METAHEURÍSTICA ILS PARA LA SOLUCIÓN DEL PROBLEMA DE RUTEO DE VEHÍCULOS CON FLOTA PROPIA Y SUBCONTRATADA VRPPC

JOHN FREDY CASTAÑEDA LONDOÑO

Facultad de Ingeniería Industrial

Maestría en Investigación Operativa y Estadística

Universidad Tecnológica de Pereira

Pereira, Risaralda

2017

Información General

Titulo: Metaheurística ILS para la solución del problema de ruteo de vehículos con flota propia y subcontratada VRPPC

Área de investigación: Esta propuesta de investigación se enmarca en la línea de Optimización aproximada, por ser una propuesta que aporta una aplicación de los modelos cuantitativos, a problemas generados en los sistemas de transporte terrestre.

Materias asociadas a la investigación: Programación lineal entera mixta, heurísticas y Metaheurísticas.

Director: Eliana Mirledy Toro Ocampo. Ing. Industrial, Magíster en Ingeniería Eléctrica en la línea de Investigación operativa y Magíster en Investigación de Operaciones y Estadística, y Ph.D. en Ingeniería. Profesora Titular, Facultad de ingeniería industrial. Universidad Tecnológica de Pereira.
E-mail: elianam@utp.edu.co

CoDirector: Ramon Alfonso Gallego. Ing. Eléctricista, Magíster y Ph.D. en Ingeniería Eléctrica. Profesor Titular, Facultad de Ingenierías. Universidad Tecnológica de Pereira.
E-mail: ragr@utp.edu.co

Resumen

El problema de ruteo de vehículos con ruta propia y subcontratada *Vehicle Routing Problem with Private Fleet and Common Carriers* (VRPPC) es una variante del problema de ruteo de vehículos clásico con restricción de capacidad *Capacitated Vehicle Routing Problem* (CVRP), que consiste en encontrar rutas con el menor costo posible atendiendo la totalidad de clientes. El VRPPC se da cuando la demanda de los clientes supera la capacidad de los vehículos propios o por indisponibilidad de los mismos y se tiene que subcontratar la operación de algunas rutas o eventualmente todas las rutas. Se considera que este planteamiento del problema es relativamente nuevo de acuerdo a la revisión de literatura realizada en este trabajo, con un campo de aplicación en una industria del transporte cambiante donde la demanda de los clientes puede tener picos considerables que se hacen imposibles de atender con vehículos propios.

Una revisión a la literatura relacionada al problema es llevada a cabo con el fin de verificar el estado del arte y contribuciones en este tema. Además de lo anterior, se identifican vacíos de conocimiento en los cuales se pueden realizar aportes.

Este problema de ruteo de vehículos se define como un problema de optimización combinatorial de la clase NP - completo por su complejidad computacional. Para este tipo de problemas existen dos formas de encontrar su solución: el primero corresponde a los métodos exactos, y el segundo a los métodos aproximados. En este trabajo se desarrolla un algoritmo de optimización aproximado denominado *Iterated Local Search* (ILS) que implementa la búsqueda en vecindarios variables *Variable Neighborhood Search* (VNS). Es considerado en la primera etapa del algoritmo, la obtención de una solución inicial con base en la heurística de ahorros modificada, la cual resulta ser determinante en el desempeño global del algoritmo. En la segunda etapa se aplica la metaheurística ILS, la cual se fundamenta en conceptos de intensificación y diversificación, que son aplicados a través de la heurística VNS y un esquema apropiado de perturbación, respectivamente.

Una comprobación final al desempeño del algoritmo desarrollado se implementa usando instancias de referencia de la literatura. Los resultados de la metodología propuesta basada en la metaheurística ILS son comprobados con los obtenidos con métodos exactos, con el fin de validar la efectividad del método. Posteriormente son usadas instancias de gran complejidad matemática, donde los resultados obtenidos no presentan respuesta en la literatura especializada y se definen como nuevos *Best known solution* (BKS) para el problema estudiado.

Abstract

Vehicle Routing Problem with Private Fleet and Common Carriers (VRPPC) is a variant of Capacitated Vehicle Routing Problem (CVRP) that searches for lowest cost routes while serving a complete set of clients. VRPPC is most prominent when demand is higher than the number of owned vehicles or due to private fleet unavailability which leads to outsourcing operation of some or all routes. This problem statement is considered relatively new based on the literature review made in this work. The scope of this work is the always changing transport industry, where demand can reach significant peaks that are impossible to serve with a private fleet.

A literature review on the problem is made with the objective of verifying the state of the art and inputs on the subject. In addition to the above, gaps where contributions can be provided are identified.

This vehicle routing problem is defined as an NP-complete combinatorial optimization problem due to its computational complexity. For these types of problems, there are two approaches to find their solution: generation of exact optimization algorithms or generation of approximate optimization algorithms. An approximate optimization algorithm is developed in this work, named Iterated Local Search (ILS), which implements Variable Neighborhood Search (VNS). Procurement of an initial solution is considered in the first stage based on modified savings heuristic; this is a basis for the global performance of the algorithm. In the second stage, ILS metaheuristics is applied based on concepts of intensification and diversification. These concepts are applied through VNS heuristics and an appropriate perturbation scheme respectively.

A final verification on the performance of the developed algorithm is implemented using as reference literature examples. Results from the proposed methodology based on ILS metaheuristics are verified with results obtained from exact methods to validate the effectiveness of the proposed method. Subsequently, instances of great mathematical complexity are used. Results obtained are not answered in specialized literature and are defined as new BKS (Best Known Solution) for the problem under study.

Índice de cuadros

2.1. Comparativo modelos matemáticos 43

5.1. Variables generales ILS 60

5.2. Variables de diversificación del ILS 61

5.3. Variables de parámetros del problema 61

6.1. Resultados instancias estudiadas 81

6.2. Resultado Rutas 84

6.3. Valores de la función objetivo de las 13 instancias estudiadas 85

6.4. Tiempo en segundos de las 13 instancias estudiadas 86

6.5. Resultados instancias de gran complejidad matemática 88

6.6. Resultados de la función objetivo de las instancias estudiadas 89

6.7. Resultados de los tiempos en segundos de las instancias estudiadas 89

6.8. Resultado rutas instancias estudiadas 95

Índice de figuras

3.1. Gráfico ejemplo solución del VRPPC 45

4.1. Rutas atendidas por vehículos subcontratados 51
4.2. Ruta atendida por la flota propia 51
4.3. Ruta atendida con vehículo propio y ruta subcontratada 52
4.4. Ruta atendida únicamente con vehículo propio 52
4.5. Cálculo ahorro nodos flota subcontratada 55
4.6. Mejoramiento Or-opt 56

5.1. Esquema RVNS 65
5.2. Operador $Shift(\lambda, 0)$ 70
5.3. Operador $Swap(\lambda_1, \lambda_2)$ 72
5.4. Movimientos intra-ruta 74
5.5. Criterio de distancia 77

Índice de algoritmos

1. ILS 59
2. Función SolucionInicial 63
3. Función RVNS 66
4. Función Shift 69
5. Función Swap 71

Índice general

1. Introducción **10**

1.1. Planteamiento del problema 11

1.2. Delimitación del problema 12

1.3. Justificación 13

1.4. Objetivos 13

1.4.1. Objetivo General 13

1.4.2. Objetivos Específicos 14

1.5. Metodología 14

1.6. Resultados Esperados 15

2. Revisión del VRP y la variante VRPPC **16**

2.1. Antecedentes 16

2.2. Estado del Arte del VRPPC 20

2.3. Comparativo modelos matemáticos para el VRPPC 38

3. Modelo matemático para el VRPPC **44**

3.1. Consideraciones iniciales 44

3.2. Construcción del modelo matemático 45

4. Construcción de la solución inicial **49**

4.1. Algoritmo de ahorros 49

4.1.1. Cálculo del ahorro 50

4.1.2. Procedimiento de inserción (Descripción para implementación) 52

4.1.3. Algoritmo de ahorros para la flota subcontratada 54

4.1.4. Etapa de mejoramiento 55

5. Algoritmo ILS para la solución del VRPPC **57**

5.1. Solución inicial 62

5.2. Random Variable Neighborhood Search (RNVS) 64

5.3. Operadores Inter - Ruta 67

5.3.1. Shift 67

5.3.2. Swap 68

5.3.3. K-Shift 68

5.4. Operadores Intra - Ruta 73

5.5. Perturbación 75

5.5.1. Criterio de Distancia 76

6. Análisis de resultados **78**

6.1. Codificación 78

6.2. Parámetros e implementación del algoritmo 79

6.2.1. Instancias usadas en el análisis de resultados 80

6.2.2. Análisis de resultados para instancias de mediana y gran complejidad matemática 87

7. Conclusiones **96**

Capítulo 1

Introducción

Los problemas del *Vehicle Routing Problem* (VRP) tiene sus orígenes a partir del planteamiento del problema del *Travelling Salesman Problem* (TSP) que fue propuesto por primera vez por Flood en 1956, que se describe como un agente viajero el cual debe visitar una cantidad *"n"* de ciudades, de forma tal que termine en el mismo punto de partida. Todo el recorrido debe realizarse al mínimo costo (o proporcional a la distancia mínima recorrida).

Este problema es de simple planteamiento, pero no así su solución. Analizando la cantidad de rutas posibles para el enrutamiento que tiene este único agente, es calculado como el número de permutaciones para $n-1$ trayectorias a las n ciudades o lugares $(P_{n-1} = (n-1)!)$. Si se considera la aproximación de que las distancias entre ciudades es igual en la ida que en la vuelta, entonces, el numero de posibles rutas se reduce a la mitad: $^{(n-1)!}/_{2}$. Como ejemplo, se puede considerar el caso para el que se tienen 50 ciudades o lugares por visitar por parte del agente, el número de rutas posibles se calcula en $3{,}0414 * 10^{62}$. Sí intentáramos probar todas estas soluciones (incluidas las infactibles) con el fin de encontrar cual es la mejor factible, un computador personal actual con una capacidad de procesamiento mediana de 9,4 Ghz, podría (como ejemplo) construir una solución y verificar si es factible en $1 * 10^{-10}$ segundos, por lo que para probarlas todas tardaría $3{,}0414 * 10^{52}$segundos. Para entender esta cifra, se la puede comparar con la edad del universo, la cual es de $4{,}7304 * 10^{20}$ segundos. Este tipo de problemas aumenta su complejidad de manera exponencial a medida que aumenta el número n de ciudades o lugares por visitar. A esto se le conoce a menudo como explosión combinatorial. Las técnicas de solución para este problema no prueban todas las soluciones matemáticas, sino que a través de algoritmos se delimita el problema

a las soluciones factibles y luego a las soluciones candidatas a óptimo. De esta manera se pueden encontrar buenas soluciones en tiempos polinomiales.

Sería en el año 1959 cuando Dantzing y Ramser proponen una variación al problema del TPS, donde ya se tienen en cuenta la utilización de m vehículos (en remplazo de agentes) con restricciones de capacidad. A este problema se le conoce como CVRP. Posteriormente, se fueron generando nuevos planteamientos al problema con todo tipo de nuevas variantes como la utilización de ventanas de tiempo, restricciones en las distancias recorridas, múltiples depósitos, entre otros que se describirán en la sección 2.1.

El VRPPC consiste en un problema de optimización combinatorial en el cual se tiene un número de clientes n los cuales deben ser atendidos por un número de vehículos m, que tienen restricción de capacidad de carga. Cuando la capacidad de los vehículos es superada por la demandada de los clientes, hay vehículos en mantenimiento o en indisponibilidad, se considera la utilización de vehículos subcontratados, los cuales inician en el depósito y terminan su recorrido en el último cliente atendido. Chu en 2005[4], plantea el VRPPC proponiendo para su solución un modelo matemático y una heurística para su solución. Esta heurística se desarrollara en el capítulo 4 de este trabajo, del cual se plantea como inicializador del VRPPC.

Para la solución del VRPPC se propone una técnica metaheurística denominada *Iterated Local Search* (ILS) en la cual se generan vecindarios variables VNS y posteriormente son aplicados procedimientos de diversificación. El algoritmo para esta técnica de solución se describe en varias etapas: Generación de soluciones iniciales, búsqueda local y perturbación. El desarrollo de esta técnica con la descripción de los algoritmos se realizará en el capítulo 5.

El algoritmo desarrollado será probado usando instancias de la literatura especializada. Como criterios de comparación se usa la calidad de las soluciones y tiempo computacional. Con base en los resultados se concluirá acerca del desempeño del algoritmo. Inicialmente se utilizan instancias de baja complejidad matemática, resueltos con métodos exactos y de esta forma concluir sobre la confiabilidad de los resultados. Posteriormente son estudian instancias de gran complejidad matemática.

1.1. Planteamiento del problema

En términos de competitividad económica, el problema de ruteo de vehículos tiene gran importancia debido a las implicaciones en costos que tienen los procesos asociados al desempeño logístico y es de gran interés para los sectores productivos del país, lo que implica que tiene un efecto directo de tipo macroeconómico. Los negocios de transporte como aprovisionamiento y distribución de

productos o servicios juegan un papel importante en términos de productividad, teniendo en cuenta que los costos asociados al transporte son un porcentaje importante del costo final de los productos y servicios. El transporte de productos y servicios puede atenderse de dos diferentes formas, una con vehículos propios y otra con vehículos de terceros o subcontratados. El hecho de que se puedan utilizar vehículos contratados, permite que la cantidad de vehículos de la flota propia permanezca constante, permitiendo eficiencia en los costos de operación.

En el negocio de transporte para el aprovisionamiento o distribución de bienes o servicios, existe una variable en su implementación operacional, la cual define dos tipos de flota, una propia y la otra subcontratada o suministrada por un tercero, esto permite garantizar la atención de la totalidad de los clientes. La utilización de vehículos de la flota subcontratada tendrá un costo mayor, por lo que se hace necesario encontrar un equilibrio entre los vehículos requeridos en la flota propia y la subcontratada. Esta práctica de utilizar estos dos tipos de flotas, permite tener una ventaja operacional, ya que existen ciertos casos en donde la flota propia no esta todo el tiempo disponible por fallas en los vehículos o mantenimientos, incapacidades o licencias en los conductores u operarios y altos costos en el pago de horas extras a los mismos generado por las situaciones en donde la demanda es mayor a la capacidad de la flota.

1.2. Delimitación del problema

Para esta implementación, el VRPPC tendrá las siguientes características:

1. Se tendrán restricciones de capacidad para los vehículos.
2. Flota de vehículos homogénea.
3. Se considera el problema como simétrico.
4. Las distancia entre los clientes se toma como la distancia euclidiana entre las ubicaciones.
5. Clientes con demanda constante.
6. Único depósito.
7. Los vehículos propios deberán retornar al depósito al final del recorrido.
8. Los vehículos subcontratados inician en el depósito pero no requieren de su regreso.

1.3. Justificación

En el desarrollo del proyecto, se pretende dar solución al problema de ruteo de vehículos con flota propia y subcontratada VRPPC, donde los dos tipos de flota serán homogéneas. Esta clase de problema tiene muchos tipos de aplicación en la industria del transporte y logística como son el transporte de mercancías, entrega de productos, rutas de transporte de personas, transporte aéreo de personas y de mercancías.

El VRPPC se plantea como un problema en el que existe la posibilidad de una demanda variable en los clientes, la cual en cualquier momento puede exceder la capacidad de los vehículos propios y que hace necesario utilizar vehículos contratados externamente, por lo que se hace ineludible tener en cuenta el problema de ruteo de vehículos con restricción de capacidad CVRP. Se considera también que se tendrá un único deposito, al cual todos los vehículos de la flota propia regresarán al terminar la ruta y los vehículos de la ruta subcontratada no tendrían que hacerlo. La solución de este tipo de problemas es de alta complejidad matemática y computacional, y se consideran como NP – Duros (NP-Hard).

Se buscará dar solución al VRPPC por medio de una técnica metaheurística denominada búsqueda local iterativa o ILS. Con la implementación se busca realizar la verificación de rendimiento computacional y compararlo con otro tipo de implementaciones con técnicas de solución exacta y otras metaheurísticas de búsqueda local. Para esto, se realizará una revisión del estado del arte, donde se verificarán los modelos exactos propuestos y se utilizarán instancias y modelos reportados en la literatura especializada, para la comparación de resultados.

Con respecto a las implementaciones de metaherísticas utilizando la metodología ILS, no se encuentran referentes para el VRPCC. Para este caso, fue adaptada la metodología propuesta por Penna, Subramanian, & Ochi, 2013. Este tipo de procedimiento no ha sido utilizado para encontrar soluciones al problema de ruteo VRPCC.

1.4. Objetivos

1.4.1. Objetivo General

Proponer una técnica metaheurística denominada ILS para la solución del problema de ruteo con flota propia y subcontratada VRPPC, considerando un único depósito.

1.4.2. Objetivos Específicos

- Realizar una revisión del estado del arte para la solución al problema de ruteo con flota propia y subcontratada VRPPC utilizando técnicas metaheurísticas
- Plantear un modelo matemático para la solución del problema de ruteo VRPPC con un único depósito y flota homogénea.
- Implementar un algoritmo para la ejecución de la técnica metaherística basada en la metodología búsqueda local iterativa ILS.
- Validar los resultados con los obtenidos con métodos exactos, con instancias de baja complejidad matemática.
- Estudiar la metodología de solución usando instancias de la literatura de gran complejidad matemática.
- Elaboración de documentos (tesis y articulo donde se presente el estudio realizado).

1.5. Metodología

Este proyecto será efectuado de manera metodológica siguiendo el desarrollo de cada una de sus etapas que se muestran a continuación:

Etapa 1: Realizar una revisión del estado del arte para el problema de ruteo con flota propia y subcontratada VRPCC, empleando bases de datos académicas donde se encuentran revistas indexadas nacionales e internacionales, memorias de congresos, tesis de maestrías y doctorados. Se deberá estudiar las metodologías de solución con el fin de determinar su eficiencia computacional y generar un punto de referencia comparativo.

Etapa 2: Analizar los modelos matemáticos planteados para el problema VRPCC, encontrados mediante la revisión del estado del arte. Estos modelos permitirán construir una codificación del problema, definiendo sus restricciones y función objetivo. También permitirá definir la complejidad matemática del problema.

Etapa 3: Plantear un modelo matemático para la solución del modelo propuesto.

Etapa 4: Plantear un esquema de codificación para la solución del problema propuesto. Plantear una heurística constructiva para la identificación de soluciones iniciales. Plantear un método de solución al problema usando la metaheuristicas ILS.

Etapa 5: Realizar análisis de resultados utilizando instancias de baja y media complejidad matemática para la validación del modelo con resultados obtenidos aplicando métodos exactos. Análisis de instancias de gran complejidad matemática.

Etapa 6: Publicar resultados obtenidos en revistas indexadas.

1.6. Resultados Esperados

Se espera que la metodología desarrollada permita encontrar buenas soluciones para el problema del VRPCC, medidos en tiempo y calidad de la solución, al ser comparados con los resultados obtenidos con métodos exactos. Finalmente se valorará la factibilidad de aplicarse a instancias de gran complejidad matemática.

Capítulo 2

Revisión del VRP y la variante VRPPC

2.1. Antecedentes

El VRP es un problema clásico de optimización, el cual desde que se planteó por primera vez por [7] como una variación del problema del agente viajero *Travelling Salesman Problem* (TSP), ha sido uno de los más estudiados. El análisis del VRP inicia como una generalización del TSP con la utilización de varios vehículos, es decir, se considera el caso análogo del TSP con múltiples viajeros. Cambiando los viajeros por vehículos se obtiene el VRP. Teniendo en cuenta la restricción de capacidad de los vehículos, el problema se denomina CVRP.

El planteamiento del VRP donde se considera la restricción de carga CVRP se plantea definiendo un conjunto de clientes $i = \{1, 2, \ldots, n\}$ donde $i = 0$ corresponde al depósito. Las trayectorias que conectan a un cliente i hasta un cliente j donde $i < j$ son llamadas arcos y están asociadas a variables como distancia, tiempo de recorrido o se puede generalizar como el costo de desplazamiento. Este costo se define como c_{ij}^k que representa el costo asociado de atravesar el arco (i, j) en el vehículo k donde $k = \{1, 2..., m\}$, con m como la cantidad de vehículos disponibles. Para el caso donde el problema se considera asimétrico, se tiene que $c_{ij}^k \neq c_{ji}^k$ y simétrico cuando $c_{ij}^k = c_{ji}^k$. La definición anterior puede describirse como un grafo completo representado como $G = (V, A)$ donde $V = i/i = 1, 2, \ldots, n$, es decir, un conjunto que contiene a los clientes, y donde A corresponde al conjunto de arcos que tiene asociado un valor de c_{ij}^k. La solución del problema CVRP radica en encontrar una ruta para visitar a cada cliente al menor costo teniendo en cuenta las res-

tricciones: (1) un vehículo tiene una única ruta que debe finalizar en el depósito, (2) cada cliente se visita una sola vez y (3) la demanda total en cada ruta servida por un vehículo k no debe exceder su capacidad Q. Para construir una función objetivo, se debe crear una variable binaria x_{ij} que corresponde a la utilización de un arco entre los clientes (i, j). Esta función objetivo será minimizar la sumatoria de $c_{ij}^k * x_{ij}$. A partir de esta definición general, se pueden desprender otros análisis asociados a componentes de las variables del problema, como son la utilización de vehículos con diferentes capacidades, clientes con diferentes tipos de recursos solicitados, ventanas de tiempo, variables asociadas con procesos estocásticos y diferenciación en el retorno a los depósitos.

Después de plantear el problema CVRP, este ha tenido muchos enfoques de solución, donde los mas importantes fueron los planteados para ser solucionados por medio de los algoritmos exactos basados en Branch-and-Bound, Branch-and-Cut (BC), posteriormente se presentarían metodologías de solución mas sofisticadas como el robust Branch-Cut-and-Price (BCP) y el Set Partitioning (SP) con cortes adicionales. Con el desarrollo de técnicas heurísticas y metaheurísticas para la solución de problemas de optimización combinatoríal, se definieron dos grupos metodológicos: (1) heurísticas constructivas con fases de mejoramiento, (2) metaheurísticas con empleo de mecanismos de memoria para la búsqueda, intercambios de partes de las soluciones y perturbaciones [17].

Con variaciones al problema del CVRP ya planteado, se crean nuevas familias de definiciones del problema del VRP. Las siguientes son los planteamientos más comunes a partir del CVRP :

VRP Asimétrico (Asymmetric cost matrix - ACVRP)

Como se menciona anteriormente, este problema se define cuando las distancias de (i,j) son diferentes a las distancias (j,i). Esta situación implica que la matriz de costos $\mathbb{C}$ sea asimétrica, lo que se traduce en un posible incremento en el esfuerzo computacional para resolver el problema. Las restricciones de capacidad se mantienen idénticas al del CVRP.

VRP abierto (Open VRP - OVRP)

Para este problema, se tiene como definición, que un vehículo con una ruta asignada pueda terminar su recorrido en cualquier cliente sin la necesidad de tener que volver al depósito. Esta formulación tiene aplicaciones reales como por ejemplo empresas de entrega de paquetes de transporte aéreo, donde los aviones no deben regresar al depósito. Desde la formulación del CVRP, se puede obtener el planeamiento del OVRP mediante una modificación en la matriz de costos $\mathbb{C}$, donde los valores $c_{io} = 0, \forall i \in V$ [17].

VRP con restricción de distancias (Distance-Constrained VRP - DCVRP)

Este problema define la necesidad de imponer un tope de valor a la distancia que puede tener una ruta, estas distancias son tomadas generalmente de la evaluación de distancias euclidianas.

VRP con flota heterogénea (Heterogeneous fleet VRP - HVRP)

Este problema se define por la utilización de varios tipos de vehículos con diferentes características, ya sea por la carga máxima de cada tipo de vehículo, por las distancias que pueden recorrer o la especialidad de algún tipo de carga que requiera de vehículos especiales. El hecho de que existan varios tipos de vehículos hace que el problema del VRP aumente su complejidad computacional. El problema se plantea como una variación del CVRP, teniendo en cuenta la flota heterogénea $M = \{m_1, m_2, ...m_n\}$, donde M son el conjunto de vehículos heterogéneos y $Q = \{q_1, q_2, ..., q_n\}$, representa el conjunto de capacidades de carga de cada uno de los vehículos m. Adicionalmente, es necesario tener en cuenta que cada vehículo tendrá un costo diferenciado de acuerdo a sus características, por lo tanto, se tendrá que incluir una nueva variable en la función objetivo que tenga en cuenta el costo fijo de la utilización de cada vehículo en relación a su distancia recorrida. Todas las demás consideraciones y restricciones se conservan.

