 19(1-2):3–22, 2012. (Citado en la página 2).

[7] Alina Fernández Arias. Problema de enrutamiento de vehículos con recogida y entrega simultánea considerando una flota heterogénea.

Master's thesis, Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba, July 2010. (Citado en las páginas 10 y 11).

[8] Alina Fernández Arias. *Modelos y métodos para el problema de enrutamiento de vehículos con recogida y entrega simultánea*. PhD thesis, Facultad de Matemática y Computación, Universidad de La Habana, 2017. (Citado en la página 11).

[9] Alina Fernández Arias and Sira Allende Alonso. Estrategia grasp para el problema de enrutamiento de vehículos con recogida y entrega simultánea. *Investigación Operacional*, 38(4):424–435, 2018.

[10] Ph Augerat, Jose Manuel Belenguer, Enrique Benavent, A Corberán, D Naddef, and G Rinaldi. *Computational results with a branch and cut code for the capacitated vehicle routing problem*, volume 34. IMAG, 1995. (Citado en la página 54).

[11] Roberto Baldacci, Maria Battarra, and Daniele Vigo. Routing a heterogeneous fleet of vehicles. In *The vehicle routing problem: latest advances and new challenges*, pages 3–27. Springer, 2008. (Citado en la página 5).

[12] Roberto Baldacci, Paolo Toth, and Daniele Vigo. Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175(1):213–245, 2010. (Citado en la página 2).

[13] Edward J Beltrami and Lawrence D Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974. (Citado en la página 6).

[14] Russell Bent and Pascal Van Hentenryck. A two-stage hybrid local search for the vehicle routing problem with time windows. *Transportation Science*, 38(4):515–530, 2004. (Citado en la página 7).

[15] Leonora Bianchi and Ann Melissa Campbell. Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem. *European Journal of Operational Research*, 176(1):131–144, 2007. (Citado en la página 2).

[16] Antonio Bolufé-Röhler, Stephen Chen, and Dania Tamayo-Vera. An analysis of minimum population search on large scale global optimization. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1228–1235. IEEE, 2019. (Citado en las páginas 12 y 52).

[17] Antonio Bolufé-Röhler, Sonia Fiol-González, and Stephen Chen. A minimum population search hybrid for large scale global optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1958–1965. IEEE, 2015. (Citado en la página 12).

[18] Antonio Bolufé-Röhler and Dania Tamayo-Vera. Machine learning based metaheuristic hybrids for s-box optimization. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–14, 2020. (Citado en la página 12).

[19] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering*, 99:300–313, 2016. (Citado en la página 5).

[20] Olli Bräysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003. (Citado en la página 7).

[21] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118, 2005. (Citado en la página 5).

[22] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, 39(1):119–139, 2005. (Citado en las páginas 5 y 7).

[23] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. Classification and regression trees. belmont, ca: Wadsworth. *International Group*, 432:151–166, 1984. (Citado en la página 13).

[24] Ann Melissa Campbell and Jill Hardin Wilson. Forty years of periodic vehicle routing. *Networks*, 63(1):2–15, 2014. (Citado en las páginas 2 y 6).

[25] Diego Cattaruzza, Nabil Absi, and Dominique Feillet. The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science*, 50(2):676–693, 2016. (Citado en la página 2).

[26] Diego Cattaruzza, Nabil Absi, and Dominique Feillet. Vehicle routing problems with multiple trips. *4OR*, 14(3):223–259, 2016. (Citado en la página 2).

[27] Stephen Chen, James Montgomery, Antonio Bolufé-Röhler, and Yasser Gonzalez-Fernandez. A review of thresheld convergence. *GECONTEC: Revista Internacional de Gestión del Conocimiento y la Tecnología*, 3(1), 2015. (Citado en las páginas 12 y 51).

[28] Wen-Chyuan Chiang and Robert A Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63(1):3–27, 1996. (Citado en la página 7).

[29] Wen-Chyuan Chiang and Robert A Russell. A reactive tabu search metaheuristic for the vehicle routing problem with time windows. *INFORMS Journal on computing*, 9(4):417–430, 1997. (Citado en la página 7).

[30] Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964. (Citado en las páginas 1 y 2).

[31] Jean-François Cordeau, Gilbert Laporte, and Anne Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society*, 52(8):928–936, 2001. (Citado en la página 7).

[32] Peter I Cowling and Ralf Keuthen. Embedded local search approaches for routing optimization. *Computers & operations research*, 32(3):465–490, 2005. (Citado en la página 7).

[33] Benoit Crevier, Jean-François Cordeau, and Gilbert Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European journal of operational research*, 176(2):756–773, 2007. (Citado en la página 5).

[34] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959. (Citado en las páginas 1, 2 y 4).

[35] Dafne García de Armas y Alina Fernández Arias. Una estrategia de penalizacion aplicada al problema de enrutamiento de vehículos considerando una flota heterogénea. 2013.

[36] Eduardo D. Nuñez de Villavicencio Sánchez. Propuesta de exploración ágil de cualquier vecindad en un vrp. Master's thesis, Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba, May 2018. (Citado en las páginas 8 y 9).

[37] John H Drake, Nikolaos Kililis, and Ender Özcan. Generation of vns components with grammatical evolution for vehicle routing. In *European Conference on Genetic Programming*, pages 25–36. Springer, 2013. (Citado en la página 7).

[38] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009. (Citado en las páginas 4 y 5).

[39] Miguel Andres Figliozzi. An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies*, 18(5):668–679, 2010. (Citado en la página 5).

[40] Heidy Abreu Fumero. Exploración de una vecindad para una solución de vrp combinando diferentes estrategias de exploración y de selección. Master's thesis, Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba, May 2019. (Citado en la página 21).

[41] Michel Gendreau, Gilbert Laporte, and Jean-Yves Potvin. Metaheuristics for the capacitated vrp. In *The vehicle routing problem*, pages 129–154. SIAM, 2002. (Citado en las páginas 2 y 6).

[42] Filip Gheysens, Bruce Golden, and Arjang Assad. A comparison of techniques for solving the fleet size and mix vehicle routing problem. *Operations-Research-Spektrum*, 6(4):207–216, 1984. (Citado en la página 5).

[43] Fred Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989. (Citado en la página 7).

[44] Fred Glover. Tabu search—part ii. *ORSA Journal on computing*, 2(1):4–32, 1990. (Citado en la página 7).

[45] Damon Gulczynski, Bruce Golden, and Edward Wasil. The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering*, 61(3):794–804, 2011. (Citado en la página 6).

[46] Pierre Hansen and Nenad Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001. (Citado en la página 7).

[47] Pierre Hansen, Nenad Mladenovic, and Jose Andrés Moreno Pérez. Variable neighbourhood search. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 19:77–92, 2003. (Citado en la página 10).

[48] Liangjun Ke and Zuren Feng. A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 40(2):633–638, 2013. (Citado en la página 5).

[49] Philip Kilby and Paul Shaw. Vehicle routing. In Francesca Rossi, Peter Van Beek, and Toby Walsh, editors, *Handbook of constraint programming*, pages 801–836. Elsevier, 2006. (Citado en la página 6).

[50] Tae Kyun Kim. Understanding one-way anova using conceptual figures. *Korean journal of anesthesiology*, 70(1):22, 2017. (Citado en la página 13).

[51] Yiyo Kuo and Chi-Chang Wang. A variable neighborhood search for the multi-depot vehicle routing problem with loading cost. *Expert Systems with Applications*, 39(8):6949–6954, 2012. (Citado en la página 6).

[52] Rahma Lahyani, Mahdi Khemakhem, and Frédéric Semet. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1–14, 2015. (Citado en la página 5).

[53] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981. (Citado en las páginas 2 y 6).

[54] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011. (Citado en la página 13).

[55] Francesco Maffioli. The vehicle routing problem: A book review. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):149–153, 2003. (Citado en la página 1).

[56] Blanca Mena, M José, Rafael Alarcón, Jaume Arnau Gras, Roser Bono Cabré, and Rebecca Bendayan. Non-normal data: Is anova still a valid option? *Psicothema, 2017, vol. 29, num. 4, p. 552-557*, 2017. (Citado en la página 57).

[57] N Mladenovic. A variable neighborhood algorithm–a new metaheuristic for optimization combinatorial. In *Abstract of papers presented at*

Optimization Days, Montreal, volume 12, page 234, 1995. (Citado en la página 2).

[58] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997. (Citado en la página 2).

[59] Karl Moder. Alternatives to f-test in one way anova in case of heterogeneity of variances (a simulation study). *Psychological Test and Assessment Modeling*, 52(4):343–353, 2010. (Citado en la página 57).

[60] GG Moisen. Classification and regression trees. *In: Jørgensen, Sven Erik; Fath, Brian D.(Editor-in-Chief). Encyclopedia of Ecology, volume 1. Oxford, UK: Elsevier. p. 582-588.*, pages 582–588, 2008. (Citado en la página 15).

[61] Jairo R Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79:115–129, 2015. (Citado en la página 6).

[62] James N Morgan and John A Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, 58(302):415–434, 1963. (Citado en la página 14).

[63] Camila Pérez Mosquera. Primeras aproximaciones a la búsqueda de vecindad infinitamente variable. Master's thesis, Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba, May 2017. (Citado en las páginas 2, 7, 8, 9, 10, 17, 27 y 28).

[64] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998. (Citado en la página 13).