VRP con múltiples depósitos (Multiple Depots VRP - MDVRP)

Cuando se habla de múltiples depósitos, el problema consiste en encontrar rutas para cada vehículo eligiendo de antemano el depósito para cada vehículo disponible y este deberá iniciar y terminar en el mismo depósito. Este problema se propuso inicialmente para solucionar un problema con demanda probabilística para múltiples depósitos [19]. Se tendrá entonces un conjunto $D = \{D_1, D_2, ..., D_n\}$ correspondiente a cada depósito y estos tendrá un conjunto $W = \{W_1, W_2, ..., W_n\}$ correspondiente a la capacidad de almacenamiento de cada depósito D. Los despachos desde un depósito D_n no deben de exceder la capacidad W_n. El problema puede plantearse inicialmente con una cantidad finita de depósitos y vehículos, los cuales como se dijo anteriormente, se asignarán desde el inicio a cada uno de los depósitos. Existen también variantes al anterior planteamiento en el cual se puede construir una restricción que permita hallar el número óptimo de depósitos requeridos, así como el número de vehículos y podría además permitirse que los vehículos puedan terminar en un depósito diferente del que partieron.

VRP con recogida y entrega (Pickup-and-delivery VRP - PDVRP)

En este caso, el planteamiento del problema se da cuando a los clientes se les puede entregar un producto o ellos pueden retornar el mismo, haciendo que aumente la carga del vehículo. Después de visitar un cliente, los vehículos podrían salir con una capacidad menor de carga que con la que llegaron al cliente. Esta carga puede ya sea, ser regresada al depósito o entregada a otro cliente. Por lo tanto, es necesario definir un nuevo conjunto de clientes $V^{'} = \{i/i = 1, 2, ..., n\}$. La carga que entregan los clientes de V' está dada por $Q' = \{q'_1, q'_2, ..., q'_n\}$. Este problema también es conocido y generalizado como Problema de ruteo con carga y entrega simultanea (Vehicle Routing Problem with Simultaneous Pickup and Delivery - VRPSPD).

Existe una variación a este problema, denominado *VRP with Backhauls,* a diferencia del anterior, los vehículos deberán primero visitar a los clientes que se les entregará algún producto o mercancía, y posteriormente deberán visitar a los clientes que desean retornar mercancía. Por esta razón, se necesitará de un planteamiento el cual divida los clientes en los dos grupos anteriormente mencionados. Este problema también es denominado como *one-to-many-to-one,* que significa que toda la demanda de mercancía estará inicialmente localizada en un depósito como un conjunto y la mercancía que se debe cargar como otro conjunto secundario. [11].

VRP con entrega dividida (Split-delivery VRP - SDVRP)

Esta variación del problema que fue planteado por primera vez por [8]. Surge de la relajación del planteamiento general, en la cual se permite que se realicen entregas parciales a los clientes y hace que sea necesario que varios vehículos puedan visitar un mismo cliente para completar la entrega, por lo tanto, en el modelo, se deberá eliminar la restricción de que cada cliente será visitado una única vez. Este tipo de problema puede aplicarse en los casos donde los clientes tengan demandas mayores a la capacidad de los vehículos.

VRP con ventanas de tiempo (VRP with TimeWindows - VRPTW)

Esta es una generalización del problema de ruteo con restricción de capacidad CVRP, donde la diferencia se encuentra en que existirán intervalos de tiempo denominadas generalmente como ventanas, en las cuales los clientes solo pueden ser visitados al iniciar estos intervalos y el vehículo deberá permanecer en la ubicación del cliente hasta que termine la ventana de tiempo. Existen dos variantes llamadas ventanas de tiempo suaves y duras. En la primera, se considera la relajación de la ventana de tiempo en la que el vehículo puede llegar después del inicio de la ventana de tiempo, pero por cada unidad de tiempo de retraso tendrá una penalización en la función de costos.

La segunda ventana (Dura), no se permite la entrega o arribo del vehículo después de iniciada la ventana de tiempo, por lo que deberá llegar antes de iniciada la ventana para estar listo para el servicio. [6]

la formulación de este problema consiste en representar el grafo completo del CVRP donde el depósito se representa por el par de nodos 0 y $n+1$, y se define la primera ventana de tiempo como $[a_0, b_0] = [a_{n+1}, b_{n+1}]$ donde el valor a representa la salida más temprana posible del depósito y b su último posible arribo. También se define que la demanda del depósito y el tiempo del servicio es cero, es decir, $d_0 = d_{n+1} = 0$, $s_0 = s_{n+1} = 0$. Finalmente, es necesario tener en cuenta el número de vehículos a utilizar, por lo que en caso se crea el arco $(0, n+1)$ que indica que el vehículo no se mueve del depósito, con costos y tiempos iguales a cero. [6].

VRP Estocástico (Stochastic VRP)

El VRP estocástico surge de la consideración probabilística en el tiempo que se le da a variables como la demanda de los clientes y el tiempo en el recorrido entre clientes debido por ejemplo al tráfico o condiciones externas en los recorridos. Debido a estas características, este tipo de problemas inicialmente solo ha podido implementarse a escala de pocas instancias, ya que su complejidad computacional lo hace muy difícil de resolver, además, las técnicas para solucionarlo son complejas de construir y evaluar.[10].

2.2. Estado del Arte del VRPPC

El problema de ruteo con flota propia y flota subcontratada VRPPC, surge como una variante del VRP para considerar los casos en los que, por fluctuaciones de demanda de los clientes o problemas en la operación de los vehículos, la flota propia no tiene la capacidad de atender toda la demanda de los clientes. Los primeros planteamientos de este problema se hallan en [1], donde la propuesta fue motivada por un problema de planeación de entregas o envíos de una firma comercial, donde los vehículos deben hacer largos recorridos para llegar a los clientes, por lo que se debió considerar en la solución incluir el pago y utilización de vehículos subcontratados. Posteriormente en [13], se propone implementar decisiones periódicas para considerar la cantidad de vehículos en las flotas propias y subcontratadas, mediante un modelo que considera el área geográfica con demandas diarias aleatorias con un único depósito, y debido a que la localización de los clientes cambia cada día, se divide el área geográfica por sectores. Para la solución, se utiliza una formulación de un *Facility Location Problem* (FLP), donde se obtienen las rutas de las flotas.

A continuación se realizara un análisis de las publicaciones más importantes relacionadas con VRPPC:

A heuristic algorithm for the truckload and less-than-truckload problem [4]

Autores: Ching-Wu Chu

Fecha de publicación: 2005

Objetivo: Desarrollar algoritmo para una técnica heurística que permita minimizar la función de costos del problema de ruteo con vehículos propios y subcontratados

Resumen: El envío de bienes desde un depósito o almacén a consumidores locales es un reconocido e importante problema de transporte y logística. Se considera que la demanda de los clientes es variable y en determinado momento puede presentar que esta supere la capacidad de los vehículos propios, por lo tanto, se hace necesario considerar subcontratar vehículos externos para atender esta demanda. En este artículo, se considera que se tiene un número fijo de vehículos con una capacidad limitada, con un solo depósito y los clientes tienen demandas conocidas. Por lo tanto, para la solución de este problema, se plantea un algoritmo para la implementación de una heurística que permita minimizar la función de costos.

El algoritmo para la implementación de la heurística es llamado TL-LTL y se divide en 3 pasos:

1. Selección: Se encuentra el valor de la demanda que supera la capacidad de los vehículos propios, posteriormente se ordenan los clientes en orden ascendente de acuerdo al costo de ser visitado por un vehículo de la flota subcontratada. Finalmente se suma la demanda de cada cliente hasta que esta sea inmediatamente superior a la demanda insatisfecha por la flota propia. Estos clientes serán atendidos por la flota subconcontratada y los restantes por la flota propia.

2. Solución inicial: El algoritmo de ahorros es implementado con dos modificaciones. La primera modificación está en el intercambio del criterio de distancia por el de costo. La segunda modificación es un cambio en el cálculo del ahorro.

3. Refinamiento: Este refinamiento es aplicado a la solución inicial obtenida en el punto anterior. El mejoramiento se obtiene implementando sucesivos intercambios inter-ruta e intra-ruta de arcos.

Modelo: El modelo matemático se formula bajo los siguientes supuestos: (1) se considera un solo depósito donde todos los vehículos inician y terminan en él. (2) La demanda de los consumidores

es conocida y no debe exceder la capacidad de los vehículos. (3) Los consumidores solo pueden ser atendidos por un solo vehículo que puede ser propio o subcontratado. (4) los vehículos se limitan a la entrega únicamente. (5) La función de costos se construye a través de los costos fijos y variables de la operación de los vehículos. La formulación del modelo se define como:

$$min\ z = \sum_{k}^{m} FC_k + \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n} C_{ijk}X_{ijk} + \sum_{i}^{n} CL_iL_i \tag{2.1}$$

Sujeto a:

$$\sum_{k}^{m} Y_{0k} = m\ (k = 1, ..., m) \tag{2.2}$$

$$\sum_{k}^{m} Y_{ik} + L_i = 1\ (i = 1, ..., m) \tag{2.3}$$

$$\sum_{i}^{n} q_iY_{ik} \leq Q_k\ (i = 1, ..., n;\ k = 1, ..., m) \tag{2.4}$$

$$\sum_{j}^{n} X_{ijk} = Y_{ik}\ (i = 1, ..., n;\ k = 1, ..., m) \tag{2.5}$$

$$\sum_{j}^{n} X_{jik} = Y_{ik}\ (i = 1, ..., n;\ k = 1, ..., m) \tag{2.6}$$

$$\sum_{j}^{n} X_{ijk} \leq |S| - 1\ \forall\ S \subseteq \{2, ..., n\}\ (k = 1, ..., m) \tag{2.7}$$

donde las variables se describen así:

FC_k :Corresponde a los costos fijos del vehículo k

C_{ijk} :Son los costos del vehículo k viajando del cliente i al j

CL_i :Costo asociado a la utilización de un vehículo de la flota subcontratada que atiende al cliente i

$X_{ijk} \in \{0,1\}$; $Y_{ik} \in \{0,1\}$;

$L_i \in \{0,1\}$

$(i = 1, ..., n;\ j = 0; ...n;\ ; k = 1, ..., m)$;

$i : \{i = 0, ..., n\}$, *Indica el conjunto de clientes, donde cero denota el depósito.*

$j : \{i = 0, ..., n\}$, *Indica el conjunto de clientes.*

$k : \{i = 1, ..., m\}$, *Indica el conjunto de vehículos.*

n : *El número de clientes.*

m : *El número de vehículos.*

A heuristic for the routing and carrier selection problem [2]

Autores: Marie-Claude Bolduc, Jacques Renaud , Fayez Boctor

Fecha de publicación: 2007

Objetivo: Proponer una heurística mejorada para encontrar mejores resultados del problema de ruteo con flota de vehículos propios y subcontratados.

Resumen: Considerar el problema de seleccionar simultáneamente vehículos entre las flotas propias y subcontratadas, para atender a los clientes o consumidores. Este problema anteriormente no se le ha dedicado mucha atención. En este artículo se muestra como obtener mejores resultados utilizando una heurística con un método simple de implementación.

Modelo: Se propone una heurística llamada SRI (Selection, routing y improvement) que realiza los siguientes pasos: (1) selecciona los clientes que serán atendidos por la flota subcontratada. (2) construye una primera solución a través de una modificación del algoritmo de ahorros de Clarke and Wright. (3) mejorar la solución obtenida aplicando el 4-opt*. (4) se construye una nueva solución inicial de manera idéntica a la anteriormente obtenida (5) Se mejora la segunda solución aplicando el 4-opt*. Por lo tanto, este algoritmo utiliza dos soluciones iniciales que incrementaría las posibilidades de encontrar buenas soluciones.

Después de obtenidas estas soluciones iniciales, se realizan un procedimiento de mejoramiento final para obtener la solución final, que consiste en aplicar un intercambio de clientes por medio de un procedimiento denominado $\lambda-$*interchange* que fue propuesto por Osman (1993), en el que se toman dos rutas simultáneamente y realizando todas las posibles transferencias e intercambio de clientes a λvértices. Esta implementación considera un valor de $\lambda = 2$.

A Perturbation Metaheuristic for the Vehicle Routing Problem with Private Fleet andCommon Carriers [3]

Autores: M.-C. Bolduc, J. Renaud, F. Boctor and G. Laporte

Fecha publicación: 2008

Objetivo: El objetivo de este artículo es consiste en proponer una metaheurística con operadores de perturbación para diversificación, que permita resolver el problema de enrutamiento de vehículos con flota propia y subcontratada VRPPC.

Resumen: Este artículo describe una nueva metaheurística para resolver el VRPPC, el cual utiliza un procedimiento de perturbación y fases de mejoramiento, también implementa intercambios entre conjuntos de clientes atendidos por la flota propia o la flota subcontratada.

Modelo: la función objetivo es definida de la forma:

$$Min\ z = \sum_{k}^{m} f_k y_{0k} + \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n} C_{ijk}X_{ijk} + \sum_{i}^{n} e_i z_i \tag{2.8}$$

donde:

$$x_{ijk} = \begin{cases} 1 & \text{Si el vehiculo } k \text{ visitó } j \text{ después de } i \\ 0 & \text{De lo contrario} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \text{Si el vehiculo } k \text{ visitó } i \\ 0 & \text{De lo contrario} \end{cases}$$

$$z_i = \begin{cases} 1 & \text{Si el clilente } i \text{ es asignado a la flota propia} \\ 0 & \text{De lo contrario} \end{cases}$$

u_{ik}: Limite superior de la carga del vehículo *k* al dejar el cliente *i*

Se desarrolla una metaheurística que implementa perturbaciones para encontrar mejores soluciones, esta metaheurística la llaman RIP (randomized construction – improvement - perturbation) que consiste en la construcción de una solución inicial con un algoritmo de ahorros aleatorizado que posteriormente tendrá dos etapas de mejoramiento a través de la implementación de algoritmos de intercambio de clientes entre rutas como el 4-opt*. Seguido, se aplica una versión limitada el procedimiento λ- *interchange* (con $\lambda = 2$), donde se aplica no a 2 clientes, sino que opera sobre dos cadenas $(i_s + i_{s+1}, i_{s+2})$ y (j_t, j_{t+1}, j_{t+2}). Se dice que es limitada debido a que los movimientos

entre cadenas se permiten únicamente veinticinco. Seguido, se aplica un procedimiento de mejoramiento denominado add-drop, el cual repite los movimientos: (i) Transferir dos usuarios de la flota privada o propia a la subcontratada (ii) Se intercambia un cliente interno con uno externo. Finalmente se aplica la etapa de perturbación, la cual consiste en intercambiar pares de clientes, donde hay un parámetro de control entre cero y uno. Cada par pertenece ya sea a dos clientes internos (igual depósito) de dos diferentes rutas o de un cliente interno y uno externo.

Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier [16]

Autores: J-Y Potvin and M-A Naud

Fecha de publicación: 2011

Objetivo: Minimizar los costos fijos asociados a la utilización de flota privada y flota subcontratada, utilizando una heurística de búsqueda tabú con vecindario estructurado basado en la rotación de clientes enlazados a través de las rutas, denominado cadena de eyección.

Resumen: Se implementó una metaheurística basada en la metodología de búsqueda tabú en la que se ordenan los clientes de acuerdo a una ponderación para asignarlos a la flota subcontratada y a los restantes clientes se le crearán rutas para ser atendidos por la flota propia a través de una heurística basada en los costos de viaje, estas rutas serán la solución inicial que tendrá una fase de doble mejoramiento a través de la implementación de una búsqueda tabú en un vecindario estructurado. Estas soluciones se almacenan como un conjunto de atributos que corresponden a las mejores soluciones del vecindario y que marcaran las posiciones que no son tabú para los movimientos de la búsqueda.

la heurística basada en búsqueda tabú con cadenas de eyección se describe mediante los siguientes procedimientos: (1) asignación de los clientes a la flota subcontratada: Sí la demanda total excede la capacidad de la flota propia, se procede con esta asignación, que se obtiene ordenando los clientes en orden ascendente de acuerdo a la relación e_i/q_i, donde e_i corresponde a la utilización de un vehículo de la flota subcontratada para visitar el cliente i y q_i corresponde a la demanda del cliente i. Este permite que los clientes con el costo más bajo por unidad de demanda sean asignados a la flota subcontratada. (2) Soluciones iniciales: Son creadas rutas para atender a los clientes que no fueron asignados a la flota subcontratada. Para crear estas rutas se utilizan dos tipos de heurísticas aleatorias como *menor costo de inserción (Least-cost insertion)* e *inserción convexa basada en cascaras (Convex hull-based insertion).* (3) Vecindarios: Los vecindarios estructurados son usados a través de toda la ejecución de la búsqueda tabú.

Modelo: El modelo matemático presentado es el mismo al propuesto en [3]. Conservando las mismas restricciones, pero permitiendo una relajación del problema permitiendo infactibilidad desde el sobrepaso de la capacidad de los vehículos.

Se define una nueva función objetivo como: $g(s) = f(s) + \alpha q(s)$, donde $f(s)$ es la solución de la función objetivo original, α es un parámetro de peso positivo y $q(s)$ corresponde a la penalización por exceder la capacidad de carga del vehículo. La heurística basada en búsqueda tabú con cadenas de eyección utiliza los siguientes pasos: (1) asignación de los clientes a la flota subcontratada. (2) Soluciones iniciales. (3) Vecindario. Estos vecindarios son construidos a través de cadenas de eyección que se aplica a situaciones particulares del proceso. Estas cadenas de eyección consisten en un procedimiento de transferir o rotar clientes de un vecindario a otro. Básicamente, un cliente es expulsado de una ruta e insertado en otra, lo que fuerza a un cliente de esta última ruta a expulsar un cliente. El largo de la cadena estará limitado por el número de rutas y puede ser cíclico o no. Finalmente se aplica el algoritmo de búsqueda tabú, donde sí un cliente i es removido desde una de las rutas de algún vehículo k, estará prohibido realizar el movimiento inverso (re-insertar) este mismo cliente a la misma ruta) en las próximas iteraciones.

A Lagrangian Heuristic for the Vehicle Routing Problems with the Private Fleet and the Common Carrier [12]

Autores: Huang, Kuancheng and Hsu, Cheng-Po

Fecha de Publicación: 2011

Objetivo: Estudiar el VRPPC, desarrollando un algoritmo de una heurística, teniendo un balance entre carga computacional y calidad de la solución. Este algoritmo es desarrollado en base a modelamientos matemáticos clásicos.

Resumen: El VRPPC es inicialmente formulado en forma de un conjunto que abarca todo el problema, y una relajación Lagraniana es usada como eje principal en el diseño del algoritmo iterativo. Además, es utilizado un concepto similar al de generación de columnas que es usado para actualizar el espacio de soluciones con un conjunto parcial de rutas. Basado en un experimento numérico, la calidad de la solución del algoritmo es estable, con tiempos computacionales aceptables bajo el más alto entorno dinámico.

Modelo: Se inicia con la formulación matemática realizada en [3][Página:24] considerando unicamente flota heterogénea. Esta formulación es transformada en base al planteamiento del SCP (*Set Coving Problem*) el cual es una formulación mucho más sencilla:

$$min \ \sum_{r}^{R} C_r X_r + \sum_{i}^{n} p_i z_i \tag{2.9}$$

Sujeto a:

$$\sum_{r \in R} a_{ir} x_r + z_i \geq 1 \forall i \in I 0 \tag{2.10}$$

$$\sum_{r \in R} x_r \leq m \tag{2.11}$$

a_r, x_r, z_i :***Binarios*** $\forall r \in R, i : 1, ..., n$

r: indice de rutas; C_r: costo fijo de la ruta r; p_i: Costo fijo de uso de un vehículo subcontratado.

La función objetivo minimiza los costos de las rutas de la flota propia y subcontratada. La primera restricción asegura que los clientes sean atendidos por algún vehículo de cualquiera de las dos flotas y la última restricción especifica que la asignación de vehículos propios no puede ser mayor al número de vehículos disponibles.

A pesar de que esta es una formulación sencilla, tiene una pesada carga computacional debido a que R es el conjunto de todas las posibles rutas que sirven a los clientes en el conjunto I y se hace imposible enumerar todas las rutas que derivan desde a_{ir} y calcularles su correspondiente costo C_r. Para solucionar esto, los autores plantean la estabilización y actualización del espacio de soluciones con un conjunto parcial de rutas y aplicar el concepto de *Column Generation* (CG) que es también conocido como el método de descomposición de Dantzing-Wolfe para diseñar un procedimiento de adicionar las rutas prometedoras. Además, debido a que la formulación anterior supone un problema tipo NP-Hard, se relaja la primera restricción (de cobertura de clientes) para desarrollar una heurística basada en relajación Lagraniana.

La relajación lagraniana consiste en transformar la formulación del problema relajando la primera restricción del SCP de la siguiente forma:

$$L(\lambda) = Min \sum_{r \in R} c_r(\lambda) x_r + \sum_{i \in I} (p_i - \lambda_i) z_i + \sum_{i \in I} \lambda_i \tag{2.12}$$

$$c_r(\lambda) = c_r - \sum_{i \in i_r} \lambda_i \forall r \in R \tag{2.13}$$

$$\sum_{r \in R} x_r \leq m \tag{2.14}$$

x_r, z_i :***Binarias*** $\forall r \in R$,***i=1...n***

$C_r(\lambda)$: ***Costo Lagraniano de la ruta***

Allí, λ_i corresponde al costo de atender al cliente *i* desde el punto de vista del precio dual y este es llamado como el multiplicador Lagraniano del cliente *i*. Para cada iteración es necesario realizar la actualización de los multiplicadores de Lagrange, por lo que se utiliza el método del subgradiente.

Posteriormente, se encuentra la solución factible, la cual se obtiene inicialmente aplicando una heuristica para la asignación de rutas a la flota propia, que consiste en determinar las rutas a través de la asignación de un puntaje dado por $C_r(\lambda)$, donde el valor más negativo es el mejor.

El concepto de *sweeping* (Barrido) es aplicado a un grupo de clientes para la generación de soluciones iniciales. Esto consiste en ordenar a los clientes en forma creciente en referencia al ángulo radial que forman con respecto al depósito, comenzando desde el primer nodo y agrupando clientes de acuerdo a un parámetro fijo. Después es determinado mediante la solución de un TSP la secuencia óptima de clientes. Para adicionar nuevas rutas a la solución, es implementado un procedimiento denominado PFIH (Push Forward Insertion Heuristic), donde las rutas son ordenadas en base a su puntaje $c_r(\lambda)$, también llamado puntaje Lagraniano. Una ruta es seleccionada aleatoriamente y se efectúan los siguientes pasos: Paso 1: Se calcula la mejor posición de inserción de un cliente en la ruta elegida aleatoriamente, mediante la evaluación del costo Lagraniano asociado con la distancia entre dos nodos menos el multiplicador de Lagrange del nodo de salida, donde el menor valor es el mejor lugar para realizar la inserción. Paso 2: Por comparación en el incremento del costo Lagraniano de todos los nodos no incluidos en la ruta seleccionada. El mejor nodo para hacer la inserción es determinado. Paso 3: Repetir los dos pasos anteriores y genera la ruta con un cliente más, siempre que el límite de capacidad del vehículo no sea superada.

El planteamiento general, consiste en una relajación de la primera restricción de la formulación inicial, para ellos se utiliza una transformación a través de una relajación denominada Lagraniana, en la que incorpora multiplicadores que deben ser actualizados en cada iteración. Estos multiplicadores Lagranianos son conocidos como el costo dual de las rutas. Pero para obtener buenas soluciones, es necesario tener en cuenta la ubicación geográfica de cada cliente, la cual las rutas tienen definido un ángulo entre sí, de acuerdo al sector que atienden, estos ángulos son usados para modificar el cálculo de los multiplicadores de Lagrange y así, por medio de un proceso iterativo, se obtienen las rutas factibles para la flota propia.

La asignación de de rutas a la flota subcontratada, se efectúan los siguientes pasos: Paso 1: se calcula y se ordena el mejoramiento de costos de la inserción de los clientes externos a las rutas

existentes más cercanos de las rutas de la flota propia sin considerar las limitaciones de capacidad. Paso 2: se enfoca en el más prometedor cliente externo y se prueba la inserción en las rutas de la flota propia. Si se excede la capacidad del vehículo, un cliente existente en la ruta es eliminado con el menor costo de eliminación que consiste en evaluar el costo de adicionar el nuevo cliente externo sin asignar y restar el costo del cliente eliminado de la ruta. Este proceso se mantiene hasta que se cumpla con el límite de capacidad del vehículo. Si después de este proceso se obtiene una mejora, se avanza al paso 3, de otra forma se termina con el procedimiento. Paso 3: Si uno o más clientes pasan a ser no atendidos debido al proceso anterior de eliminación, se continua con el paso 1, de otra manera, se va al paso 2 para continuar con la inserción de clientes sin asignación. Al final, los clientes que no puedan ser atendidos por la flota propia serán simplemente asignados a la flota subcontratada.

Vehicle routing problem with time windows considering overtime and outsourcing vehicles [14]

Autores: IlKyeong Moon, Jeong-Hun Lee, June Seong

Fecha de publicación: 2012

Objetivo: Presentar la solución al problema de ruteo de vehículos considerando tiempo extra de los conductores y vehículos subcontratados, mediante la implementación de una modelo de programación entera mixta, un algoritmo genético, un algoritmo híbrido basado en recocido simulado *(Simulated annealing*).

Resumen: En este artículo se presenta un problema con un único depósito, considerando vehículos subcontratados y horas extras de los conductores, este problema es llamado VRPTWOV - *Vehicle routing problem with time windows considering overtime and outsourcing vehicles.* Para este se plantea un modelo de programación entera mixta, el cual será usado como referente para encontrar la solución al problema aplicando técnicas metaheurísticas, ya que encontrar los valores óptimos a través de métodos exactos requiere tiempos computacionales prohibitivos. Por lo tanto, es desarrollado un algoritmo genético híbrido para resolver el VRPTWOV.

El modelo presentado tiene el objetivo de resolver el VRPTWOV y se plantea un modelo de programación entera mixta:

$$min \sum_{i=0}^{n} \sum_{j=0, i\neq j}^{n} \sum_{i=1}^{K} CT_k t_{ij} x_{ijk} + \sum_{k=1}^{K} CR_k(T_{0k} - TO_k) + \sum_{k=1}^{k} CO_k TO_k + \sum_{k=1}^{k} CF_k Y_k \qquad (2.15)$$

Donde:

CT_k: costo del viaje del vehículo *k* por unidad de tiempo

CR_k: Costo de la labor en tiempo regualr para el vehículo *k* por unidad de tiempo

CO_k: Costo de la labor en horas extras o adicionales en el vehículo *k* por unidad de tiempo

CF_k: Costo del vehículo *k* de la flota subcontratada.

t_{ij}: Tiempo de viaje entre los nodos *i-j*

TR_k: Tiempo regular del vehículo *k.*

Variables de decisión:

T_{0k}: Tiempo de arribo del vehículo *k* al depósito.

TO_k: Tiempo extra del vehículo *k.*

x_{ijk}: 1, si el vehículo *k* viaja directamente desde *i* a *j,* 0, de otra forma

Y_k: 1, si el vehículo *k* es usado, 0, de otra forma

Esta función objetivo busca minimizar los costos del total del viaje, los costos totales de la labor ejecutada en tiempo regular y en horas extras y el costo total de los vehículos. Las restricciones se definen como (1) cada vehículo inicie solo una vez en el depósito. (2) asegurar que un vehículo *k* que visite un cliente, debe dejar el mismo cliente. (3) define que cada vehículo debe ser visitado una vez por un vehículo. (4) la restricción de la capacidad del vehículo. (5-8) asegura compatibilidad en los tiempos de llegada y define la ventana de tiempo del vehículo en el cliente. (9) define las horas extras o tiempo adicional de cada vehículo. (10) define que si un vehículo inicia en el depósito Y_k es 1, de otra forma es cero por un costo fijo.

El algoritmo genético (*GA*) inicia con un conjunto de soluciones aleatorias llamadas población y cada individuo de aquella población es llamado cromosoma y representa una solución individual al problema. Estos cromosomas evolucionan a través de iteraciones, cada evolución es denominada como una generación. Para cada generación es evaluada su función objetivo. Esta descendencia es formada a través del cruce de dos cromosomas por medio de la utilización de un operador de cruzamiento y/o modificando un cromosoma usando un operador de mutación. Las nuevas generaciones son seleccionadas de acuerdo a su función objetivo. Los cromosomas con peor función objetivo son eliminados de la población que debe tener una cantidad de miembros constante. El algoritmo híbrido implementado, consiste en la utilización de la metaheurística recocido simulado *(Simulated annealing - SA)* junto con el *GA*. El *SA* radica en simular el calentamiento o fundición de un metal para posteriormente enfriarlo lentamente y obtener una estructura cristalizada, es decir, se busca por medio del enfriamiento del material que su estructura inicialmente que se encontraba en desorden pase a un estado de orden.

La representación o codificación del problema se construye mediante dos vectores o dos tipos de cromosomas. El primero representa la secuencia de visita de los clientes y el segundo representa la asignación de vehículos. El VRPTWOV tiene una alta dependencia de la solución inicial, por lo tanto, se genera una solución inicial factible a través de una heurística golosa.

El *SA* es una poderosa técnica para resolver complicados problemas combinatoriales ya que su principal característica es que puede escapar de óptimos locales, pero tiene una velocidad de convergencia lenta que depende de la velocidad del enfriamiento que busca encontrar el mínimo de energía del material. Por lo tanto, se desarrolla un *SA* híbrido que evita velocidades lentas y prematuras convergencias del algoritmo genético, es decir, en este algoritmo híbrido, actúan de manera colaborativa el *GA* y el *SA*. Finalmente, los detalles de los componentes del algoritmo híbrido son: Generar soluciones iniciales: Se implementa una técnica golosa. Generación de vecindarios (Perturbaciones): Se Implementa el GA ya que con esta técnica se obtienen rápidamente utilizando los operadores de mutación y cruzamiento. Evaluación y actualización (Tiempo de refrigeración): Se utiliza el SA para encontrar mejores soluciones a través de la comparación de la mejor solución con la solución del vecindario. Parámetros y condiciones de terminación: Allí se definen los parámetros de temperatura del SA y las tasas de cruzamiento y mutación del GA, además del máximo de iteraciones.

New Evolutionary Algorithm Based on 2-Opt Local Search to Solve the Vehicle Routing Problem with Private Fleet and Common Carrier [9]

Autores: Jalel Euchi, Habib Chabchoub, Adnan Yassine

Fecha de Publicación: 2013

Objetivo: Examinar el VRPPC para desarrollar una técnica de solución descrita como algoritmo de estimación evolutiva de densidad iterativa (*Iterated Density Estimation Evolutionary Algorithm - IDEA*) con 2-opt actuando como método de búsqueda local, para determinar la asignación específica de cada ruta de la flota propia y la subcontratada.

Resumen: El planteamiento del problema se basa en la clásica formulación del VRPPC en el cual se tiene una formulación del problema igual a la presentada en [3]. La metaheurística IDEA se muestra como un tipo de algoritmo evolucionario en el cual una población inicial de individuos codifica una posible solución al problema, que es mejorada por la aplicación de operadores estocásticos. IDEA itera los tres pasos siguientes hasta que se cumpla algún criterio de terminación:

1. Selecciona buenos candidatos (soluciones) desde una población de soluciones inicial generada aleatoriamente.

2. Estima la distribución de probabilidad de los individuos seleccionados.

3. Genera nuevos candidatos o descendencia desde la distribución estimada.

Modelo: Se presenta un modelo basado en un algoritmo evolucionario el cual inicia con la construcción de una solución inicial o población, por medio de una heurística de inserción la cual es mejorada aplicando una búsqueda local con el 2-opt, seguido se realiza la selección de los mejores individuos por medio de un operador de selección. El paso siguiente será estimar la distribución de probabilidad de los individuos seleccionados aplicando el modelo de probabilidad generando nuevas soluciones que serán insertadas en la población por medio de un remplazo por torneo. Finalmente se aplica nuevamente el 2-opt y el todo el proceso se repite hasta cumplir con los criterios de terminación.

La formulación de la función objetivo se da de la forma:

$$min \sum_{k=1}^{m} F_k y_{0k} + \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{m} c_{ijk} x_{ijk} + \sum_{i=1}^{n} L_i Z_i \tag{2.16}$$

que incluyendo las restricciones, es idéntica a la propuesta en [4], pero con modificaciones en la notación.

La metodología de solución consiste en implementar un IDEA hibrido (IDEA / 2-opt) para mejorar las soluciones generadas después de la creación de la población inicial y la generación de nuevas soluciones. Este algoritmo híbrido se implementa mediante las siguientes partes:

Representación de las rutas: una adecuada representación consiste en un cromosoma que contiene varias rutas, cada una contiene un subconjunto de clientes que deben ser visitados en el mismo orden en el que ellos aparecen. Este vector tendrá una dimensión de $n + k$, donde n es el número de clientes y k es el número vehículos. Cada ruta será separada dentro del vector en el índice donde aparezca un cero. Las últimas rutas corresponderán a la asignación a los vehículos subcontratados.

2-opt como búsqueda local: para el mejoramiento de las rutas, se utiliza el 2-opt para búsquedas locales, que consiste en un movimiento intra-ruta. Este operador utiliza un intercambio (*Swapping)* de los arcos de dos clientes de la misma ruta.

Inicialización: Para el inicio, se desarrolla una heurística de inserción propuesta por Euchi and Chabchoub (2009), que posteriormente será mejorada por la aplicación de un 2-opt.

Operadores de Selección: Los operadores de selección en algoritmos evolucionarios permiten seleccionar las mejores soluciones de la población. Aquí es usado el procedimiento llamado selección por presión hacia diversidad de Bosman and Thierens (2002).

Modelo Probabilístico: Una de las partes exitosas de IDEA es el uso de un modelo de probabilidad, que captura la importancia de la correlación de la distribución de la búsqueda, asignando altos valores de probabilidad a la solución seleccionada. IDEA construye un modelo probabilístico con los mejores individuos y luego toma muestras del mismo modelo para generar otros nuevos individuos.

Remplazo: Cómo todo algoritmo evolutivo, para mantener una población constante, es necesario remplazar individuos por los nuevos creados a través del modelo probabilístico. Esto puede realizarse aplicando un remplazo por torneo.

An Adaptable Variable Neighborhood Search For The Vehicle Routing Problem With Order Outsourcing [18]

Autores: Sybren Huijink, Goos Kant, René Peeters

Fecha de publicación: 2014

Objetivo: Presentar dos heurísticas de búsquedas adaptables en vecindarios variables con una alta competitividad y nuevas pruebas de instancias, basadas en eficiencia en lugar de necesidad, para encontrar la solución al problema de ruteo con vehículos propios y subcontratados.

Resumen: Se plantea una metodología basada en una búsqueda adaptable en un vecindario variable (AVNS), donde este puede ser de comportamiento rápido o lento. En ambas aplicaciones, se utilizan operadores de sacudida o perturbación como: *Cycling Move, Jumps, Shifting, create, Destroy, Split* y *Bomb*. La heurística de búsqueda tabú es usada como la técnica de búsqueda local con una variación de resiembra o reasignación para el mejoramiento de todas las soluciones encontradas después de la aplicación de los operadores. Este procedimiento es usado para la construcción de las dos heurísticas: AVNS - Fast y AVNS - Slow. Los resultados obtenidos con estas implementaciones son significativamente buenos para casos donde la flota es heterogénea para la aplicación AVNS-Slow (Lento).

Modelo: La formulación base es definida con el siguiente conjunto de ecuaciones:

$$Min \sum_{k\in\Omega} C(r)x_r + \sum_{i=1}^{n} p_i \left(1 - \sum_{k\in\Omega} a_i(r)\, x_r\right) \tag{2.17}$$

Sujeto a:

$$\sum_{k\in\Omega} a_i(r)\, x_r \leq 1 \;\; \forall\, i \in [n] \tag{2.18}$$

$$\sum_{k \in \Omega} x_r \leq m \tag{2.19}$$

$$x_r \in \{0,\, 1\} \;\; \forall\, r \in \Omega \tag{2.20}$$

Donde:

Conjunto de ordenes: $\{0, 1, ..., n\}$, cero corresponde al depósito.

$c(r)$: Es el costo total de la ruta *r* donde todos los vehículos *m* son homogéneos con capacidad *Q*.

$a(r)$: es igual a uno si la ruta $r \in \Omega$ y las ordenes de envío $i \in [n] = \{1, ..., n\}$,y cero de otra forma.

x_r: Variable de decisión que implica que la ruta *r* se esta conduciendo o utilizando.

p_i: Precio de visitar el cliente *i* con un vehículo de la flota subcontratada.

Ω : Corresponde al conjunto de todas las rutas factibles.

El AVNS cuenta con movimientos denominados de sacudida, lo que implica la utilización de distintos tipos de operadores para la transferencia de clientes entre vehículos y la creación de nuevas rutas. Después de aplicado cualquier movimiento de sacudida siempre es utilizada una búsqueda tabú como búsqueda local. Debido a la utilización de estos movimientos de sacudida, algunos vehículos tendrán capacidad disponible que debe ser llenada con clientes de vehículos cercanos para que estos sigan siendo rentables. Para lo anterior, es utilizado un procedimiento llamado *Shifting Method* que consiste en separar los vehículos en vehículos recibidores y vehículos proveedores, los cuales transfieren unos pocos clientes desde los proveedores a los recibidores. Este procedimiento se realiza a través de varias denominadas olas de transferencia de clientes que garantizan que todos los vehículos al final de este procedimiento sean lo más rentables posibles.

Para realizar los movimientos de sacudida o perturbación se requieren de diferentes operadores como *Cyclic Move*. Este operador inicia con la selección de vehículos de acuerdo a un método empleado para este fin. A estos vehículos se les remueve cierta cantidad de clientes de acuerdo a otro criterio de selección. Estos clientes removidos son insertados en vehículos que apenas ingresan al ciclo y que tienen el menor valor de acuerdo a un criterio de inserción por distancias. El ciclo termina cuando se llega a la selección de nuevo al primer vehículo. El segundo operador llamado *Create,* que consiste en verificar si existen vehículos no usados, los cuales pueden ser beneficiados para ser llenados con clientes u órdenes de servicio rentables. La asignación de clientes se da consiguiendo un cliente semilla obtenido del mejor puntaje asignado de acuerdo a su precio mínimo o distancia. Posteriormente se aplica un *Shifting Method* donde este nuevo vehículo es catalogado como recibidor y los demás como proveedores. El tercer operador de perturbación es el *Destroy.*

Este operador consiste en elegir un vehículo de manera aleatoria y remover todos sus clientes para posteriormente crearle una nueva ruta con el operador *Create,* lo que permitiría explorar si algún vehículo es más rentable en otra área. El cuarto operador de llamado *Split,* el cual considera que es posible mejorar la rentabilidad creando dos rutas a partir de una existente, donde se elige un vehículo aleatoriamente para realizar la división que es efectuada entre los clientes que tienen mayor distancia entre si. La primera parte de los clientes permanece en el vehículo inicial y la restante es asignada a un vehículo que no esté en uso. posteriormente se aplica un *Shifting Method* para llenar ambos vehículos de manera óptima. El quinto operador es denominado *Bomb,* el cual aplica un movimiento de destrucción a un área específica alrededor de un cliente, para posteriormente reconstruirla, para lo que se deben definir los límites de afectación de clientes. El operador seis corresponde al *Jump* el cual remueve clientes o ordenes de un vehículo e inserta clientes que aún no han sido asignados. Posterior al movimiento es necesario realizar otros movimientos aleatorios de clientes para eliminar infactibilidades. El séptimo operador es el *Reseeding* que hace parte del conjunto de movimientos de destrucción de rutas y se encarga de remover partes de varias rutas y reconstruirlas para obtener nuevas soluciones. Ha estas nuevas soluciones se les aplica una *Shifting Method* para poder la carga de los vehículos. Finalmente, el último movimiento es la Búsqueda tabú que es utilizada como método búsqueda local.

Modelo matemático para resolver el problema de localización y ruteo con restricciones de capacidad considerando flota propia y subcontratada [20]

Autores: Toro-Ocampo Eliana Mirledy, Franco-Baquero John Fredy, Gallego-Rendón Ramón Alfonso

Fecha de publicación: 2016

Objetivo: Presentar un nuevo modelo matemático para el problema de localización y ruteo con flota propia y subcontratada (CLRPPC), en el que las restricciones clásicas para evitar sub-tours se reemplazan por un conjunto de restricciones que establecen conexiones radiales entre los clientes y depósitos.

Resumen: Para la construcción del modelo, se deben de tener en cuenta las siguientes consideraciones: Se cuenta con un conjunto I de centros de distribución candidatos a abrirse, un conjunto de clientes J con una demanda conocida que se debe atender; una flota propia de vehículos homogéneos de tamaño k, insuficiente para atender la demanda total de los clientes. Para obtener los grafos requeridos, es necesario, además, tener las siguientes consideraciones:

1. Un vehículo de la flota propia sirve una sola ruta que comienza y termina en el CD.

2. Cada cliente es visitado una única vez por cualquier tipo de vehículo.
3. La capacidad *Q* de cada vehículo no puede ser excedida.
4. Los vehículos de la flota contratada solo realizan una ruta *r.*
5. Los despachos desde el depósito *i* no pueden exceder su capacidad W_i
6. La suma de todos los costos incluye: Costo fijo de apertura de los centros de distribución, costos fijos de los vehículos propios y costos asociados a los recorridos realizados bien sea por la flota propia o la subcontratada.

Modelo: Se presenta un modelo matemático para el problema de localización y ruteo con flota propia y subcontratada (CLRPPC), que permite que los vehículos de la flota propia realicen trayectorias cerradas (siempre llegando al depósito donde iniciaron) y trayectorias radiales para los vehículos de la flota subcontratada. La implementación de este modelo se hace a través de instancias reconocidas en la literatura de este problema y para resolverlo se utiliza un solver comercial CPLEX 12.5 donde se implementa la formulación del problema lineal entero mixto. Este modelo además de resolver el CLRPPC, con algunas variaciones o adaptaciones tiene la capacidad de resolver los problemas VRPPC (Problema de ruteo con flota propia y subcontratada) y el MDVRPPC (Problema de ruteo con flota propia y subcontratada para múltiples depósitos).

Se definen las siguientes variables como:

c_{ij}: Costo asociado al recorrido entre los nodos *i-j.*

x_{ij}: Corresponde a los arcos recorridos por la ruta propia.

a_{ij}: Identifican los arcos de retorno del depósito de inicio de la ruta.

s_{ij}: Variable binaria que indica cuando se utiliza un vehículo de la flota subcontratada.

P: Penalización por la utilización de la flota subcontratada.

o_i: Costo fijo de apertura del depósito *i.*

y_i: Variable binaria que indica que se abrió el depósito *i*

F: Costo fijo del vehículo.

t_{ij}: Flujo de mercancías entre los nodos *i-j* para la flota propia.

l_{ij}: Flujo de mercancías entre los nodos *i-j* para la flota subcontratada.

f_{ij}: Variable binaria que indica que el nodo *j* está conectado con el centro de distribución *i.*

El modelo matemático se presenta como:

$$min = \sum_{i \in I} o_i y_i + \sum_{\substack{i \in I \\ j \in J}} F a_{ij} + \sum_{i,j \in V} C_{ij} x_{ij} + \sum_{\substack{i \in I \\ j \in J}} c_{ij} a_{ij} + P \sum_{\substack{i \in I \\ j \in J}} c_{ij} s_{ij} \tag{2.21}$$

Sujeto a:

$$\sum_{i \in V} x_{ij} + \sum_{i \in V} s_{ij} = 1 \tag{2.22}$$

$$\sum_{k \in J} x_{ij} + \sum_{i \in I} a_{ij} = \sum_{i \in V} x_{ij} \tag{2.23}$$

$$\sum_{j \in J} x_{ij} = \sum_{j \in J} a_{ij} \tag{2.24}$$

$$\sum_{k \in J} s_{ik} \leq \sum_{i \in V} s_{ij} \tag{2.25}$$

$$x_{ij} + x_{ji} + s_{ij} + s_{ji} \leq 1 \tag{2.26}$$

$$\sum_{\substack{i \in V \\ i \neq J}} t_{ij} + l_{ij} = \sum_{\substack{k \in V \\ k \neq j}} (t_{jk} + l_{jk}) + D_j \tag{2.27}$$

$$\sum_{\substack{i \in I \\ j \in J}} x_{ij} + s_{ij} = card(J) \tag{2.28}$$

$$\sum_{i \in I} f_{ij} \leq 1 \tag{2.29}$$

$$t_{ij} \leq Q x_{ij} \tag{2.30}$$

$$l_{ij} \leq Q s_{ij} \tag{2.31}$$

$$\sum_{j \in J} t_{ij} + l_{ij} \leq w_i y_i \tag{2.32}$$

$$\sum_{i \in V} s_{ij} + \sum_{k \in V} x_{kj} = 1 - z_j \tag{2.33}$$

$$1 + a_{ij} \geq f_{ij} + z_j \tag{2.34}$$

$$-(1 - x_{ju} - x_{uj}) \leq f_{ij} - f_{iu} \tag{2.35}$$

$$f_{ij} - f_{iu} \leq (1 - x_{ju} - x_{uj}) \tag{2.36}$$

$$f_{ij} \geq x_{ij} \tag{2.37}$$

$$\sum_{i \in I} y_i \geq \sum_{j \in J} D_j / \sum_{i \in I} w_i \tag{2.38}$$

$$\sum_{\substack{i \in I \\ j \in J}} x_{ij} + s_{ij} \leq \sum_{j \in J} D_j / Q \tag{2.39}$$

$$\sum_{\substack{i \in I \\ j \in J}} a_{ij} \leq N V_a \tag{2.40}$$

Se define que las variables x_{ij}, s_{ij}, y_i, f_{ij}, z_j y a_{ij} Son binarias.

Esta formulación define al problema como lineal entero mixto. El conjunto de restricciones 2.22-2.29 aseguran las trayectorias radiales del CLRPPC de todos los vehículos, que inician desde un depósito y conectan a todos los clientes, de esta manera no se presentan sub-rutas o sub-tours. Las restricciones desde las ecuaciones 2.33 a la 2.37 se aplican únicamente para la flota propia e identifican los nodos terminales de ruta para conectarlos con el depósito y asegurar que los vehículos de la flota propia terminen allí la ruta, es decir, trayectorias cerradas. Las restricciones 2.30 a la 2.32 y 2.38 a la 2.40, establecen todos los límites operativos.

2.3. Comparativo modelos matemáticos para el VRPPC

Para esta sección, se construye el cuadro 2.1, donde se describen los modelos matemáticos propuestos en las publicaciones revisadas anteriormente. Algunos trabajos utilizan la formulación de otros, por lo tanto, solo se incluirán aquellos modelamientos propuestos por los autores originales.