[65] United Nations. World population projected to reach 9.8 billion in 2050, and 11.2 billion in 2100. `https://www.un.org/development/desa/en/news/population/world-population-prospects-2017.html`, 2017. Visitado: 2020-05-27. (Citado en la página 1).

[66] Sandra Ulrich Ngueveu, Christian Prins, and Roberto Wolfler Calvo. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 37(11):1877–1885, 2010. (Citado en la página 5).

[67] Puca Huachi Vaz Penna, Anand Subramanian, and Luiz Satoru Ochi. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232, 2013. (Citado en la página 2).

[68] Michael Polacek, Richard F Hartl, Karl Doerner, and Marc Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, 10(6):613–627, 2004. (Citado en la página 7).

[69] Lorena Pradenas, Boris Oportus, and VíCtor Parada. Mitigation of greenhouse gas emissions in vehicle routing problems with backhauling. *Expert Systems with Applications*, 40(8):2985–2991, 2013. (Citado en la página 6).

[70] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986. (Citado en la página 13).

[71] JR Quirdan. C4. 5: programs for machine learning, 1993. (Citado en la página 13).

[72] Glaydston Mattos Ribeiro and Gilbert Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & operations research*, 39(3):728–735, 2012. (Citado en la página 5).

[73] Sheldon Ross. Generating a random permutation. In *A first course in probability*, pages 439–440. Pearson, Upper Saddle River, New Jersey, 2010. (Citado en la página 50).

[74] Martin WP Savelsbergh. Local search in routing problems with time windows. *Annals of Operations research*, 4(1):285–305, 1985. (Citado en la página 6).

[75] Steven F Sawyer. Analysis of variance: the fundamental concepts. *Journal of Manual & Manipulative Therapy*, 17(2):27E–38E, 2009. (Citado en la página 13).

[76] Andreas Stenger, Daniele Vigo, Steffen Enz, and Michael Schwind. An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science*, 47(1):64–80, 2013. (Citado en la página 6).

[77] Carolin Strobl, Florian Wickelmaier, and Achim Zeileis. Accounting for individual differences in bradley-terry models by means of recursive partitioning. *Journal of Educational and Behavioral Statistics*, 36(2):135–153, 2011. (Citado en la página 13).

[78] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation science*, 31(2):170–186, 1997. (Citado en la página 7).

[79] Duygu Taş, Nico Dellaert, Tom Van Woensel, and Ton De Kok. Vehicle routing problem with stochastic travel times including soft time windows and service costs. *Computers & Operations Research*, 40(1):214–224, 2013. (Citado en la página 5).

[80] A Serdar Tasan and Mitsuo Gen. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, 62(3):755–761, 2012. (Citado en la página 6).

[81] O Toro, M Eliana, Z Escobar, H Antonio, E Granada, et al. Literature review on the vehicle routing problem in the green transportation context. *Luna Azul*, (42):362–387, 2016. (Citado en la página 5).

[82] P Toth and D Vigo. The vehicle routing problem. monographs. *Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics, Philadelphia*, 2002.

[83] Paolo Toth and Daniele Vigo. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, 15(4):333–346, 2003. (Citado en la página 7).

[84] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. SIAM, 2014. (Citado en la página 5).

[85] Alex Van Breedam. Improvement heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operational Research*, 86(3):480–490, 1995. (Citado en la página 7).

[86] Yoel Mederos Vargas. Propuesta para la evaluación ágil de soluciones del vrp. Master's thesis, Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba, May 2017.

[87] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research*, 40(1):475–489, 2013. (Citado en la página 5).

[88] Zheng Wang, Ying Li, and Xiangpei Hu. A heuristic approach and a tabu search for the heterogeneous multi-type fleet vehicle routing problem with time windows and an incompatible loading constraint. *Computers & Industrial Engineering*, 89:162–176, 2015. (Citado en la página 5).

[89] A. Mor y M. G. Speranza. Vehicle routing problems over time: a survey. 2018. (Citado en las páginas 1 y 5).

[90] Alina Fernández Arias y Sira María Allende Alonso. Modelos y métodos para el problema de enrutamiento de vehículos con recogida y entrega simultánea. 2013. (Citado en la página 5).

[91] Hiba Yahyaoui, Islem Kaabachi, Saoussen Krichen, and Abdulkader Dekdouk. Two metaheuristic approaches for solving the multi-compartment vehicle routing problem. *Operational Research*, pages 1–24, 2018. (Citado en la página 2).

[92] Esra Yigit and Fikri Gokpinar. A simulation study on tests for one-way anova under the unequal variance assumption. *Commun Fac Sci Univ Ankara, Ser A*, 1:15–34, 2010. (Citado en la página 57).

[93] Bin Yu and Zhong Zhen Yang. An ant colony optimization model: The period vehicle routing problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 47(2):166–181, 2011. (Citado en la página 6).