Año	Ref.	Modelo	Descripción de las Restricciones	Detalle de las variables
2005	[4]	$min\ z = \sum_{k}^{m} FC_k + \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n} C_{ijk}X_{ijk} + \sum_{i}^{n} CL_iL_i$ (1) Sujeto a: $\sum_{k}^{m} Y_{0k} = m\ (k = 1, ..., m)$ (2) $\sum_{k}^{m} Y_{ik} + L_i = 1\ (i = 1, ..., m)$ (3) $\sum_{i}^{n} q_iY_{ik} \leq Q_k\ (i = 1, ..., n;\ k = 1, ..., m)$ (4) $\sum_{j}^{n} X_{ijk} = Y_{ik}\ (i = 1, ..., n;\ k = 1, ..., m)$ (5) $\sum_{j}^{n} X_{jik} = Y_{ik}\ (i = 1, ..., n;\ k = 1, ..., m)$ (6) $\sum_{j}^{n} X_{ijk} \leq \lvert S\rvert - 1\ \forall\ S \subseteq \{2, ..., n\}\ (k = 1, ..., m)$ (7)	(2) asegura que a todos los vehículos se les haya asignado clientes. (3) Asegura que cada clientes sea atendido por un vehículo de cualquiera de las flotas. (4) Restringe la capacidad de carga de los vehículos. (5) y (6) aseguran que un vehículo que llega a un cliente también salga de allí. (7) no permite la creación se subtours.	FC_k :Corresponde a los costos fijos del vehículo k C_{ijk} :Son los costos del vehículo k viajando del cliente i al j CL_i :Costo asociado a la utilización de un vehículo de la flota subcontratada que atiende al cliente i $X_{ijk} \in \{0, 1\};\ Y_{ik} \in \{0, 1\};$ $L_i \in \{0, 1\}$ $(i = 1, ..., n;\ j = 0; ...n;\ ; k = 1, ..., m);$ $i : \{i = 0, ..., n\}$, *Indica el conjunto de clientes, donde cero denota el depósito.* $j : \{i = 0, ..., n\}$, *Indica el conjunto de clientes.* $k : \{i = 1, ..., m\}$, *Indica el conjunto de vehículos.* n : *El número de clientes.* m : *El número de vehículos.*
2008	[3]	$minz = \sum_{k}^{m} f_k y_{0k} + \sum_{i}^{n}\sum_{j}^{n}\sum_{k}^{n} C_{ijk}X_{ijk} + \sum_{i}^{n} e_iz_i$ (1) Sujeto a: $\sum_{j=1}^{n}\sum_{k=1}^{m} x_{0jk} = \sum_{i=1}^{n}\sum_{k=1}^{m} x_{i0k} \leq m$ (2) $\sum_{\substack{j=1 \\ j\neq h}}^{n} x_{hjk} = \sum_{\substack{i=1 \\ i\neq h}}^{n} x_{ijk} = y_{hk};$ $h \in \{0, ..., n\};\ k \in \{1, ..., m\})$ (3) $z_i + \sum_{k=1}^{m} y_{ik} = 1; i \in \{1, ..., n\}$ (4) $\sum_{i=1}^{n} q_iy_{ik} = 1; k \in \{1, ..., m\}$ (5) $U_{ik} - U_{jk} + Q_kx_{ijk} \leq Q_k - q_j$ $i,\ j \in \{1, ..., n\},\ i \neq j;\ k \in \{1, ..., m\}$ (6)	Con las restricciones (2) al menos uno de los vehículos de la flota propia debe ser usado. (3) el mismo vehículo debe entrar y dejar el mismo cliente. (4) asignar cada vehículo a la flota propia o subcontratada. (5) la capacidad del vehículo nunca sera excedida. (6) se eliminan las subrutas.	$x_{ijk} =$ {1 Si el vehiculo *k* visitó *j* después de *i*; 0 De lo contrario} $y_{ik} =$ {1 Si el vehiculo *k* visitó *i*; 0 De lo contrario} $z_i =$ {1 Si el clilente *i* es asignado a la flota propia; 0 De lo contrario} u_{ik}: Limite superior de la carga del vehículo *k* al dejar el cliente *i*

Cuadro 2.1 continua en siguiente página...

Año	Ref.	Modelo	Descripción de las Restricciones	Detalle de las variables
2011	[12]	$L(\lambda) = Min \sum_{r \in R} c_r(\lambda) x_r + \sum_{i \in I} (p_i - \lambda_i) z_i + \sum_{i \in I} \lambda_i$ (1) $c_r(\lambda) = c_r - \sum_{i \in i_r} \lambda_i \; \forall r \in R$ (2) $\sum_{r \in R} x_r \leq m$ (3)	La única restricción (3) indica que las rutas propias no debe exceder la cantidad de vehículos propios.	x_r, z_i :*Binarias* $\forall r \in R$,*i=1...n* λ_i: corresponde al multiplicador Lagraniano del cliente *i.* $C_r(\lambda)$: *Costo Lagraniano de la ruta* p_i: Costo lineal z_i: variable de decisión para el uso de vehículos subcontratados para el cliente *i* m: Cantidad de vehículos propios

Cuadro 2.1 continua en siguiente página...

Año	Ref.	Modelo	Descripción de las Restricciones	Detalle de las variables
2012	[14]	$min \sum_{i=0}^{n}\sum_{j=0,i\neq j}^{n}\sum_{i=1}^{K} CT_k t_{ij} x_{ijk} + \sum_{k=1}^{K} CR_k(T_{0k} - TO_k)$ $+ \sum_{k=1}^{k} CO_k TO_k + \sum_{k=1}^{k} CF_k Y_k$ (1) Sujeto a: $\sum_{j=1}^{N} X_{0jk} \leq 1, \forall\ k \in \{1, ..., K\}$ (2) $\sum_{j=0}^{N} X_{ijk} = \sum_{j=1}^{N} X_{jik}, \forall\ i \in \{1, ..., N\}\ \ k \in \{1, ..., K\},\ and\ i \neq j$ (3) $\sum_{k=1}^{N}\sum_{i=0,i\neq j}^{N} X_{ijk} = 1, \forall\ i \in \{1, ..., N\}$ (4) $\sum_{i=1}^{N} m_i \sum_{j=0,i\neq j}^{N} x_{ijk} \leq q_k, \forall\ k \in \{1, ..., K\}$ (5) $r_k \geq T_{0k} \geq T_i + s_i + w_i + t_{ij} + M\left(x_{i0k} - 1\right),$ $\forall\ i \in \{1, ..., N\}\ \ k \in \{1, ..., K\}$ (6) $e_i \leq T_i + w_i \leq l_i\ \forall\ i \in \{1, ..., N\}$ (7) $T_0 = s_0 = w_0 = 0$ (8) $TO_k - NO_k = T_{0k} - TR_k\ \forall\ k \in \{1, ..., K\}$ (9) $Y_k = \sum_{j=1}^{N} x_{0jk}\ \forall\ k \in \{1, ..., K\}$ (10) $\forall\ x_{ijk}, \forall\ Y_k \in \{0,\ 1\}$ (11)	(2) cada vehículo inicie solo una vez en el depósito. (3) asegurar que un vehículo *k* que visite un cliente, debe dejar el mismo cliente. (4) define que cTada vehículo debe ser visitado una vez por un vehículo. (5) la restricción de la capacidad del vehículo. (6-9) asegura compatibilidad en los tiempos de llegada y define la ventana de tiempo del vehículo en el cliente. (10) define las horas extras o tiempo adicional de cada vehículo. (11) define que si un vehículo inicia en el depósito Y_k es 1, de otra forma es cero por un costo fijo.	CT_k: costo del viaje del vehículo *k* por unidad de tiempo CR_k: Costo de la labor en tiempo regualr para el vehículo *k* por unidad de tiempo CO_k: Costo de la labor en horas extras o adicionales en el vehículo *k* por unidad de tiempo CF_k: Costo del vehículo *k* de la flota subcontratada. t_{ij}: Tiempo de viaje entre los nodos *i-j* TR_k: Tiempo regular del vehículo *k*. s_i: tiempo de servicio en el cliente *i* e_i: tiempo prematuro para el cliente *i* l_i: tiempo tardío para el cliente *i* r_k : máximo tiempo permitido para la ruta del vehículo *k* m_i : demanda del cliente *i* q_k: capacidad del vehículo *k* K: número total de vehículos N: número total de clientes M: gran número Variables de decisión: T_{0k}: Tiempo de arribo del vehículo *k* al depósito. TO_k: Tiempo extra del vehículo *k*. T_i: tiempo de llegada del cliente *i* w_i: tiempo de espera para el cliente *i* NO_k: $max\ \{TR_k - T_{0k},\ 0\}$ del vehículo *k* x_{ijk}: 1, si el vehículo *k* viaja directamente desde *i* a *j,* 0, de otra forma Y_k: 1, si el vehículo *k* es usado, 0, de otra forma

Cuadro 2.1 continua en siguiente página...

Año	Ref.	Modelo	Descripción de las Restricciones	Detalle de las variables
2014	[18]	$Min \sum_{k\in\Omega} C(r)x_r + \sum_{i=1}^{n} p_i \left(1 - \sum_{k\in\Omega} a_i(r)\, x_r\right)$ (1) Sujeto a: $\sum_{k\in\Omega} a_i(r)\, x_r \leq 1 \;\forall\, i \in [n]$ (2) $\sum_{k\in\Omega} x_r \leq m$ (3) $x_r \in \{0,\, 1\} \;\forall\, r \in \Omega$ (4)	(2) Asegura que cada cliente sea atendido no más de una vez. (3) El número de rutas no puede ser mayor al número de vehículos disponibles	Conjunto de ordenes: $\{0, 1, ..., n\}$, cero corresponde al depósito. $c(r)$: Es el costo total de la ruta *r* donde todos los vehículos *m* son homogéneos con capacidad *Q*. $a(r)$: es igual a uno si la ruta $r \in \Omega$ y las ordenes de envío $i \in [n] = \{1, ..., n\}$,y cero de otra forma. x_r: Variable de decisión que implica que la ruta *r* se esta conduciendo o utilizando. p_i: Precio de visitar el cliente *i* con un vehículo de la flota subcontratada. Ω : Corresponde al conjunto de todas las rutas factibles.

Cuadro 2.1 continua en siguiente página...

Año	Ref.	Modelo	Descripción de las Restricciones	Detalle de las variables
2016	[20]	$min = \sum_{i\in I} o_i y_i + \sum_{\substack{i\in I \\ j\in J}} F a_{ij} + \sum_{i,j\in V} C_{ij} x_{ij} + \sum_{\substack{i\in I \\ j\in J}} c_{ij} a_{ij} + P \sum_{\substack{i\in I \\ j\in J}} c_{ij} s_{ij}$ (1) sujeto a: $\sum_{i\in V} x_{ij} + \sum_{i\in V} s_{ij} = 1$ (2) $\sum_{k\in J} x_{ij} + \sum_{i\in I} a_{ij} = \sum_{i\in V} x_{ij}$ (3) $\sum_{j\in J} x_{ij} = \sum_{j\in J} a_{ij}$ (4) $\sum_{k\in J} s_{ik} \leq \sum_{i\in V} s_{ij}$ (5) $x_{ij} + x_{ji} + s_{ij} + s_{ji} \leq 1$ (6) $\sum_{\substack{i\in V \\ i\neq J}} t_{ij} + l_{ij} = \sum_{\substack{k\in V \\ k\neq j}} (t_{jk} + l_{jk}) + D_j$ (7) $\sum_{\substack{i\in I \\ j\in J}} x_{ij} + s_{ij} = card(J)$ (8) $\sum_{i\in I} f_{ij} \leq 1$ (9) $t_{ij} \leq Q x_{ij}$ (10) $l_{ij} \leq Q s_{ij}$ (11) $\sum_{j\in J} t_{ij} + l_{ij} \leq w_i y_i$ (12) $\sum_{i\in V} s_{ij} + \sum_{k\in V} x_{kj} = 1 - z_j$ (13) $1 + a_{ij} \geq f_{ij} + z_j$ (14) $-(1 - x_{ju} - x_{uj}) \leq f_{ij} - f_{iu}$ (15) $f_{ij} - f_{iu} \leq (1 - x_{ju} - x_{uj})$ (16) $f_{ij} \geq x_{ij}$ (17) $\sum_{i\in I} y_i \geq \sum_{j\in J} D_j / \sum_{i\in I} w_i$ (18) $\sum_{\substack{i\in I \\ j\in J}} x_{ij} + s_{ij} \leq \sum_{j\in J} D_j / Q$ (19) $\sum_{\substack{i\in I \\ j\in J}} a_{ij} \leq N V_a$ (20)	El conjunto de restricciones (2)-(9) aseguran las trayectorias radiales del CLRPPC de todos los vehículos, que inician desde un depósito y conectan a todos los clientes, de esta manera no se presentan sub-rutas o sub-tours. Las restricciones del número (13) a la (17) se aplican únicamente para la flota propia e identifican los nodos terminales de ruta para conectarlos con el depósito y asegurar que los vehículos de la flota propia terminen allí la ruta, es decir, trayectorias cerradas. Las restricciones (10) a la (12) y (18) a la (20), establecen todos los límites operativos.	c_{ij}: Costo asociado al recorrido entre los nodos *i-j*. x_{ij}: Corresponde a los arcos recorridos por la ruta propia. a_{ij}: Identifican los arcos de retorno del depósito de inicio de la ruta. s_{ij}: Variable binaria que indica cuando se utiliza un vehículo de la flota subcontratada. P: Penalización por la utilización de la flota subcontratada. o_i: Costo fijo de apertura del depósito *i*. y_i: Variable binaria que indica que se abrió el depósito *i* F: Costo fijo del vehículo. t_{ij}: Flujo de mercancías entre los nodos *i-j* para la flota propia. l_{ij}: Flujo de mercancías entre los nodos *i-j* para la flota subcontratada. f_{ij}: Variable binaria que indica que el nodo *j* está conectado con el centro de distribución *i*. Se define que las variables x_{ij},s_{ij}, y_i, f_{ij}, z_j ya_{ij} Son binarias.

Cuadro 2.1: Comparativo modelos matemáticos

Capítulo 3

Modelo matemático para el VRPPC

3.1. Consideraciones iniciales

Para este problema se tienen las siguientes consideraciones las cuales hacen una delimitación del problema:

1. Se cuenta con un único depósito, donde los vehículos propios retornaran posterior a visitar el último cliente, los vehículos subcontratados no deberán volver al depósito.
2. La capacidad del depósito es infinita o suficiente para atender la demanda del problema.
3. Todos los vehículos tienen una capacidad de carga limitada y uniforme, incluyendo los vehículos subcontratados. Esto hace que el problema se denomine homogéneo.
4. El conjunto de clientes *"i"* tienen una demanda conocida y la suma de todas ellas es mayor a la capacidad junta de todos los vehículos propios *"k"*.
5. La red asociada al problema consta de dos grafos completos, el primero asociado a los arcos recorridos por las rutas propias y el segundo a las subcontratadas.[20]
6. Cada cliente será visitado una única vez con cualquiera de los vehículos, ya sea propio o subcontratado.

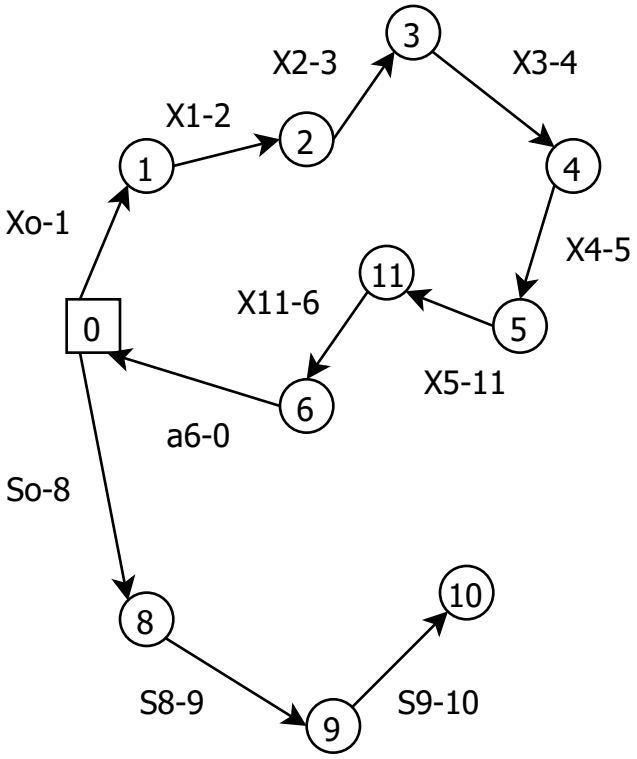

Figura 3.1: Gráfico ejemplo solución del VRPPC

7. Cada vehículo de las rutas subcontratadas podrá realizar solo un recorrido.

En la figura 3.1, se muestra un ejemplo de la solución del VRPPC para un vehículo propio y uno subcontratado. La variable x_{ij} indica cuando un arco es válido, a_{i0} indica que existe un arco de cierre de ruta entre los nodos i y el depósito en las rutas propias. La variable s_{ij} indica cuando es válido un arco de la ruta subcontratada.

3.2. Construcción del modelo matemático

El VRPPC se formula como un problema lineal entero mixto y se define en los siguientes conjuntos y ecuaciones [20]:

Se tiene que el problema puede representarse como un grafo completo de la forma $G = (V, A)$, donde V representa un conjunto con la totalidad de los vértices del grafo (depósitos y clientes), y A el conjunto de arcos que conectan los clientes y el depósito. Se tiene el conjunto $J = \{1, 2, 3, ...n\}$, contiene los nodos de clientes (el depósito tendrá siempre asignado el nodo cero). Por lo tanto, su definición se da cómo:

$$min \sum_{i,\, j \in V} c_{ij} x_{ij} + \sum_{j \in V} c_{j-0}\, a_{j-0} + P \sum_{i,\, j \in V} c_{ij} s_{ij} \tag{3.1}$$

Sujeto a:

$$\sum_{\substack{i \in V \\ j \in J}} x_{ij} + \sum_{\substack{i \in V \\ j \in J}} s_{ij} = 1 \tag{3.2}$$

$$\sum_{\substack{i \in V \\ j \in J}} x_{jk} + \sum_{j \in J} a_{j-0} = \sum_{\substack{i \in V \\ j \in J}} x_{ij} \tag{3.3}$$

$$\sum_{j \in J} x_{0-j} = \sum_{j \in J} a_{j-0} \tag{3.4}$$

$$\sum_{\substack{k \in J \\ j \in J}} s_{jk} \leq \sum_{\substack{i \in V \\ j \in J}} s_{ij} \tag{3.5}$$

$$x_{ij} + x_{ji} + s_{ij} + s_{ji} \leq 1 \quad \forall\, i \in V \;\; j \in J \tag{3.6}$$

$$\sum_{\substack{i \in V \\ i \neq j}} (t_{ij} + l_{ij}) = \sum_{\substack{k \in V \\ k \neq j}} (t_{jk} + l_{jk}) + D_j \quad \forall\, j \in J \tag{3.7}$$

$$\sum_{j \in J} (x_{ij} + s_{ij}) = card(J) \tag{3.8}$$

$$t_{ij} \leq Q x_{ij} \quad \forall\, j \in J \tag{3.9}$$

$$l_{ij} \leq Q s_{ij} \quad \forall\, j \in J \tag{3.10}$$

$$\sum_{\substack{i \in V \\ j \in J}} s_{ij} + \sum_{\substack{k \in V \\ j \in J}} x_{kj} = 1 - z_j \tag{3.11}$$

$$a_{j-0} \geq z_j \quad \forall\, j \in J \tag{3.12}$$

$$\sum_{j \in J}(x_{0-j} + s_{0-j}) \leq \sum_{j \in J} D_j/Q \tag{3.13}$$

$$\sum_{j \in J} a_{0-j} \leq NV_a \tag{3.14}$$

Las variables x_{ij}, s_{ij}, z_j, a_{j-0} $\forall i,\ j \in V$ son binarias y $l_{ij} \in \Re$ $\forall\ i,\ j \in V$

Se define cada variable así:

c_{ij}: Costo asociado al recorrido entre los nodos *i-j*.

x_{ij}: Corresponde a la activación de los arcos recorridos por la ruta propia.

a_{ij}: Corresponde a la activación de los arcos de retorno al depósito.

s_{ij}: Variable binaria que indica cuando se utiliza un vehículo de la flota subcontratada.

P: Penalización por la utilización de la flota subcontratada.

t_{ij}: Flujo de mercancías entre los nodos *i-j* para la flota propia.

l_{ij}: Flujo de mercancías entre los nodos *i-j* para la flota subcontratada.

D_j: Demanda del cliente j

Q: Capacidad total de carga de todos los vehículos

NV_a: Número de vehículos disponibles de la flota propia.

La ecuación 3.1, describe la función objetivo como la suma de los costos c_{ij} que se activan con la variable de decisión x_{ij}; Los costos c_{j-0} que están asociados a los arcos de retorno y se activan con la variable de decisión a_{ij}; y los costos de los vehículos subcontratados que son penalizados por su utilización con la variable P.

Las ecuaciones en 3.2 a 3.14 definen las restricciones de la siguiente forma:

3.2. Cada nodo tendrá un único arco de llegada.

3.3. Permite asegurar que un nodo tendrá el mismo arco de entrada como de salida, es decir, la sumatoria de los arcos que interconectan clientes más la sumatoria de arcos que cierran rutas al depósito debe ser igual a la cantidad total de arcos de la ruta sin contar el retorno.

3.4. Garantiza que el número de arcos de salida de un depósito de las rutas propias sea igual al número de llegada al mismo.

3.5. Permite que se asegure que si está saliendo con arco de ruta subcontratada, se hubiese llegado con un arco del mismo tipo. También permite qué si se llega a un nodo terminal, no hayan rutas de salida o retorno al depósito.

3.6. Esta restricción evita la duplicación de arcos e indica el sentido del mismo.

3.7. Con esta restricción se garantiza el balance de flujos de demanda en los vehículos, ya sean de flota propia o subcontratada. Se tiene la variable t_{ij} y l_{ij} como la cantidad de carga que lleva un vehículo de flota propia y subcontratada respectivamente. Esta restricción indica que el flujo de carga que sale de un nodo es igual al flujo que llega más la demanda del mismo nodo.

3.8. Funciona como un identificador de arcos activos que generan topologías radiales.

3.9. Con esta restricción se limita el flujo de carga por una ruta propia, haciendo que no se sobrepase la capacidad de carga del vehículo.

3.10. Al igual que la restricción anterior, esta busca limitar el flujo de los vehículos, pero en este caso se trata de la flota subcontratada.

3.11. Si j es un nodo terminal de ruta propia ($z = 1$), la restricción impide que se genere un arco de retorno con la variable x.

3.12. Esta restricción obliga a que exista un arco de retorno con la variable a.

3.13. Se garantiza que el número de rutas es suficiente para atender toda la demanda de los clientes. Esto se consigue haciendo que la sumatoria de todos los arcos que salen del depósito sean igual (o eventualmente menor si la demanda puede ser atendida con menos vehículos) al cociente de la demanda total con la capacidad de los vehículos.

3.14. Con esta restricción se establece el número máximo de rutas que se pueden atender con la flota de vehículos propia.

En general, lo que se busca con este modelo matemático es la creación de soluciones con rutas construidas de manera radial, pero que para las rutas de vehículos propios se puedan generar arcos de retorno al depósito.

La solución a este problema por medio de este modelo matemático fue realizado en [20], donde se utilizó CPLEX 12.5 escrito en el lenguaje AMPL. El número de vehículos se estableció mediante la ecuación: $k = 0{,}8q/Q$, Donde q indica el valor de la demanda total de los clientes y Q corresponde al valor de la capacidad de los vehículos homogéneos. Los resultados serán comparados en el capítulo 5 con los obtenidos en este trabajo.

Capítulo 4

Construcción de la solución inicial

Para la mayoría de técnicas metaherísticas de búsqueda local, la construcción de la solución inicial es parte fundamental del desempeño. En este trabajo, se utiliza una heurística para construir las soluciones iniciales y para esto, se toman los conceptos presentados en Chu en 2005 [4] y la heurística de Clark and Wright denominada "algoritmo de ahorros" *(Savings algotithm).* Esta heurística es adaptada para resolver el VRPPC.

4.1. Algoritmo de ahorros

La implementación del algoritmo de ahorros se lleva a cabo de manera tal que se minimiza el costo asociado a la distancia recorrida. Esto se realiza a través de la modificación del cálculo del ahorro de Clark and Wright que minimiza directamente el valor de la distancia recorrida. En la fase de mejoramiento (Fase 2) se calcula nuevamente el ahorro haciendo combinaciones de las rutas inicialmente construidas. La inserción de clientes a las rutas se hace de forma paralela, esto permite inserciones en todas las rutas creadas en el avance del algoritmo.

El algoritmo sigue el procedimiento propuesto en [4], que consiste en seleccionar los clientes que serán atendidos por la flota propia y subcontratada. Esta selección se basa en ordenar de manera ascendente los clientes por el costo, debido a que este costo en los vehículos de la flota subcontratada al ser penalizado es mucho mayor al de la flota propia. Por lo tanto, los clientes con mayor costo de transporte en la flota subcontratada se priorizarán a ser atendidos con la flota propia.

La selección sigue los siguientes pasos:

1. Calcular la demanda de los clientes.
2. Calcular la capacidad total de la flota propia, que en este caso y para este problema es homogénea.
3. La demanda de los clientes deberá ser mayor a la capacidad de carga disponible por la flota propia, en caso contrario, el problema se convertiría en un CVRP con flota homogénea, ya que no se necesitaría de la flota subcontratada.
4. Ordenar los clientes de forma ascendente (de menor a mayor) para el caso de que sean atendidos por la flota subcontratada.
5. Con la lista del paso (4), sumar la demanda de los clientes, de forma que no se sobrepase la capacidad de la flota propia. Estos clientes serán atendidos por la flota propia y los restantes serán atendidos por la flota subcontratada.

De la forma anterior, se seleccionan los clientes que serán atendidos por cada una de las flotas, posteriormente tendrá una fase de mejoramiento donde se aplica nuevamente el cálculo del ahorro.

4.1.1. Cálculo del ahorro

Como se mencionó anteriormente, el cálculo del ahorro se realiza con base en costos. En el algoritmo propuesto en [5] por Clark and Wright [4] se plantea con base en las distancias. Se definen las variables así:

S_{ij}: ahorro calculado en costos al recorrer la trayectoria $0-i-j-0$ comparado con las trayectorias $0-i-0$ y $0-j-0$ de forma separada.

LTL_i: costo asociado a la utilización de un vehículo subcontratado para atender el cliente i desde el depósito. Este valor se calcula como $LTL_i = d_{ij} * CF * P$, donde la variable d_{ij} indica la distancia entre dos clientes, CF es el costo fijo de usar un vehículo y P es la penalización por utilizar el vehículo de la flota subcontratada. Generalmente el valor de CF se fija en la unidad, debido a que se trata de una constante proporcional qué sin importar su valor, no afecta el orden final.

T_{ij}: costo total asociado a la utilización de un vehículo propio para atender inicialmente desde el depósito al cliente i, después al cliente j y finalmente regresar al depósito. Este costo se calcula como $T_{ij} = \sum d_{ij} * v$, donde v indica el costo por unidad de desplazamiento ($/km).

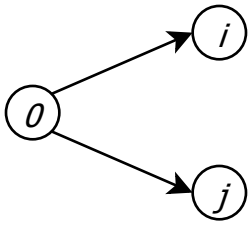

Figura 4.1: Rutas atendidas por vehículos subcontratados

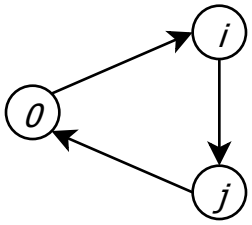

Figura 4.2: Ruta atendida por la flota propia

El ahorro para la primera fase se calcula de acuerdo a los gráficos 4.1 y 4.2:

Para la figura 4.1, los costos asociados al visitar los clientes i y j se obtienen de: $LTL_i + LTL_j$.

En la figura 4.2, Los costos asociados al visitar los clientes i y j con vehículo de flota propia se obtienen de: $\sum d_{0-i-j-0} * v = (d_{0i} + d_{ij} + d_{j0}) * v$.

Por lo tanto, el ahorro calculado de utilizar un vehículo de la flota propia en vez de uno subcontratado es definido como la diferencia entre el costo de visitar los clientes con flota subcontratada (mayor costo) y con flota propia:

$$S_{ij} = LTL_i + LTL_j - (d_{0i} + d_{ij} + d_{j0}) * v \tag{4.1}$$

De esta primera fase, las rutas subcontratadas estarán conformadas por un único nodo, por lo tanto, deberá realizarse un paso posterior que permita unificar eficientemente estas rutas en una o varias, de acuerdo a las limitaciones de capacidad de carga de los vehículos. Esto procedimiento se describe en la sección 4.1.3.

La segunda fase, el cálculo del ahorro se realiza de acuerdo a los gráficos 4.3 y 4.4:

En la la figura 4.3, los costos asociados de la ruta propia y la subcontratada se calculan como: $\sum d_{0-k-j-0} * v + LTL_i = (d_{0k} + d_{kj} + d_{j0}) * v + LTL_i$

De acuerdo a lo que se muestra en la figura 4.4, el cliente i que anteriormente se encontraba siendo atendido por uno de los vehículos de la flota subcontratada, pasaría a ser atendido por la flota propia. El costo se determina como: $\sum d_{0-k-j-i-0} * v = (d_{0k} + d_{kj} + d_{ji} + d_{i0}) * v$

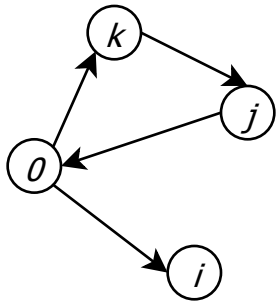

Figura 4.3: Ruta atendida con vehículo propio y ruta subcontratada

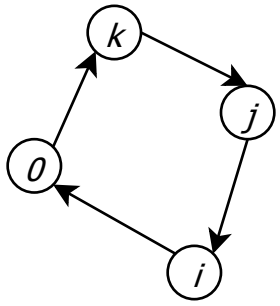

Figura 4.4: Ruta atendida únicamente con vehículo propio

De acuerdo a las dos posibles configuraciones anteriores, el valor del ahorro de la segunda fase se calcula como: $S_{ij} = [(d_{0k} + d_{kj} + d_{j0}) * v + LTL_i] - (d_{0k} + d_{kj} + d_{ji} + d_{i0}) * v$, simplificando:

$$S_{ij} = (d_{j0} - d_{ji} - d_{i0}) * v + LTL_i \tag{4.2}$$

El cálculo del ahorro en la segunda fase permite verificar si los clientes que están siendo visitados por vehículos de flota subcontratada pueden visitarse a menor costo con alguno de los vehículos de la flota propia.

4.1.2. Procedimiento de inserción (Descripción para implementación)

Continuando con la construcción de cada una de las rutas, es necesario construir un arreglo que contenga los datos de ahorros del problema. Para este arreglo se obtuvieron las dimensiones de la siguiente forma:

n: Número total de nodos

n_s :Número de nodos que en el proceso de selección inicial, deben ser atendidos por la flota subcontratada.

$N = n - n_s$, número de nodos disponibles para inserción

Se tiene por definición que la cantidad permutaciones posibles con r individuos para un conjunto x de nodos es $\frac{(x)!}{(x-r)!}$. Por lo tanto, para una cantidad de nodos N, la cantidad de permutaciones posibles teniendo en cuenta el problema simétrico, es: $\frac{(N)!}{(N-2)!*2}$. Se define entonces las dimensiones del arreglo como:

$$\left(\frac{(N)!}{(N-2)!*2}, 3\right) \tag{4.3}$$

El número de columnas se fija en tres con el fin de almacenar el par de nodos con su valor de ahorro. Este arreglo contiene el cálculo del ahorro para todas las posibles combinaciones entre los clientes.

El procedimiento para la inserción de nodos es el siguiente:

Fase 1:

1. Se obtiene el arreglo con los datos de los ahorros de acuerdo a lo mostrado en la subsección 4.1.3.

2. Ordenar el arreglo con los datos de los ahorros en orden descendente (de mayor a menor) de acuerdo a su valor de ahorro.

3. Desde el inicio del arreglo con los datos de los ahorros, definir como nodos de inicio al primer par de nodos (los que tienen mayor valor de ahorro).

4. Continuar a través de las filas del arreglo encontrando conexiones factibles con todos los pares de nodos. Una conexión factible se da cuando hay coincidencia entre un nodo extremo de la ruta y cualquiera de los dos nodos que se encuentran en el arreglo de los ahorros. Si hay coincidencia, el otro nodo entraría a formar parte de la ruta en el extremo donde se encuentra el nodo coincidente. El hecho de que esta conexión sea factible, implica qué al insertarse en la ruta, no se supere la capacidad del vehículo.

5. Si no es posible insertar alguno de los nodos del arreglo de ahorros en la ruta actual, se crea una nueva ruta con el par de nodos del arreglo de ahorros, verificando que haya vehículos de la flota propia disponibles. Sí ya no hay vehículos disponibles, se descarta el par de nodos y se continua con los siguientes arcos de la matriz de ahorros.

6. Repetir los pasos (4) y (5) hasta el final del arreglo de ahorros.

7. Buscar si hay nodos por asignar a rutas. En el caso de que haya nodos pendientes de asignación, se deberán crear una nueva ruta con este único nodo si hay vehículos propios disponibles. Sí ya no hay vehículos disponibles, se debe crear una ruta nueva de vehículos subcontratados.

Del procedimiento anterior, quedan construidas las rutas propias y las subcontratadas. Estas últimas, quedarán siempre con un único nodo. En la segunda fase, se procederá a insertar nodos que están con vehículos subcontratados a las rutas propias, por lo que se debe calcular un nuevo arreglo de ahorros donde se verifique a que ruta es mejor la inserción.

Fase 2:

1. Se verifica que vehículo tiene disponibilidad de carga para cada uno de los nodos que deben ser atendidos por flota subcontratada.
2. Se realiza la construcción de un nuevo arreglo de ahorros, el cual tendrá las siguientes dimensiones: $(V_s\ ,\ 3)$, donde V_s es el número de vehículos subcontratados obtenidos de la primera fase. Las tres columnas indican: (1) El vehículo de la ruta propia disponible. (2) El nodo de la ruta subcontratada. (3) El ahorro.
3. Ordenar la matriz de ahorros de la segunda fase en orden descendente (de mayor a menor) de acuerdo a su valor de ahorro.
4. Conectar los nodos de la flota subcontratada a las rutas de los vehículos propios. Esta conexión debe ser factible y la inserción se realiza al final de la ruta.
5. Repetir el paso anterior hasta que se realicen las inserciones factibles.
6. Los nodos que no se asignaron en esta etapa seguirán siendo de la flota subcontratada.

4.1.3. Algoritmo de ahorros para la flota subcontratada

Posterior a las fases anteriores, donde se construyeron las rutas para los vehículos propios y se identificaron los nodos que deben ser atendidos por la flota subcontratada, estos son propuestos para ser atendidos en una o más rutas. Para ello, se construye un algoritmo de ahorros clásico, donde el ahorro se calcula para las distancias.

En la figura 4.5, se muestran los gráficos para el cálculo del ahorro así:

$$S_{ij} = (d_{0i} + d_{oj} - d_{ij}) * v \tag{4.4}$$

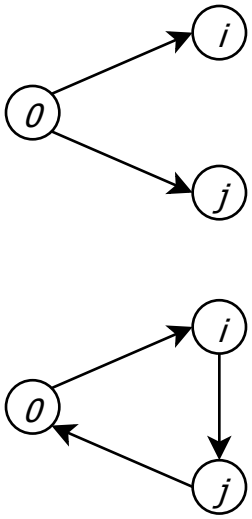

Figura 4.5: Cálculo ahorro nodos flota subcontratada

El resultado de este proceso es la construcción de una o varias rutas de los nodos que deben ser atendidos por la flota subcontratada. Este procedimiento también realiza inserciones en paralelo con los siguientes pasos:

1. Desde el inicio del arreglo con los datos de los ahorros, definir como nodos de inicio al primer par de nodos (los que tienen mayor valor de ahorro).

2. Continuar a través de las filas del arreglo encontrando conexiones factibles con todos los pares de nodos. Una conexión factible se da cuando hay coincidencia entre un nodo extremo de la ruta y cualquiera de los dos nodos que se encuentran en el arreglo de los ahorros. Si hay coincidencia, el otro nodo entraría a formar parte de la ruta en el extremo donde se encuentra el nodo coincidente. El hecho de que esta conexión sea factible, implica que al insertarse en la ruta, no se supere la capacidad del vehículo.

3. Si no es posible insertar alguno de los nodos del arreglo de ahorros en la ruta actual, se crea una nueva ruta con el par de nodos del arreglo de ahorros. No hay restricción con la creación de nuevas rutas.

4. Repetir los pasos (4) y (5) hasta el final del arreglo de ahorros.

4.1.4. Etapa de mejoramiento

Al finalizar las ejecuciones de las funciones que ejecutan el algoritmo de ahorros general para el VRPPC y el algoritmo de ahorros para los nodos de las rutas subcontratadas, es necesario construir

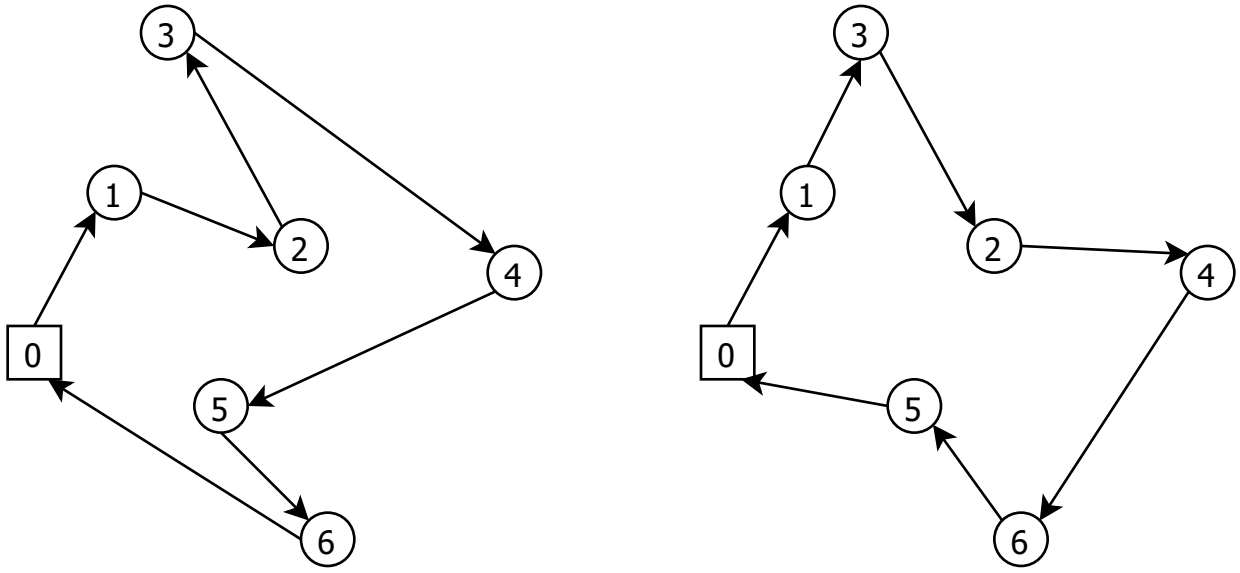

Figura 4.6: Mejoramiento Or-opt

una estructura de datos que almacene la solución completa (un arreglo que almacene conjuntamente las rutas propias y subcontratadas), con el fin de que posteriormente pueda aplicarse toda clase de exploraciones por medio de intercambios de clientes que permitan obtener las mejores soluciones al problema general del VRPPC.

El mejoramiento que se realiza posterior a la construcción de la matriz de solución, que contiene en primera instancia la solución inicial obtenida de los pasos de la aplicación del algoritmo de ahorros, es denominado como un mejoramiento intra - ruta: *Or - opt*. Este mejoramiento consiste en qué para cada una de las rutas, se remueve uno de los clientes y es insertado en otra posición de la misma ruta. El procedimiento se realiza para cada ruta, cada cliente y para cada posición en la ruta, y la mejor solución es almacenada para modificar la solución inicial. Este mejoramiento permite refinar la ruta qué a su vez, mejora su función objetivo. En la figura 4.6, se muestra un ejemplo de este mejoramiento donde se eliminan las trayectorias en "zig-zag" que deterioran la función objetivo.

Como lo muestra la figura 4.6, el mejoramiento consistió en la aplicación de los intercambios que modifican la ruta así: $[0, 1, 2, 3, 4, 5, 6, 0] \rightarrow [0, 1, 3, 2, 4, 6, 5, 0]$. A pesar de que en el ejemplo se muestra un mejoramiento para una ruta propia (termina en el depósito).

Posterior a este mejoramiento, se obtiene la primera solución usada en el algoritmo ILS. No obstante, esta no será la única solución inicial del ILS, como se verá más adelante, a esta se efectuan algunas variaciones que permitirán al algoritmo ILS iniciar su búsqueda local desde diferentes puntos, mejorando la diversificación de la técnica metaheurística.

Capítulo 5

Algoritmo ILS para la solución del VRPPC

La metaheurística *Iterated Local Search* (ILS) genera subconjuntos del espacio de solución alrededor de soluciones óptimas locales, en las cuales se ejecutarán procedimientos de búsqueda local aplicando adecuados esquemas de vecindad, y para salir de los óptimos locales se aplican procedimientos de diversificación, que permiten salir a otros espacios de solución. Las características del *Variable Neighborhood Search* (VNS) se basan en la utilización de los operadores de búsqueda en los vecindarios. Se denominará *Random Variable Neighborhood Search* (RVNS), a la selección aleatoria de los operadores en la búsqueda local, denominados intra-ruta e inter-ruta.

La implementación del algoritmo ILS comprende los siguientes pasos:

1. Generación de la solución inicial: Se construye una solución usando una heurística constructiva, y a partir de esta, iniciar la búsqueda local que permita llegar a mejores soluciones. Su construcción fue descrita en el capítulo 4. Esta solución es el punto de partida del algoritmo en la primera iteración. La construcción de la solución inicial varía a través de la ejecución general, permitiendo elegir aleatoriamente entre la solución única del algoritmo de ahorros y alternaciones a esta, que permitan identificar un conjunto de soluciones, logrando obtener múltiples puntos de partida para el algoritmo ILS.

2. Búsqueda Local: Aplica mejoramientos continuos a través de la búsqueda dentro de los vecindarios. Estas búsquedas se realizan por medio de operadores que aplican movimientos inicialmente inter-ruta y posteriormente a la mejor solución encontrada se realizan movimientos intra-ruta.

3. Perturbación: Aplicación de movimientos aleatorios inter-ruta (sin importar si exista mejoramiento). A partir de estos movimientos, se logra escapar de óptimos locales, y llegar a otras soluciones del espacio de solución del problema. Estos movimientos también son denominados de diversificación, ya que permiten al algoritmo encontrar una nuevo punto desde el cual es iniciado un proceso de búsqueda local.

4. Criterio de aceptación: Determina cuando el algoritmo debe detenerse en la búsqueda de soluciones. Este criterio básicamente es la definición de un número máximo de iteraciones en la búsqueda local y en la ejecución del algoritmo completo, es decir, se tendrán dos parámetros como criterio de aceptación que se definen como:

    - Iteraciones locales del ILS: corresponde al número de iteraciones del algoritmo donde se ejecutan los pasos del (1) al (3).
    - Iteraciones globales del ILS: corresponde al número de iteraciones donde se inicia nuevamente el algoritmo con la obtención de una nueva solución del espacio de búsqueda.

El algoritmo general del ILS es presentado en el algoritmo 1 . Entre las líneas (2) y (11) del algoritmo, se definen las variables del proceso, que son detalladas en el cuadro 5.1. Las demás variables corresponden a los parámetros del problema y se detallan en los cuadros 5.2 y 5.3.

Algoritmo 1 (Descripción para implementación)

Se definen las siguientes variables: para el control de la diversificación y la intensificación del algoritmo,

- *FI: Factor de iteración.* Posterior a la calibración del algoritmo, se define un numero básico de iteraciones en las que este funciona de manera eficiente, por lo tanto, el factor de iteración define cuantas veces se debe repetir la ejecución con el fin de aumentar la intensificación.

- *FD*: *Factor de Distancia.* Indica el valor de cuantas veces la distancia de referencia se utiliza para realizar movimientos de perturbación. La distancia de referencia es la que se encuentra desde el depósito hasta el centroide generado por cada ruta. El criterio para la utilización de valores de distancia se describe en la subsección 5.5.1.

Las iteraciones globales del algoritmo inician en la línea (12) y finaliza en la línea (41). En la línea (13) inicia el contador de iteraciones locales. En la línea (14) se obtienen la solución inicial para la iteración actual, la cual se describe en la sección 5.1. En la línea (16) se obtiene el valor de la función objetivo de referencia para la primera iteración la cual es equivalente a la solución

Algoritmo 1: ILS

```
1:  Procedimiento ILS;
2:  FI←Int;
3:  MaxIter← 75 ∗ FI ;                                  // parámetro de ejecución
4:  B← 20 ;                                             // parámetro de ejecución
5:  Fobj←inf;
6:  NStatus←NStatus(V);
7:  IterILS←nodos + B ∗ V F ←Inf;
8:  AMIPP ←AIP(S0ini, Coor, 1);
9:  TM ←15;
10: Mem ←MemSol(S0ini, V, m, TM);
11: cm ←1;
12: for (i = 1; i < MaxIter; i + +) do
13:     Iter ← 0;
14:     [S0] ←SolucionInicial(S, Mem, TM, AMIPP);
15:     S1 ←S0;
16:     F ←FuncionObjetivo(S0, Svs, Dist, Cv, CF, Pen);
17:     while (IterILS > Iter) do
18:         FD ←FactorDist();
19:         [S0, NStatus] ←RVNS(S0, Svs, Dist, Cv, Cf, Vcap, q, Pen, Nstatus);
20:         F1 ←FuncionObjetivo(S0, Svs, Dist, Cv, Cf, Pen);
21:         if (F1 < F) then
22:             S1 ←S0;
23:             Iter ← 0;
24:             F ←F1;
25:             AMIPP ←AIP(S0ini, Coor, FD);
26:         end
27:         [S0, NStatus] ←Perturb(S1, Vcap, q, NStatus, AMIPP);
28:         Iter + +;
29:     end
30:     if (F < Fobj) then
31:         S ←S1;
32:         Fobj ←F;
33:     end
34:     if (F <= Fobj ∗ 1,1)&(F! = Fobj) then
35:         Mem(V, m, cm) ←S1 ;              // m:número de columnas de la solución
36:         cm + +;
37:         if (cm >= TM − 1) then
38:             cm ←1;
39:         end
40:     end
41: end
```

Variable	Tipo	Descripción
MaxIter	Entero	Es el parámetro que indica la cantidad de iteraciones generales del algoritmo.
B	Entero	Esta variable se utiliza para construir la relación entre la cantidad de iteraciones locales y el número de vehículos del problema.
Fobj	Entero	Función objetivo general.
NStatus	Arreglo	Esta variable es un arreglo binario multidimensional [r1][r2][N] que inicia con todos sus valores en uno (1) por medio de la función que lleva su mismo nombre, donde r1 y r2 son el número de vehículos del problema y la variable N indica la cantidad de vecindarios utilizados que en este trabajo son (N=8). Sus valores se convierten en cero (0) cuando un operador fue utilizado para la búsqueda local y este falló en encontrar una solución mejor. Sus valores cambiarán ha uno (1) cuando el vecindario a cambiado, es decir, cuando se aplica un procedimiento de perturbación. El uso de esta variable evita repetir intercambios que no mejoran la función objetivo.
IterILS	Entero	Es el parámetro de iteraciones locales.
F	Float	Variable auxiliar que se utiliza para el almacenamiento de la actual función objetivo (también puede ser llamada función objetivo local). Inicia con el valor entregado de la solución inicial.
S0ini	Arreglo	Variable auxiliar que se utiliza para el almacenamiento de la solución inicial (las rutas entregadas por el algoritmo de ahorros). La solución inicial general del ILS es generada por medio de la función SolucionInicial(), la cual se explicara en detalle más adelante.
F1	Float	Variable auxiliar que almacena la mejor función objetivo encontrada después de aplicar el RVNS y se actualiza en cada iteración.
S0	Arreglo	Este arreglo contiene la solución inicial general del algoritmo. Sus dimensiones vienen definidas desde la aplicación del algoritmo de ahorros.
S1	Arreglo	Arreglo que contiene la mejor solución actual local.
S	Arreglo	Solución actual general del problema.

Cuadro 5.1: Variables generales ILS

Variable	Tipo	Descripción
AMIPP	Arreglo	(Arreglo de intercambios permitidos en perturbaciones) Variable como arreglo binario que indica de acuerdo al FD (factor de distancia) que nodos son perturbables para cada ruta. Es decir, su dimensión será [#rutas][#nodos], por ejemplo, [2][5]=1 indica que para la ruta dos (2) el nodo cinco (5) es un candidato a trasladarse desde la ruta en la que se encuentre, hacia la ruta dos (2). La función asociada a esta variable AIP construye el arreglo encontrando para cada uno de los centroides de las rutas como el promedio de la distancia de cada nodo al depósito y escogiendo candidatos de acuerdo al factor de distancia. La elección de candidatos para las perturbaciones se explicara más adelante en la sección 5.5.
TM	Entero	Variable que indica la cantidad de soluciones a almacenar en la variable Mem que serán utilizadas para inicial el algoritmo.
Mem	Arreglo	Arreglo multidimensional en cual almacena la cantidad de TM soluciones encontradas. Estas soluciones son utilizadas como solución inicial del algoritmo.
cm	Entero	Contador de utilización de memoria de Mem.

Cuadro 5.2: Variables de diversificación del ILS

Variable	Tipo	Descripción
Vcap	Float	La capacidad individual de los vehículos homogéneos.
q	Arreglo	Arreglo que contiene la demanda de cada cliente.
Dist	Arreglo	Arreglo que contiene las distancias entre nodos. Su dimensión es [n][n] y las distancias son calculadas como euclidianas.
Cv	Float	Costo de los vehículos en $/distancia.
Cf	Float	Costo fijo por la utilización de un vehículo.
Pen	Entero	Factor de penalización por la utilización de vehículos subcontratados.

Cuadro 5.3: Variables de parámetros del problema

inicial. En la línea (17) se inician las iteraciones de la búsqueda local. La variable que controla la diversificación FD es obtenida a través de una función propia (línea 18), que permiten que sus valores cambien en cada iteración de la siguiente forma: *FD:* su función *FactorDist()* elige valores entre los tres valores del conjunto $[1, 2, 10]$, lo que permite variar el radio de la distancia de aceptación de nodos que pueden ser transferidos a través del arreglo *AMIPP,* en el proceso de perturbación, ver subsección: 5.5.1. En la línea (19) se genera el mejoramiento a través del proceso de búsqueda local por RVNS que será detallado en la sección 5.2. La variable *Nstatus* contiene las combinaciones de rutas que fallaron en el mejoramiento. En la línea (20) se calcula la función objetivo de la solución encontrada a través del RVNS y almacenada en la variable *S0.* Entre las líneas (21-26) se actualiza la mejor solución encontrada. Se observa que en la línea 25 se calcula nuevamente la variable *AMIPP,* donde se actualizan sus valores de acuerdo a esta nueva solución encontrada y que será perturbada en la línea (27), la función correspondiente al proceso de perturbación se detallan en la sección 5.5. Entre las líneas (30-33) se almacena la mejor solución encontrada en las iteraciones locales. El almacenamiento de estas soluciones sucede entre las líneas (34-40), allí la variable *Mem* es actualizada ingresando cada solución encontrada que sea por lo menos un 10 % mayor a la mejor solución encontrada (Mayor debido a que es un problema de minimización). En la línea 38 se encuentra el condicional que permite reiniciar los espacios de almacenamiento de la variable *Mem,* que fueron llenados previamente en la línea (10) con la solución inicial. Se observa en este condicional que se reserva el último espacio de la variable *Mem* para que allí siempre esté disponible la solución inicial obtenida del algoritmo de ahorros.

5.1. Solución inicial

La solución inicial es obtenida de la heurística de ahorros modificada, la cual es alternada con otras soluciones obtenidas, con el fin de contar con diversos puntos de inicio.

Para contar con un adecuado rango de diversificación, es necesario que cada una de las soluciones sea levemente perturbada para luego ser utilizada en la búsqueda local, logrando así una cantidad significativa de puntos de inicio de buena calidad. Para la utilización de estas soluciones como inicio, son almacenadas en la variable *Mem,* que almacena las soluciones que sean mejores a la actual o en un rango del 10 %.

Algoritmo 2 (Descripción para implementación)

Las instrucciones son básicamente la selección aleatoria de dos caminos, en el primero es elegido aleatoriamente una de las mejores soluciones encontradas, y el segundo es la mejor solución actual.

Es evidente que la mejor solución también está contenida en el grupo de las mejores soluciones, pero su probabilidad de ser elegida dependerá del tamaño de soluciones que se almacenen. Después de elegida la solución inicial, se procede a realizar un proceso de perturbación con un factor $FP = 0{,}5$, es decir, se aplica la mitad del número de perturbaciones que se eligen aleatoriamente (ver detalles en sección: 5.5). Hacer este factor mayor provoca que el algoritmo no se comporte de manera eficiente debido a que necesitará de más iteraciones de búsqueda local. Los operadores de perturbación serán detallados en la sección 5.5, al igual que el operador de mejoramiento intra-ruta, que será detallado en la sección 5.4. Se puede además observar que estas perturbaciones que allí se realizan no tienen restricción de distancia o factor *FD*.

Algoritmo 2: Función SolucionInicial

```
1:  [S0] = SolucionInicial(S, Mem, TM, V, AMIPP);
2:  x ←Round(1, 10) ;                          // aleatorio con dominio [1,10]
3:  if (x >= 6) then
4:      y ←Round(1, TM);
5:      S0 ←Mem(V, m, y);
6:      AMIPP ←Ones ;                          // llena la matriz con unos
7:      z ←Round(1, 10);
8:      FP ←0,5;
9:      if (z >= 4) then
10:         S0 ←MShift(1, 1, S0, FP, AMIPP) ;  // operador de perturbación
11:     else
12:         S0 ←MSwap(1, 1, S0, FP, AMIPP) ;   // operador de perturbación
13:     end
14:     S0 ←OrOpt(1, S0) ;                     // mejoramiento intra-ruta
15: else
16:     S0 ←S;
17:     AMIPP ←Ones ;                          // llena la matriz con unos
18:     z ←Round(1, 10);
19:     FP ←0,5;
20:     if (z >= 4) then
21:         S0 ←MShift(1, 1, S0, FP, AMIPP) ;  // operador de perturbación
22:     else
23:         S0 ←MSwap(1, 1, S0, FP, AMIPP) ;   // operador de perturbación
24:     end
25:     S0 ←OrOpt(1, S0) ;                     // mejoramiento intra-ruta
26: end
```

5.2. Random Variable Neighborhood Search (RNVS)

Esta función agrupa básicamente la búsqueda local, con la característica de que se construyen vecindarios elegidos aleatoriamente de un conjunto definido $NL = [N^1, N^2, N^3..., N^n]$. Estos vecindarios son construidos a través de estructuras de intercambio Inter-ruta, donde cada vecino corresponde a una solución factible, formada del intercambio de clientes entre rutas. El RVNS explora todos los vecinos de manera exhaustiva, y almacena la mejor solución utilizando en este proceso operadores de intercambio intra-ruta. En la figura 5.1 se muestra un esquema que describe el procedimiento de búsqueda que se utiliza en el RVNS. Allí se parte de la solución inicial descrita en la sección anterior, se elige un operador para la construcción de un vecindario del conjunto NL de manera aleatoria, el cual se explora de manera exhaustiva y en el caso de mejora, se aplica una etapa de mejoramiento por medio de la selección de una estructura intra-ruta. En el caso contrario, se elimina el operador de la lista y se verifica que hayan operadores disponibles y se reinicia el proceso. Posterior a los mejoramientos Intra-ruta, se reinicia el vecindario, por lo que los operadores que fueron eliminados se restablecen. Cuando ya no hay más operadores el proceso termina. En el algoritmo 3, se describe de forma detallada las instrucciones de la función RVNS.

Algoritmo 3 (Descripción para implementación)

En la línea (2) se define la variable *NLInter* que contiene el listado de vecindarios, en la línea (3) de define la variable auxiliar *F1* que contiene la función objetivo de la solución inicial de entrada. Entre las líneas (4-26) se efectúan las iteraciones de la búsqueda local, el cual solo se detendrá cuando la lista de vecindarios no contenga ninguno activo. La variable *NInter* almacena un número aleatorio que indica cual vecindario se eligió para la actual iteración. En la línea (6) se define la función *NeighborInter* que contiene todas las estructuras Inter - ruta implementadas, el número aleatorio de *NInter* elige que estructura utilizar. La variable auxiliar *F2* almacena la función objetivo de la mejor solución, después de aplicar las estructuras Inter-ruta. En la línea (8) se verifica la existencia de mejoramiento, en caso afirmativo, se inicia el proceso de mejoramiento Intra-ruta que se define en las líneas (9-22). La variable *NIntra* contiene el listado de estructuras Intra-ruta. En las líneas siguientes se aplican los intercambios seleccionando las estructuras de manera aleatoria, se eliminan las que no presentan mejoramiento. Finalmente se reinicia el vecindario en la línea (22). En la línea 24 se eliminan los vecindarios que no presentaron mejora la función objetivo.

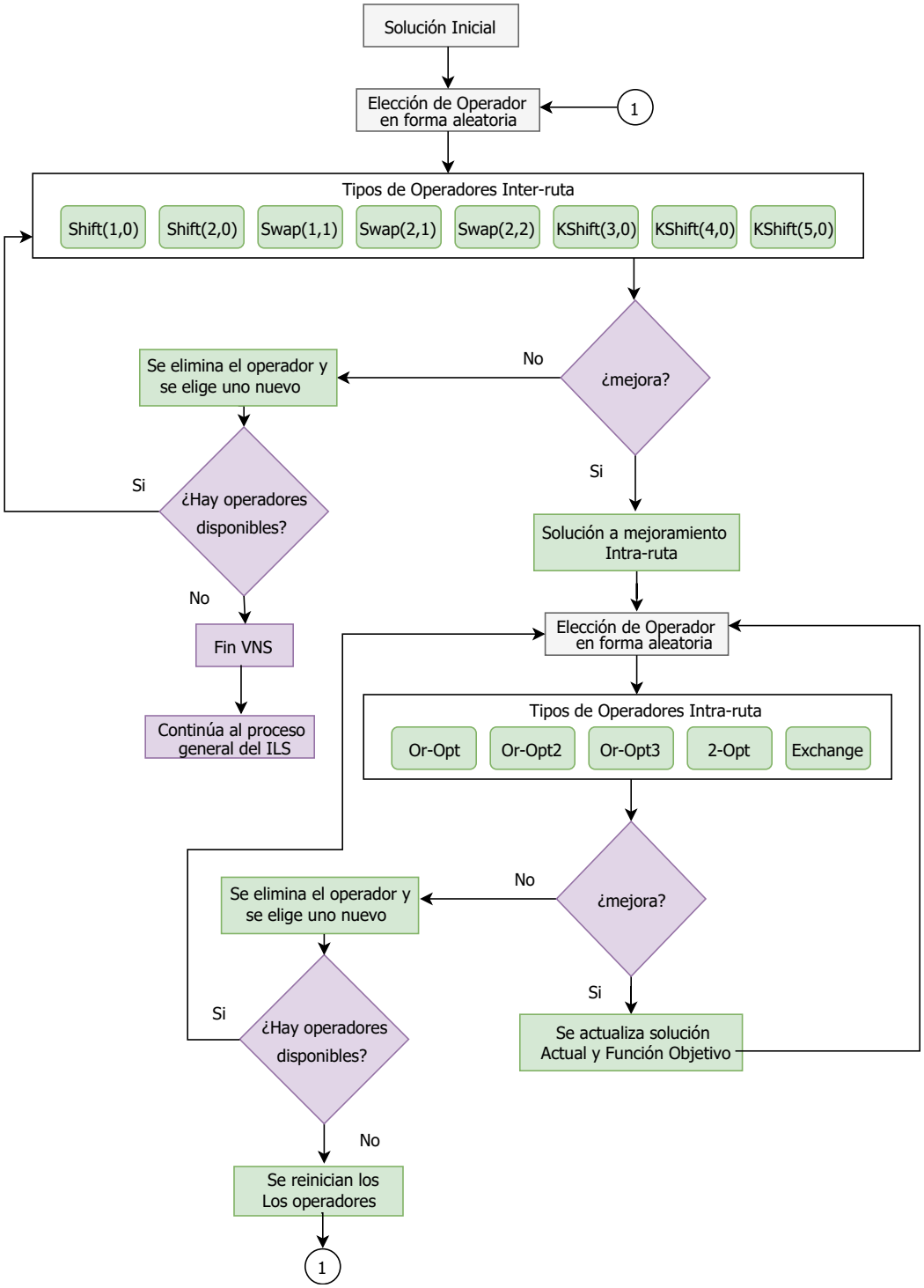

Figura 5.1: Esquema RVNS

Algoritmo 3: Función RVNS

```
1:  [S, NStatus] = RVNS(S0, Svs, NStatus);
2:  NLInter ←[1, 2, 3, 4, 5, 6, 7, 8];
3:  F1 ←Fo(S0, Dist, Cf, Cv);
4:  while max(NLInter)! = 0 do
5:      NInter ←Aleatorio(NLInter);
6:      [S0, NStatus] ←NeighborInter(Ninter, S0, Svs, NStatus);
7:      F2 ←Fo(S0, Dist, Cv, Cf);
8:      if F2 < F1 then
9:          NLIntra ← [1, 2, 3, 4, 5];
10:         while max(NLIntra)! = 0 do
11:             NIntra ←Aleatorio(NLIntra);
12:             SS ←NeighborIntra(NIntra, S0, Svs);
13:             Fs ←Fo(SS, Dist, Cv, Cf);
14:             if Fs >= F2 then
15:                 [NLIntra] ←DelNeighbor(NLIntra, NIntra);
16:             else
17:                 F2 ←FS;
18:                 S0 ←SS;
19:             end
20:         end
21:         F1 ←F2;
22:         NLInter ←[1, 2, 3, 4, 5, 6, 7, 8];
23:     else
24:         NLInter ←DelNeighbor(NLInter, NInter);
25:     end
26: end
27: S ←S0;
```

5.3. Operadores Inter - Ruta

Esta función contiene cada una de las estructuras encargadas de crear vecindarios por medio de intercambio de clientes entre las rutas. La elección de cada estructura se realiza de manera aleatoria, en cada iteración se descartan las que no mejoran la solución. Se finaliza cuando todos hayan sido descartados. Las estructuras Inter-ruta son descritas a continuación:

5.3.1. Shift

La estructura de intercambio Shift transfiere un cliente desde una ruta r_1 a la ruta r_2. El nuevo cliente que llega a la ruta r_2 construirá una cantidad de vecindarios equivalentes al número de posiciones posibles donde se puede ubicar en la ruta r_1. Este intercambio se realiza de manera exhaustiva, es decir, todos los nodos serán probados en cada una de las rutas en todas sus posiciones posibles.

Esta estructura puede definirse de manera tal que se indica el número de nodos que se transfieren. El procedimiento fue propuesto por Osman (1993) [15], donde el número de nodos que se transfieren está determinado por la variable λ. En este trabajo y como es común en otros, se define el conjunto $\lambda = \{1, 2\}$. Por lo tanto, se define la función $Shift(\lambda, 0)$ donde el valor $\lambda = 2$ implica la transferencia de dos nodos consecutivos. Las instrucciones que describen esta función están en el algoritmo 4.

Algoritmo 4 (Descripción para implementación)

En primera instancia, se usan las variables globales que son requeridas para el cálculo de las funciones objetivo. La variable *"a"* representa el valor λ de la función. En la línea (2) se define la variable *Rx* que indica el número total de rutas que tiene la solución *S0* (solución inicial). En las líneas (5) y (6) se inician los bucles principales que recorren todas las filas de la matriz de soluciones *S0* y eligen el par de rutas *r1* y *r2* para las transferencias. Los condicionales de las líneas (8) y (9) verifican los estados del vecindario y la factibilidad general de la transferencia respectivamente. En las líneas (10) y (11) se inician los bucles que permiten seleccionar los espacios *"in"* donde será insertado el nodo *"z"* . Las variables *Colr1* y *Colr2* contienen el índice donde termina la ruta, estos valores son importantes teniendo en cuenta que el vector donde se almacena la ruta contiene valores excedentes de memoria con valor en cero. La factibilidad de la actual iteración es verificada en la línea (13). En la línea (24) se genera un condicional, el cual permite verificar si en alguna de las iteraciones presentan mejoramiento, en caso afirmativo, es necesario actualizar la función objetivo,

que implica que si esta no tuvo ningún cambio se debe anular la utilización del vecindario, esto se refleja entre las líneas (29-31). La función Shift ocupa los vecindarios $NL = [N^1, N^2]$ que son representados por la variable *"n"*, donde si $a = 1 \rightarrow n = 1$ y si $a = 2 \rightarrow n = 2$.

En la figura 5.2 se muestra un ejemplo de la transferencia de nodos entre rutas, con $\lambda = 1$ se transfiere el nodo (11) y $\lambda = 2$ se transfieren los nodos (11, 6). En este ejemplo las dos rutas son atendidas con vehículos de flota propia.

5.3.2. Swap

El operador Swap realiza intercambios entre un par de rutas r_1 y r_2. A diferencia del operador Shift, este asegura que cada ruta recibirá uno o dos nodos. El número de nodos a transferir están definidos por dos valores de $\lambda \leq 2$ de la forma $Swap(\lambda_1, \lambda_2)$. Esto permite una adecuada transferencia de nodos entre rutas, con un amplio espectro de exploración.

Algoritmo 5 (Descripción para implementación)

En el algoritmo 5 se describen las instrucciones para ejecutar los intercambios entre rutas. Los intercambios se inician a partir de la línea (5), donde se generan los bucles que recorren la solución *S0*. Entre las líneas (8-9) se realizan la verificación de factibilidad, para dar inicio a los bucles de intercambio en las líneas (10-11). Las variables *in1, in2,* indican los espacios en los que se insertaran los clientes de la otra ruta. Los nodos o clientes que se insertaran en estos espacios se deben seleccionar por medio de un par de bucles que se deben construir al interior del condicional de la línea (15). Pero antes de llegar allí, en las líneas (12-13) se calcula la carga actual de la ruta a través de las variables *TR1* y *TR2,* donde se debe tener en cuenta que esta carga actual se obtiene después de ceder los $\lambda \leq 2$ nodos. Finalmente se actualiza la variable *NStatus* que como fue descrito anteriormente, es un arreglo binario que indica si un movimiento es de mejor o peor calidad. La función $Swap(\lambda_1, \lambda_2)$ identifica los vecindarios $NL = [N^3, N^4, N^5]$.

En la figura 5.3, se muestra la utilización de este operador, compuesto por una ruta propia y otra subcontratada. Para el caso $Swap(1, 1)$, se intercambian los nodos [(6) (11)], en $Swap(2, 1)$ el intercambio se realiza con los nodos [(5,6) 11]. Finalmente, en el intercambio $Swap(2, 2)$ se realiza con los nodos [(5,6) (10,11)].

5.3.3. K-Shift

Este operador consiste en la transferencia de *K* clientes consecutivos desde la ruta r_1 al final de la ruta r_2. El valor de *K* puede variar dependiendo de la cantidad de nodos del problema, ya que

Algoritmo 4: Función Shift

```
   // Cargar Variables globale Dist,Cv,Cf,Vcap,q,Pen,NStatus
 1: [S, NStatus] = Shift(a, S0, Svs);
 2: Rx ←size(S0);
 3: Fobj ←Fo(S0, Dist, Cv, Cf);
 4: Fini ←Fobj0;
   // Inicio de trasnferencia de clientes de r1 a r2
 5: for (r1 = 1; r1 <= Rx; r1 + +) do
 6:     for (r2 = 1; r2 <= Rx; r2 + +) do
 7:         if (r1! = r2) then
 8:             if (NStatus[n][r1][r2] == 1) then
 9:                 if (MinCarga(r1) + Carga(r2) <= Vcap) then
10:                     for (in = 2; in <= Colr2; in + +) do
11:                         for (z = 2; z <= Colr1; z + +) do
                               // in:inserciones en el espacio in de r2
                               // z:inserción del nodo z de la ruta r1 en espacio in
12:                            TR1 ←q[S0[r1][z]][2] + q[S0[r1][z] + 1][2] ∗ (a − 1);
                               // TR1:Variable demanda nodos trasladados
13:                            if (Carga(r2, 1) + TR1 <= Vcap) then
14:                                Se realiza la transferencia de los 'a' nodos;
15:                                if (Fobj < Fobj0) then
16:                                    Fobj0 ←Fobj;
17:                                    M ←1 ;                    // hubo mejoramiento
18:                                end
19:                            end
20:                        end
21:                    end
22:                end
23:            end
24:        end
25:        if (M == 1) then
26:            Fini ←Fobj0;
27:            Actualiza variable Carga;
28:            M ←0;
29:        end
30:    end
31:    if (Fini == Fobj0) then
32:        NStatus[n][r1] ←0;
33:    end
34: end
```

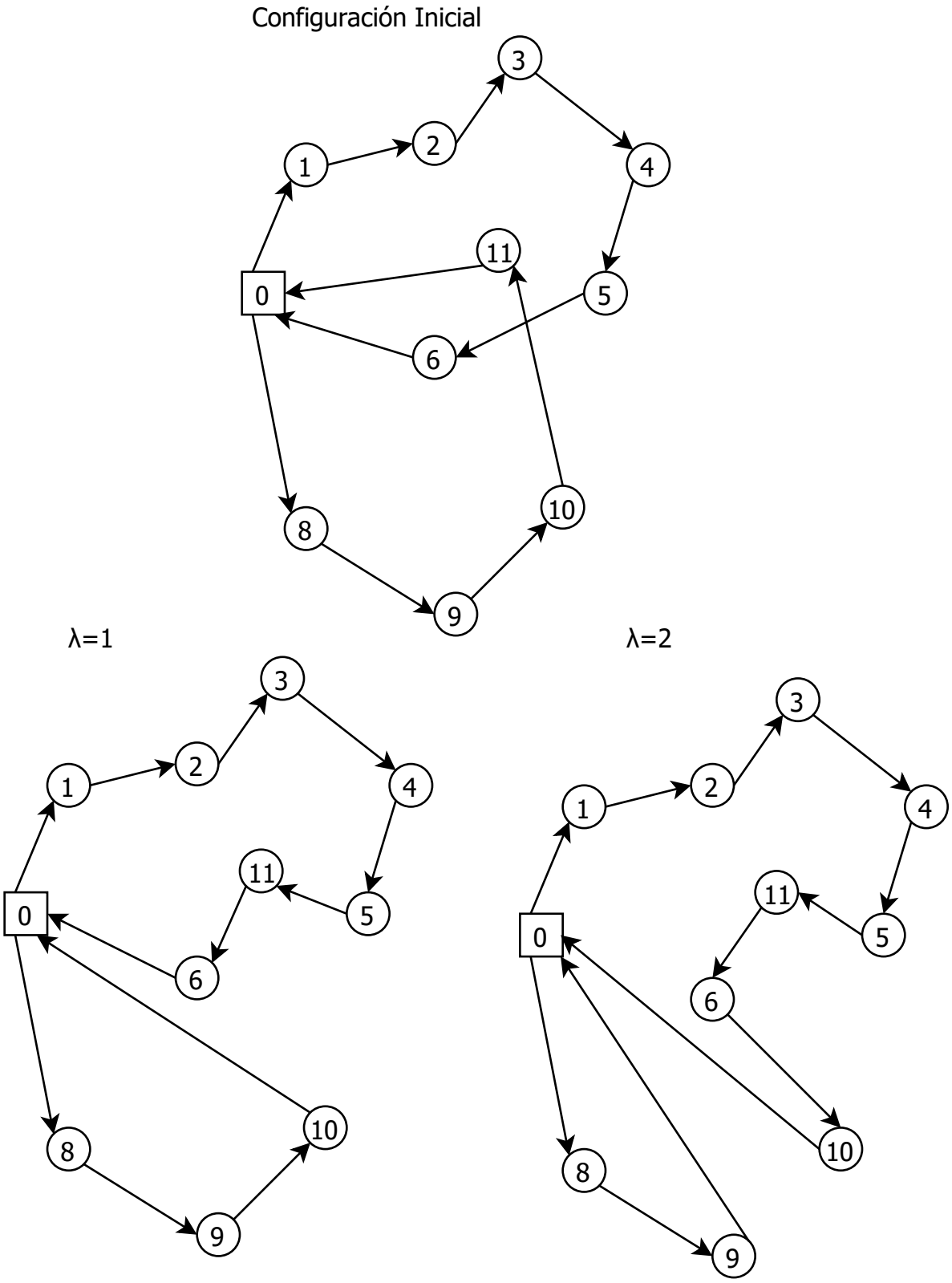

Figura 5.2: Operador $Shift(\lambda, 0)$

Algoritmo 5: Función Swap

```
   // Cargar Variables globale Dist,Cv,Cf,Vcap,q,Pen,NStatus
 1: [S, NStatus] = Swap(a, b, S0, Svs);
 2: Rx ←size(S0);
 3: Fobj ←Fo(S0, Dist, Cv, Cf);
 4: Fini ←Fobj0;
 5: for (r1 = 1; r1 <= Rx; r1 + +) do
 6:     for (r2 = 1; r2 <= Rx; r2 + +) do
 7:         if (r1! = r2) then
 8:             if (NStatus[n][r1][r2] == 1 then
 9:                 if (MinCarga(r1) − MaxCarga(r2) + Carga(r2) <= Vcap) then
10:                     for (in1 = 2; in <= Colr2; in + +) do
11:                         for (in2 = 2; z <= Colr1; z + +) do
12:                             TR1 ←CargaR1 − q[S0[r1][in1]][2] + q[S0[r1][in1] + 1][2] ∗ (a − 1);
13:                             TR2 ←CargaR2 − q[S0[r2][in2]][2] + q[S0[r2][in2] + 1][2] ∗ (b − 1);
14:                             if (Carga(r2, 1) + TR1 <= Vcap) then
15:                                 if (Carga(r1, 1) + TR2 <= Vcap) then
16:                                     Se realiza la transferencia de los 'a' nodos a r1;
17:                                     Se realiza la transferencia de los 'b' nodos a r2;
18:                                     if (Fobj < Fobj0) then
19:                                         Fobj0 ←Fobj;
20:                                         M ←1 ;                    // hubo mejoramiento
21:                                     end
22:                                 end
23:                             end
24:                         end
25:                     end
26:                 end
27:             end
28:         end
29:         if (M == 1) then
30:             Fini ←Fobj0;
31:             Actualiza variable Carga;
32:             M ←0;
33:         end
34:         if (Fini == Fobj0) then
35:             NStatus[n][r2] ←0;
36:         end
37:     end
38:     if (Fini == Fobj0) then
39:         NStatus[n][r1] ←0;
40:     end
41: end
```

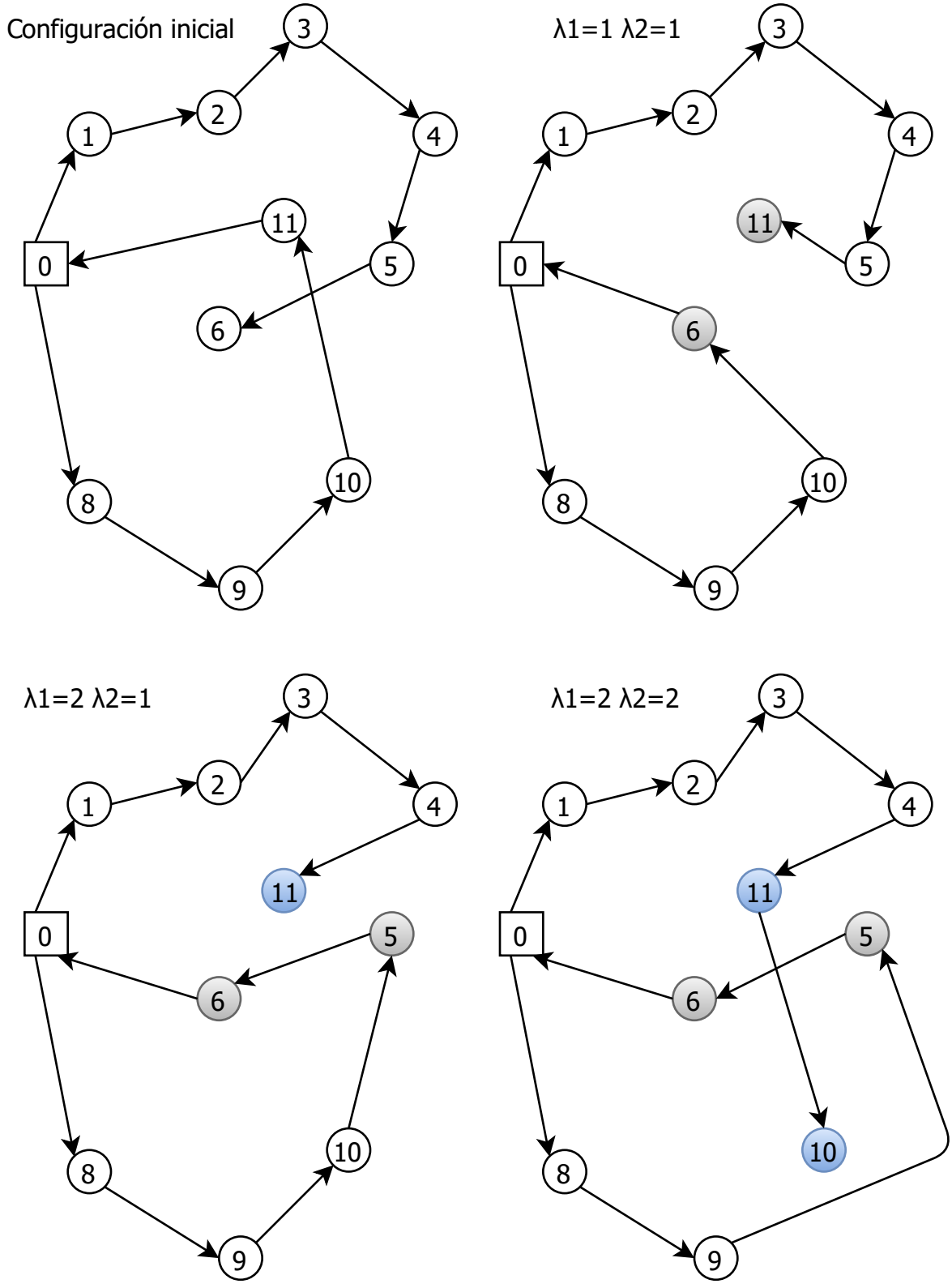

Figura 5.3: Operador $Swap(\lambda_1, \lambda_2)$

valores muy grandes tienden a ser infactibles. Por obvias razones, este valor será $K \geq 3$ ya que la función $Shift$ tiene valores $\lambda = 2$. Con respecto a las instrucciones requeridas para construir el algoritmo que efectúe la función $K - Shift$, son las mismas de la función $Shift(\lambda, 0)$, con la diferencia que el espacio de transferencia es al final de la ruta, lo que significa que es un operador con una complejidad computacional más baja que los otros operadores inter-ruta que generan vecindarios. Esta función ocupa los vecindarios $NL = [N^6, N^7, N^8]$.

5.4. Operadores Intra - Ruta

Para el tratamiento intra-ruta, son usados cinco (5) operadores, que definen el esquema de vecindad. Su característica es la de realizar movimientos en una misma ruta. Estos operadores se les denomina como estructuras de mejoramiento posteriores a las búsquedas inter-ruta. Una ventaja de estos operadores es que sus movimientos no producen infactibilidad en los vehículos. Es importante, además, definir que los movimientos intra-ruta de cada operador se realizan con mejoramientos secuenciales, es decir, si un movimiento realizado mejora la solución, se seguirá mejorando en iteraciones posteriores del mismo operador. Los operadores intra-ruta son descritos en la figura 5.4.

Or-opt

Este operador selecciona un cliente que se va a remover de su posición actual, para ser insertado en cualquier otra posición de la ruta. Este movimiento se realiza para cada uno de los nodos. En la figura 5.4 se muestra el movimiento de intercambio de posición de los nodos (6,7).

Or-opt2

Dos nodos adyacentes cambian de posición en la ruta. En la figura 5.4 se muestra el cambio de posición de los nodos (6,7) donde se ubican en las dos primeras posiciones de la ruta.

Or-opt3

Tres nodos adyacentes son ubicados en otra posición de la ruta. En el ejemplo de la figura 5.4 se cambia de posición a los nodos (2,3,5), localizándolos al final de la ruta.

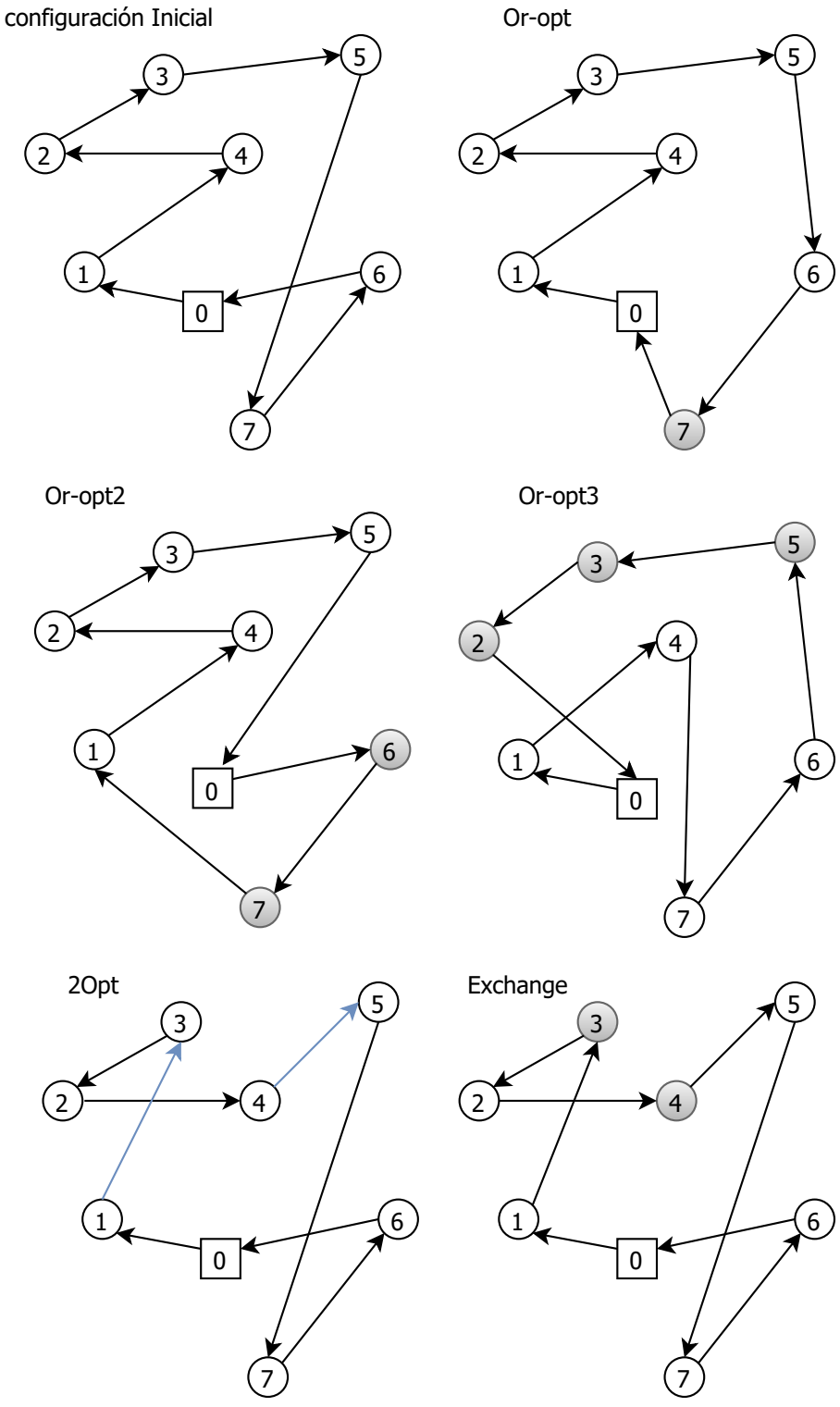

Figura 5.4: Movimientos intra-ruta

2-Opt

Este operador ejecuta un movimiento que consiste en remover dos arcos no adyacentes e ingresarlos nuevamente de forma cruzada. En la figura 5.4 se muestra la eliminación de los arcos que conectan los nodos (1,4) y (3,5) y la introducción de los arcos (1,3) y (4,5). Se observa que este movimiento consiste en eliminar un par de arcos de la forma$(x_1, x_2)(y_1, y_2)$ y construir unos nuevos con la combinación $(x_1, y_1)(x_2, y_2)$.

Exchange

Este operador realiza un movimiento de permutación de posición entre dos nodos. En la figura 5.4 los nodos (3,4) que pertenecen a la ruta $[1, 4, 2, 3, 5, 7, 6]$, son intercambiados y de esta forma conforman la nueva ruta $[1, 3, 2, 4, 5, 7, 6]$.

5.5. Perturbación

La perturbación es el mecanismo que permite la exploración en diversos espacios de la solución y consiste en aplicar movimientos intra-ruta de manera aleatoria. Estos movimientos permiten salir de óptimos locales y alcanzar de esta forma otros puntos del espacio de solución. La selección de los operadores de perturbación se realiza de manera aleatoria. Para esto son definidas dos clases de estructuras así:

Multiple-Swap(1,1)

Como se mostró con el operador $Swap(1, 1)$, consiste en el intercambio de nodos entre dos rutas. En la estructura $MultipleSwap(1, 1)$ se aplica el mismo movimiento, pero sin la necesidad de mejorar la función objetivo. La construcción de una solución a partir de una perturbación son obtenidos usando un vecindario, los cuales son seleccionados de manera aleatoria. El grado de la perturbación es controlada por el vector P, que es definido en [17] y se describe como:

$$P = [0{,}5v, 0{,}6v, 0{,}7v, 0{,}8v, 0{,}91v, 1{,}1v, 1{,}2v, 1{,}3v, 1{,}4v, 1{,}5v, 1{,}6] \tag{5.1}$$

Donde v es el número total de vehículos que tiene la solución. Es importante tener en cuenta que la limitación de esta perturbación establece que la transferencia de nodos entre rutas debe ser de forma tal que las dos rutas sigan siendo factibles. Después de aplicado el procedimiento de perturbación, se efectura un proceso de intensificación, que es efectuado con base en el RVNS.

Multiple-Shift(1,1)

Al igual que el operador de perturbación $Multiple - Swap(1,1)$, se realizan movimientos aleatorios de clientes entre rutas elegidas de manera aleatoria. Como en el caso anterior, la cantidad de movimientos están definidos por el vector P. La diferencia radica en que no es obligatorio que los movimientos sean estrictamente factibles en ambas rutas. Con esto el operador de perturbación se hace más robusto que el $MultipleSwap(1,1)$, y de esta forma, se efectúan una mayor cantidad de movimientos.

5.5.1. Criterio de Distancia

Este criterio implica la creación de un parámetro de referencia que limita los intercambios entre rutas. Consiste en generar una distancia desde un punto de referencia de una ruta y un nodo candidato a transferirse o intercambiarse. Para ello, se determina el centroide de cada ruta. La distancia de referencia será calculada entre el centroide de la ruta y el depósito. Se define que en el centroide de cada ruta existirá un radio que dibuja una circunferencia, los nodos de otras rutas que se encuentren al interior de esta circunferencia son denominados perturbables a la ruta, es decir, se podrán transferir o intercambiar. El Centroide se calcula como la distancia promedio de cada punto de la ruta al depósito. En la figura 5.5, se muestran un ejemplo con el centroide y la circunferencia que delimita los nodos que pueden incorporarse a la ruta. El radio de la circunferencias es variable al ser multiplicado por *FD (factor de distancia)*, que permite controlar el vecindario de clientes que pueden transferirse.

Inicialmente son seleccionados los operadores de perturbación y para cada uno de estos, las rutas que serán perturbadas. De las rutas son identificados los nodos a transferir. El número de configuraciones vecinas son restringidas, esto con el fin de reducir la complejidad computacional del problema. No obstante, a pesar de que se seleccionen nodos que son perturbables, no significa que vayan a ser transferidos, ya que es necesario que su movimiento sea factible en términos de la carga disponible de los vehículos.

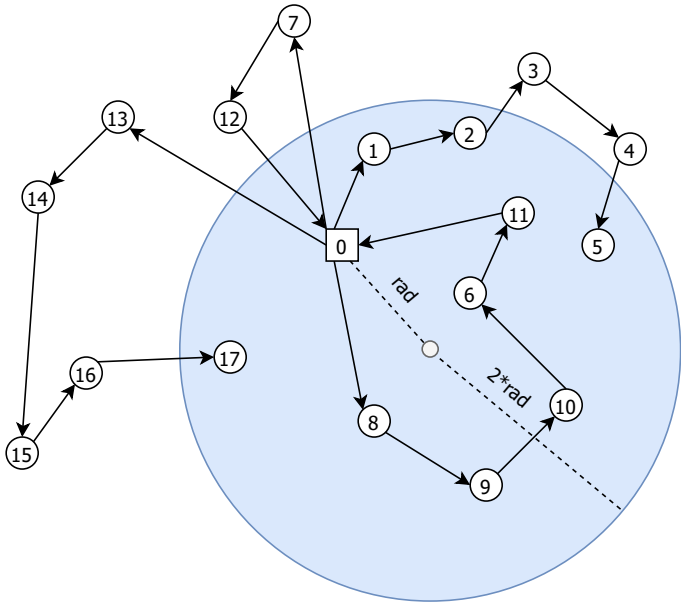

Figura 5.5: Criterio de distancia

Capítulo 6

Análisis de resultados

La metodología planteada fue descrita en los capítulos 4 y 5, los resultados obtenidos serán comparados con los presentados en [20]. Las variables de comparación son el tiempo de computo y calidad de las respuestas.

El algoritmo de optimización implementado resuelve un problema de gran complejidad matemática. Para su solución se requiere de una plataforma que efectúe operaciones computacionales complejas de manera eficiente y en bajo tiempo computacional. Para lograrlo, el algoritmo será implementado en C++ (Microsoft Visual C++ 2015). Para lograr esta codificación se usaron herramientas de $Matlab^{®}$ y el paquete $Coder^{®}$que permite la creación de código en C++ a partir del construido en $Matlab^{®}$.

El algoritmo fue ejecutado en equipo con sistema operativo Microsoft Windows 10 de 64 bits y un procesador Intel Core i5-6200U CPU @ 2.30Ghz 2.40Ghz.

6.1. Codificación

El problema fundamental de las implementaciones para problemas de ruteo es el almacenamiento de datos en las variables que contienen las rutas. La solución clásica del VRP se construye a partir de vectores que contienen los clientes que componen las rutas, donde algunas rutas tendrán más clientes que otras, es decir, los vectores donde se construyen las rutas son de dimensiones diferentes cada uno. Adicionalmente, es necesario tener en cuenta que su dimensión es dinámica, ya que

cuando se realizan las búsquedas locales pueden ceder o recibir clientes a través de los intercambios inter-ruta. Esto hace que la implementación en lenguajes como C++ sea algo más complicados, al no poder establecer una memoria dinámica, es necesario que se defina el tamaño del vector desde el inicio. Por esta razón, implementaciones directas en C++ requieren la utilización de Listas entrelazadas, que son un tipo de estructura de datos que almacena valores individuales o nodos y un puntero de memoria que permite la conexión con otros nodos.

En la implementación se uso una codificación de la solución con una idea simple: Los vectores que construyen las rutas serán agrupados en una matriz que tendrá dimensiones (p, q) donde p corresponde al número de filas que definen las rutas y q la ubicación de los clientes. El valor de q es fijo, por lo tanto, es necesario determinar su valor de manera tal que las rutas puedan expandirse y contraerse tanto como sea necesario. La dimensión de la matriz se define como:

$$(V + V_s\,,\; n + 8) \tag{6.1}$$

Donde V es la cantidad de vehículos propios y V_s es la cantidad de vehículos subcontratados, de esta manera la matriz tendrá un valor $p = V + V_s$. También se define el valor de $q = n + 8$. De esta forma, el espacio de memoria disponible para las rutas será la cantidad total de clientes del problema adicionando la constante ocho, que es el resultante de los espacios requeridos para insertar y retirar clientes, según los operadores utilizados en el RVNS. El valor mínimo de la dimensión q deberá ser $> n$.

De esta manera, es posible codificar el problema a través de vectores con espacios de memoria que por defecto serán ocupados por ceros y que permiten la expansión y contracción de las rutas, solucionando el problema de la memoria dinámica en C++.

6.2. Parámetros e implementación del algoritmo

La parametrización del algoritmo es importante ya que de ella depende el desempeño del mismo. Los datos utilizados en el análisis de resultados son tomados de la referencia [20], inicialmente son estudiadas instancias con (50) clientes y (7) Vehículos. Los parámetros usados en dicho análisis son:

$$IterILS = \#Nodos + \beta * V \tag{6.2}$$

$$\beta = 20 \tag{6.3}$$

$$FI = 4 \tag{6.4}$$

$$MaxIter = (75 * FI) \tag{6.5}$$

Donde:

$IterILS$: Define el número de iteraciones locales. se tomó como referencia la formulación propuesta en [17].

β : Establece la relación entre la cantidad de iteraciones locales y el número de vehículos del problema.

V : Cantidad de vehículos de la flota propia y subcontratada.

FI : Factor de iteraciones definido en el capítulo 5. Inicialmente se establece a $FI = 1$, en algunos casos es necesario un número mayor de iteraciones que permita alcanzar los resultados obtenidos en [20].

$MaxIter$: Número máximo de iteraciones para el ILS.

Estos parámetros presentan un buen desempeño cuando se aplica a instancias de menos de (40) clientes, obteniéndose los óptimos globales. Con instancias de más de (50) clientes se presenta GAP, que crece a medida que aumenta el número de clientes.

6.2.1. Instancias usadas en el análisis de resultados

Serán usadas instancias de la literatura especializada para el problema clásico del CVRP[1]. El valor de k corresponde al número de vehículos de la flota propia se calcula como: $k = 0{,}8q/Q$, donde q es el valor de la demanda total de los clientes y Q es la capacidad de los vehículos. La flota de vehículos es homogénea y las rutas subcontratadas no tienen restricción en su tope máximo de vehículos.

El resumen con los resultados obtenidos usando el algoritmo ILS y los obtenidos en [20] se muestran en la tabla 6.1. Allí se presenta el valor de la función función objetivo promedio de las 10 corridas, el mejor valor alcanzado de la función objetivo y el tiempo requerido para este, así como el tiempo promedio para las 10 corridas de las instancias estudiadas. También se presenta el valor del BKS (*Best known solution*) y el tiempo de cálculo requerido para las instancias presentadas en [20].

En el cuadro 6.2, se presentan las rutas, así como los vehículos requeridos por la flota propia y la subcontratada de las instancias presentadas en el cuadro 6.1.

[1]Instancias obtenidas del sitio: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

	ILS Propuesto				Modelo Exacto [20]			
Instancia	Promedio Función Objetivo en 10 Corridas (PFO)	Mejor solución alcanzada (MSA)	Menor tiempo requerido (s)	TPMS	BKS	Tiempo (s)	Gap^1	Gap^2
P-n16-k8	450	450	1	1	450	4	0,0 %	0,0 %
P-n19-k2	254	254	3	16	254	57	0,0 %	0,0 %
P-n20-k2	266	266	4	18	266	167	0,0 %	0,0 %
P-n21-k2	267	267	1	4	267	404	0,0 %	0,0 %
P-n22-k2	271	271	1	6	271	312	0,0 %	0,0 %
P-n22-k8	589	589	3	15	589	2418	0,0 %	0,0 %
P-n23-k8	532	532	1	58	532	8000	0,0 %	0,0 %
P-n40-k5	480	478	106	604	478	55165	0,0 %	0,3 %
P-n45-k5	541	540	41	820	540	66890	0,0 %	0,2 %
P-n50-k7	582	578	70	1027	578	140375	0,0 %	0,7 %
A-n32-k5	804	804	8	98	804	30500	0,0 %	0,0 %
A-n33-k5	686	686	2	42	686	40750	0,0 %	0,0 %
A-n39-k6	840	837	70	546	837	53500	0,0 %	0,3 %

TPMS: Tiempo promedio para alcanzar mejor solución
BKS: Mejor solución conocida (*Best known solution*)
MSA: Mejor solución alcanzada por el algoritmo propuesto
PFO: Promedio función objetivo de 10 corridas

$Gap^1 = \frac{MSA-BKS}{BKS}$

$Gap^2 = \frac{PFO-BKS}{BKS}$

Cuadro 6.1: Resultados instancias estudiadas

Instancia	k	Rutas Propias	Rutas SubC	Rutas
P-n16_k8	6	5	3	0 14 5 0 0 15 12 10 0 0 1 3 0 0 11 4 0 0 13 9 7 0 *0 6* *0 8* *0 2*
P-n19_k2	1	1	1	0 5 8 16 17 3 12 14 11 4 10 1 0 *0 6 18 2 7 9 15 13*
P-n20_k2	1	1	1	0 1 10 13 8 17 18 3 12 15 11 4 0 *0 6 19 5 14 16 9 7 2*
P-n21_k2	1	1	1	0 16 1 10 8 18 19 3 12 15 11 4 0 *0 6 20 5 14 17 9 13 2 7*
P-n22_k2	1	1	1	0 9 13 8 18 19 3 12 15 11 4 10 1 0 *0 16 6 20 5 14 17 21 7 2*
P-n22_k8	6	6	3	0 9 2 1 6 0 0 10 8 3 4 0 0 17 18 15 0 0 13 11 0 0 14 21 20 0 0 19 0 *0 12* *0 7 5* *0 16*

Cuadro 6.2 continúa en siguiente página...

Instancia	k	Rutas Propias	Rutas SubC	Rutas
P-n23_k8	6	6	2	0 3 19 18 0 0 17 9 13 0 0 11 15 12 10 0 0 22 14 5 0 0 7 4 0 0 1 2 0 *0 21 6 20* *0 16 8*
P-n40_k5	4	4	1	0 18 4 19 13 25 14 0 0 17 37 15 33 39 10 30 34 21 29 16 0 0 32 22 3 36 35 20 2 11 0 0 27 1 28 31 8 26 7 23 24 6 0 *0 12 5 38 9*
P-n45_k5	4	4	1	0 32 1 22 3 36 35 20 29 2 11 0 0 5 37 44 42 40 19 41 13 4 0 0 38 9 16 21 34 30 10 39 33 15 17 0 0 27 8 28 31 26 7 43 24 23 6 0 *0 12 18 14 25*
P-n50_k7	5	5	2	0 2 28 21 47 48 30 0 0 44 3 24 18 25 31 10 12 0 0 26 38 11 14 19 35 7 0 0 45 29 5 36 37 20 15 13 27 0 0 16 49 23 43 41 42 22 1 33 6 0 *0 17 40 32 9 39* *0 4 34 46 8*
A-n32_k5	3	3	2	0 6 2 3 23 4 11 28 14 0 0 29 18 8 9 22 15 10 25 5 20 0 0 26 7 13 17 19 31 21 0 *0 24 27* *0 30 16 12 1*

Cuadro 6.2 continúa en siguiente página...

Instancia	k	Rutas Propias	Rutas SubC	Rutas
A-n33_k5	3	3	2	0 24 6 19 14 21 1 31 11 0 0 20 5 26 7 8 13 32 2 0 0 15 17 9 3 16 29 0 *0 4 12 10 30 25 27* *0 22 23 28 18*
A-n39_k6	4	4	2	0 24 3 38 12 9 28 29 5 0 0 7 8 4 16 10 27 18 0 0 37 31 14 35 25 33 19 2 0 0 20 32 34 22 21 23 17 36 1 6 0 *0 26 11* *0 15 13 30*

Cuadro 6.2: Resultado Rutas

En el cuadro 6.3 se presentan los valores de la función objetivo de las 13 instancias para 10 casos estudiados con cada una de ellas, así como el valor promedio y la mejor solución obtenida. En el cuadro 6.4 se presenta el tiempo de computo requerido en segundos en las 13 instancias para los 10 casos estudiados con cada una de ellas, así como el tiempo promedio y menor tiempo requerido.

Instancia	Caso 1	Caso 2	Caso3	Caso 4	Caso 5	Caso 6	Caso 7	Caso 8	Caso 9	Caso 10	Media	Mínimo
P-n16-k8	450	450	450	450	450	450	450	450	450	450	450	450
P-n19-k2	254	254	254	254	254	254	254	254	254	254	254	254
P-n20-k2	266	266	266	266	266	266	266	266	266	266	266	266
P-n21-k2	267	267	267	267	267	267	267	267	267	267	267	267
P-n22-k2	271	271	271	271	271	271	271	271	271	271	271	271
P-n22-k8	589	589	589	589	589	589	589	589	589	589	589	589
P-n23-k8	532	532	532	532	532	532	532	532	532	532	532	532
P-n40-k5	478	480	478	478	483	480	480	478	480	480	479,5	478
P-n45-k5	540	543	543	540	543	540	540	540	540	543	541,2	540
P-n50-k7	578	578	578	591	578	578	578	578	593	589	581,9	578
A-n32-k5	804	804	804	804	804	804	804	804	804	804	804	804
A-n33-k5	686	686	686	686	686	686	686	686	686	686	686	686
A-n39-k6	842	842	837	837	849	842	837	837	837	837	839,7	837

Cuadro 6.3: Valores de la función objetivo de las 13 instancias estudiadas

Instancia	Caso 1	Caso 2	Caso3	Caso 4	Caso 5	Caso 6	Caso 7	Caso 8	Caso 9	Caso 10	Media	Mínimo
P-n16-k8	1,8	1,1	0,8	0,9	1,2	1,1	1,4	1,8	1,2	1,4	1,3	0,8
P-n19-k2	15,5	20,1	4,5	17,1	9,3	27,4	8,9	6,5	44,1	2,8	15,6	2,8
P-n20-k2	14,6	36,8	15,7	27,3	28,9	3,9	10,6	20,1	18,7	8,3	18,5	3,9
P-n21-k2	0,9	0,7	0,9	3,3	4,8	8,0	9,6	1,3	1,5	9,4	4,0	0,7
P-n22-k2	2,7	7,3	7,2	6,7	1,1	8,2	3,1	7,1	14,5	6,5	6,4	1,1
P-n22-k8	4,9	2,8	12,7	32,0	5,6	41,0	4,2	19,1	15,5	9,9	14,8	2,8
P-n23-k8	0,8	72,6	9,5	16,1	126,9	167,4	2,3	90,2	56,7	38,7	58,1	0,8
P-n40-k5	247,0	986,6	343,2	258,0	921,8	107,6	929,6	220,8	922,7	1100,5	603,8	107,6
P-n45-k5	41,5	1664,5	1522,3	120,2	1600,5	285,3	258,4	173,1	956,7	1574,9	819,7	41,5
P-n50-k7	605,7	443,7	69,6	2609,4	569,2	161,0	524,3	126,6	2579,1	2585,8	1027,4	69,6
A-n32-k5	70,8	14,6	450,8	27,2	56,6	72,0	12,0	204,9	7,9	59,8	97,6	7,9
A-n33-k5	42,3	31,1	96,2	26,9	2,4	49,0	5,3	32,2	7,6	122,8	41,6	2,4
A-n39-k6	796,2	772,9	842,0	70,0	952,9	769,6	598,0	229,9	69,8	355,1	545,7	69,8

Cuadro 6.4: Tiempo en segundos de las 13 instancias estudiadas

En el cuadro 6.4, se observa que los tiempos presentan variaciones significativas, debido a la complejidad matemática de las instancias estudiadas, lo que se refleja en altos requerimientos de tiempo de computo. Los espacios de solución de algunas instancias están constituidos por gran cantidad de óptimos locales, incrementando de manera sustancial el tiempo computacional para ser estudiado. Una forma de mejorar este tiempo, es el uso de adecuadas heurísticas de inicio, de manera que el proceso este próximo a soluciones de alta calidad.

La utilización de factores de iteraciones, perturbación y de distancia, permiten establecer una adecuada intensificación y diversificación. El valor de los factores se calibra con base en el desempeño del algoritmo. Para esto las instancias son estudiadas en un rango de valores.

Esta implementación demuestra que es posible realizar aplicaciones para soluciones de problemas relacionados al VRP en *Matlab* y posteriormente convertirlas a *C++* mediante la aplicación *Coder*. Es importante tener en cuenta que la implementación debe utilizar en todos los casos vectores y matrices que se definan con dimensiones fijas, para garantizar compatibilidad para la transformación del código. Esto permite realizar la implementación del algoritmo con tiempos de codificación cortos y que permite estudiar instancias de gran complejidad matemática.

Se demostró que el algoritmo ILS presenta un buen rendimiento de acuerdo a los resultados mostrados en la tabla 6.1, donde para las instancias estudiadas, se obtuvieron resultados con un GAP de cero en tiempos reducidos en comparación a los presentados en [20]. Por lo tanto, se verifica que esta metodología es una buena forma de encontrar soluciones para instancias de baja y mediana complejidad matemática, y una alternativa de interés para el análisis de instancias de gran complejidad matemática.

6.2.2. Análisis de resultados para instancias de mediana y gran complejidad matemática

Instancias superiores a los (50) clientes son considerados de gran complejidad matemática, por tratarse de un problema con arco doble, uno para la flota propia y otro para la flota subcontratada, aumentando la complejidad matemática con respecto al problema que los considera de manera independiente. Las instancias estudiadas no presentan respuesta en la literatura. Por su complejidad matemática el tiempo computacional es alto, el estudio contemplo (5) corridas para cada instancia. Los resultados se muestran en los cuadros del 6.5 al 6.8. En los casos estudiados se asume un valor para iteraciones en $FI = 2$. En el cuadro 6.5 se presentan los resultados de 10 instancias que fueron estudiadas. En cada una de estas se presenta el promedio de 5 casos analizados, así como la mejor solución obtenida. en el cuadro tmabién se presenta el tiempo promedio de ejecución. En el cuadro 6.6 se presentan los valores de la función objetivo para las (5) corridas, el valor

ILS Propuesto				
Instancia	Promedio Función Objetivo en 5 Corridas	Mejor solución encontrada	Menor tiempo requerido (s)	Tiempo promedio ejecución (s)
P-n50-k10	723	717	143	504
P-n55-k7	606	604	291	642
P-n55-k15	966	964	317	386
P-n60-k15	1004	990	374	652
P-n65-k10	848	844	664	1069
P-n70-k10	870	861	1493	2077
P-n76-k5	767	748	1469	2494
P-n101-k4	873	853	806	9475
P-n151-k12	1140	1130	13094	25514
M-n200-k16	1480	1428	50226	82217

Cuadro 6.5: Resultados instancias de gran complejidad matemática

medio y el mejor valor obtenido. En el cuadro 6.7 se presenta el tiempo requerido en cada uno de los casos estudiados, así como el tiempo promedio y el menor tiempo requerido. En el cuadro 6.8 se presentan las rutas de la mejor solución obtenida, así como el número de rutas propias y subcontratadas.

Instancia	Caso 1	Caso 2	Caso3	Caso 4	Caso 5	Media	Mínimo
P-n50-k10	727	722	717	726	721	723	717
P-n55-k7	604	604	608	602	610	606	602
P-n55-k15	964	966	965	968	965	966	964
P-n60-k15	1013	1004	990	1007	1008	1004	990
P-n65-k10	844	852	849	849	846	848	844
P-n70-k10	870	861	879	869	872	870	861
P-n76-k5	769	748	767	772	780	767	748
P-n101-k4	878	853	868	878	889	873	853
P-n151-k12	1134	1166	1137	1130	1135	1140	1130
M-n200-k16	1516	1458	1428	1486	1510	1480	1428

Cuadro 6.6: Resultados de la función objetivo de las instancias estudiadas

Instancia	Caso 1	Caso 2	Caso3	Caso 4	Caso 5	Media	Mínimo
P-n50-k10	998	483	143	576	322	504	143
P-n55-k7	896	291	684	684	744	660	291
P-n55-k15	317	560	317	364	371	386	317
P-n60-k15	1120	374	655	602	511	652	374
P-n65-k10	664	1848	1033	862	939	1069	664
P-n70-k10	2233	1493	2043	2593	2023	2077	1493
P-n76-k5	4292	3547	1491	1671	1469	2494	1469
P-n101-k4	806	5773	17347	20024	3427	9475	806
P-n151-k12	13094	30384	27432	26098	30561	25514	13094
M-n200-k16	50226	101834	75063	112227	71736	82217	50226

Cuadro 6.7: Resultados de los tiempos en segundos de las instancias estudiadas

Caso	k	Rutas Propias	Rutas SubC	Rutas
P-n50-k10	10	8	2	0 16 3 44 32 17 0 0 7 35 14 19 13 27 0 0 39 31 25 18 24 49 0 0 11 38 10 0 0 21 36 47 48 45 0 0 26 12 9 40 0 0 30 5 37 20 15 29 0 0 23 41 42 22 28 2 0 *0 4 34 46 8* *0 6 33 1 43*
P-n55-k7	7	5	2	0 51 16 49 24 23 43 41 42 22 1 33 0 0 45 29 5 36 37 20 15 13 27 52 0 0 26 38 10 31 25 9 39 12 0 0 30 48 47 21 28 2 6 0 0 7 35 53 11 14 19 54 8 0 *0 4 34 46* *0 17 40 3 44 32 50 18*

Cuadro 6.8 continua en siguiente página...

Caso	k	Rutas Propias	Rutas SubC	Rutas
P-n55-k15	15	12	4	0 50 18 24 49 0 0 5 37 20 15 0 0 25 31 10 0 0 6 33 16 0 0 23 41 42 43 1 0 0 39 9 26 0 0 27 13 54 52 0 0 53 14 19 0 0 11 38 0 0 7 35 8 46 0 0 32 44 3 51 0 0 12 40 17 0 *0 30 48 21* *0 4 34* *0 45 29 47 36* *0 2 28 22*

Cuadro 6.8 continua en siguiente página...

Caso	k	Rutas Propias	Rutas SubC	Rutas
P-n60-k15	15	11	4	0 1 43 41 42 22 0 0 25 55 18 24 49 51 0 0 57 15 20 37 5 0 0 6 17 12 26 0 0 14 59 53 0 0 52 27 13 54 19 0 0 21 36 47 48 0 0 30 28 2 0 0 7 11 38 0 0 16 23 56 33 0 0 3 44 50 32 0 *0 34 46 8 35* *0 40 9 39* *0 58 10 31* *0 4 45 29*
P-n65-k10	7	7	3	0 48 47 36 60 20 37 5 0 0 2 28 61 21 30 0 0 49 24 18 50 25 55 31 39 0 0 17 16 63 23 56 41 22 62 0 0 7 14 59 11 53 0 0 40 9 32 44 3 51 0 0 8 35 19 54 13 57 15 29 45 0 *0 26 12 58 10 38* *0 4 34 46 52 27* *0 6 33 1 43 42 64*

Cuadro 6.8 continua en siguiente página...

Caso	k	Rutas Propias	Rutas SubC	Rutas
P-n70-k10	10	8	2	0 68 2 28 61 21 30 0 0 51 44 32 9 39 40 0 0 6 33 63 23 56 49 16 17 0 0 62 22 64 42 41 43 1 0 0 4 29 15 57 13 54 19 35 8 0 0 3 24 18 50 25 55 31 12 0 0 48 47 36 69 60 20 37 5 0 0 14 59 65 38 10 58 0 *0 67 34 46 52 27 45* *0 26 7 53 11 66*
P-n76-k5	3	3	2	0 68 30 74 21 47 36 69 71 60 70 20 37 15 57 13 54 19 35 7 26 0 0 17 51 33 73 1 43 41 42 64 22 61 28 62 2 6 0 0 3 44 32 9 39 72 31 55 25 50 18 24 49 56 23 63 16 0 *0 75 4 67 34 46 8 52 27 45 29 48 5* *0 40 12 58 10 38 65 66 11 53 14 59*
P-n101-k4	2	2	2	0 28 26 12 76 50 1 30 20 51 9 71 66 65 35 34 78 29 24 55 25 67 23 41 2 57 15 43 42 87 13 0 0 27 69 70 31 88 7 82 48 19 11 62 10 32 90 63 64 49 36 47 46 8 45 17 84 5 60 83 18 52 0 *0 89 6 94 95 97 92 59 96 99 93 98 37 100 91 85 61 16 86 44 14 38* *0 53 58 40 21 73 72 74 22 75 56 39 4 54 80 68 77 3 79 81 33*

Cuadro 6.8 continua en siguiente página...

Caso	k	Rutas Propias	Rutas SubC	Rutas
P-n151-k12	12	9	3	0 147 118 60 83 114 125 45 5 99 104 96 0 0 6 84 17 113 61 16 141 86 140 38 14 42 137 0 0 89 18 8 46 124 47 36 143 49 64 63 126 90 108 10 70 101 0 0 138 130 55 25 139 39 67 23 56 4 110 105 0 0 27 127 31 32 131 128 66 20 30 122 1 69 132 0 0 146 88 148 62 11 107 19 123 48 82 7 106 52 0 0 111 50 102 51 103 71 65 136 35 135 34 78 129 3 0 0 53 58 40 21 73 115 2 57 15 43 142 119 44 91 100 37 98 92 97 117 0 0 12 80 150 68 121 29 24 134 54 109 149 26 0 *0 112 94 95 59 93 85* *0 13 87 144 145 41 22 133 75 74 72* *0 28 76 116 77 79 33 81 120 9*

Cuadro 6.8 continua en siguiente página...

Caso	k	Rutas Propias	Rutas SubC	Rutas
M-n200-k16	13	13	4	0 146 52 153 48 107 175 11 62 182 31 0 0 83 199 125 45 17 113 86 173 84 5 104 99 6 0 0 117 97 37 193 91 192 119 14 142 42 172 144 178 115 2 53 0 0 105 198 197 56 186 23 67 170 25 55 165 195 0 0 132 69 122 30 20 188 66 128 160 131 32 70 162 0 0 101 90 126 63 181 64 49 143 36 46 174 8 114 18 89 0 0 166 60 118 82 124 47 168 19 123 7 194 106 0 0 40 110 155 4 139 39 187 130 54 179 149 26 0 0 58 145 41 22 133 75 74 171 72 73 21 180 0 0 61 16 141 191 44 140 38 43 15 57 87 137 152 0 0 176 1 51 9 103 161 71 65 136 35 135 34 78 185 0 0 50 102 157 79 169 129 3 158 77 76 111 0 0 12 109 177 80 150 134 163 24 29 121 68 116 184 0 *0 156 112 147 183 94* *0 27 167 127 190 88 148 159 189 10 108* *0 13 95 96 59 151 92 98 100 85 93* *0 28 154 138 196 33 81 120 164*

Cuadro 6.8: Resultado rutas instancias estudiadas

Capítulo 7

Conclusiones

Se planetó un modelo matemático para la solución del problema de ruteo de vehículos con rutas propias y subcontratadas VRPPC, resuelto usando la metaheurística ILS, que usa conceptos de intensificación y diversificación, los cuales son simulados a través del algoritmo RVNS para efectuar la búsqueda local y las perturbaciones. Se usó un adecuado esquema de vecindad, donde se aplican operadores inter-ruta e intra-ruta. Para conformar la solución inicial se empleó la heurística de ahorros modificada.

Para la validación se usaron instancias de la literatura especializada. Los resultados fueron comparados en calidad y tiempo de cómputo, obteniendo soluciones de gran calidad, con GAP de 0 % en tiempos de cómputo reducidos en comparación a los presentados en [20].

La metaheurística ILS ese muestra como una herramienta eficiente para la solución del VRPPC. La capacidad para encontrar buenas soluciones fue probada a través de instancias de gran complejidad matemática, obteniendo resultados de calidad en razonables tiempos de cómputo. Este estudio permitió definir nuevos BKS para el VRPPC, en instancias de mediana y gran complejidad matemática de 50 a 200 clientes.

Trabajos Futuros

Estudiar el modelo multiobjetivo para el VRPPC que consideren tres aspectos: costos operativos, impacto ambiental medido como la cantidad de partículas emitidas al medio ambiente, y el impacto

social que establece cargas laborales similares para los conductores, tomando como medida la distancia recorrida por cada vehículo.

Emplear procesamiento paralelo, aprovechando las características propias del problema y del método de solución. El procesamiento paralelo tiene como característica explorar un área mayor del espacio de soluciones en menor tiempo de cómputo respecto al procesamiento secuencial, y que se traduce en mejoramiento en el desempeño, especialmente en instancias de gran complejidad matemática, y con bajos tiempos de cómputo.

Bibliografía

[1] Michael O Ball, BL Golden, AA Assad, and LD Bodin. Planning for truck fleet size in the presence of a common-carrier option. *Decision Sciences*, 14(1):103–120, 1983.

[2] Marie Claude Bolduc, Jacques Renaud, and Fayez Boctor. A heuristic for the routing and carrier selection problem. *European Journal of Operational Research*, 183(2):926–932, 2007. ISSN 03772217. doi: 10.1016/j.ejor.2006.10.013.

[3] Marie-Claude Bolduc, Jacques Renaud, Fayez Boctor, and Gilbert Laporte. A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. *Journal of the Operational Research Society*, 59(6):776–787, 2008.

[4] Ching Wu Chu. A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, 165(3):657–667, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2003.08.067.

[5] 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.

[6] Jean-Francois Cordeau and Québec) Groupe d'études et de recherche en analyse des décisions (Montréal. *The VRP with time windows*. Montréal: Groupe d'études et de recherche en analyse des décisions, 2000.

[7] G. B. Dantzing and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6 (1):80–91, 1959. ISSN 00251909, 15265501. doi: 10.1287/mnsc.6.1.80.

[8] Moshe Dror and Pierre Trudeau. Split delivery routing. *Naval Research Logistics (NRL)*, 37 (3):383–402, 1990.

[9] Jalel Euchi, Habib Chabchoub, and Adnan Yassine. New evolutionary algorithm based on 2-opt local search to solve the vehicle routing problem with private fleet and common carrier. 2013.

[10] Michel Gendreau, Gilbert Laporte, and René Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88(1):3–12, 1996.

[11] Irina Gribkovskaia and Gilbert Laporte. One-to-many-to-one single vehicle pickup and delivery problems. In *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, January 2008. ISBN 978-0-387-77777-1. doi: 10.1007/978-0-387-77778-8_16. URL http://dx.doi.org/10.1007/978-0-387-77778-8_16.

[12] Kuancheng Huang and Cheng-Po Hsu. A Lagrangian Heuristic for the Vehicle Routing Problems with the Private Fleet and the Common Carrier. *Journal of the Eastern Asia Society for Transportation Studies*, 9:644–659, 2011.

[13] John G Klincewicz, Hanan Luss, and Martha G Pilcher. Fleet size planning when outside carrier services are available. *Transportation Science*, 24(3):169–182, 1990.

[14] IlKyeong Moon, Jeong-Hun Lee, and June Seong. Vehicle routing problem with time windows considering overtime and outsourcing vehicles. *Expert Systems with Applications*, 39 (18):13202–13213, 2012.

[15] Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41(4):421–451, 1993.

[16] J-Y Potvin and M-a Naud. Tabu search with ejection chains for the vehicle routing problem with private fleet and common carrier. *Journal of the Operational Research Society*, 62(2): 326–336, 2011. ISSN 0160-5682. doi: 10.1057/jors.2010.102.

[17] Anand Subramanian. *UNIVERSIDADE FEDERAL FLUMINENSE Luiz Satoru Ochi Eduardo Uchoa*. PhD thesis, UNIVERSIDADE FEDERAL FLUMINENSE, 2012.

[18] Goos Kant & RenÃ© Peeters Sybren Huijink. An adaptable variable neighborhood searh for the vehile routing problem with order outsourng. *CentER Discussion Paper Series No. 2014-062*, 2014. ISSN 9780983648703. doi: 10.3386/w19846.

[19] Frank A Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3(3):192–204, 1969.

[20] Eliana Mirledy Toro-Ocampo, John Fredy Franco-Baquero, and Ramón Alfonso Gallego-Rendón. Modelo matemático para resolver el problema de localización y ruteo con restricciones de capacidad considerando flota propia y subcontratada. *Ingeniería, Investigación y Tecnología*, 17(3):357–369, 2016.