UNIVERSIDAD CATOLICA
DE LA SANTISIMA CONCEPCION

Extremal optimisation aplicado al problema de ruteo de vehículos con límites de capacidad

Josías Sanhueza Gatica

Informe de Proyecto de Título para optar al Título de
Ingeniero Civil Informático

Profesor Guía
Pedro Gómez Meneses

Departamento de Ingeniería Informática
Facultad de Ingeniería
Concepción, Marzo de 2016

Resumen

En la actualidad, el sector industrial y comercial a lo largo de todo el mundo tiene que lidiar con diversos obstáculos para ser productivo. Uno de estos problemas tiene relación con el proceso de repartir pedidos de un proveedor a sus clientes. Aquí se trata de encontrar la mejor ruta posible para así no generar gastos innecesarios con el fin de, sin violar el límite de capacidad del repartidor, optimizar variables que pueden ser: distancia total recorrida o costo total de transporte, entre otros.

Un ejemplo de este problema es el siguiente. Se tiene un distribuidor que es el único depósito que cuenta con *"k"* camiones repartidores con su respectivo límite de capacidad. A su vez existen *"n"* clientes con distinta cantidad de demanda en donde el distribuidor debe satisfacer las necesidades de cada cliente con sus camiones repartidores a través de rutas que empiezan y terminan en el depósito. Lo que se busca es optimizar la distancia total recorrida, por lo que se requiere encontrar la ruta más corta para cada camión repartidor.

Con lo anterior notamos que el problema de ruteo de vehículos se trata de encontrar la ruta óptima según el o los criterios de desempeño deseados, donde su dificultad está dentro de la categoría de NP-completo. Dada su gran aplicabilidad a problemas de logística, genera variantes con distintos requerimientos hasta el punto de que aún es materia de investigación. Este es un problema de optimización combinatoria el cual es una rama de la optimización en matemáticas y en ciencias de la computación. Dentro de esta última, se encuentra el campo de la Inteligencia Artificial el cual se puede utilizar para resolver problemas que son complejos.

Para el problema de ruteo, existen diversos algoritmos que dan soluciones aproxi-

madas, algunos de ellos son: Colonia de Hormigas, Algoritmos Genéticos, Búsqueda Tabú, entre otros. Si bien es cierto que el problema ya tiene soluciones con diferentes procedimientos, se aplicará un novedoso algoritmo llamado Extremal Optimisation para estudiar su comportamiento en este tipo de problema.

Extremal Optimisation es una técnica bastante reciente ya que fue propuesta por Boettcher y Percus en 1990. Existen trabajos de investigación que han dado buenos resultados a problemas complejos (Boettcher and Percus, 1999). Apoyado en lo anterior, se realizará la implementación que modelará el problema de ruteo de vehículos con límites de capacidad con la novedosa metaheurística. Una vez que se finalice la implementación, se realizarán experimentos computacionales utilizando un set de pruebas de los cuales obtendremos resultados. Estos resultados los compararemos y los analizaremos con valores de solución encontrados en otras investigaciones lo que nos permitirá obtener concluciones respecto al comportamiento de Extremal Optimisation.

Abstract

In the actuality, the industrial and commecial sector along around the world they have to deal with different obstacles to be productive. One of this problem have relation with the process from distribute orders from a supplier to its customers. Here is try found the best route possible so as not generate expenses unnecessary with the final purpose of, without violating the capacity limit of the dealer, optimize variables that can be: total distance travel or the total cost of transport, between others.

An example of this problem it is follows. It is has an supplier that is the only depot that wich has "k"delivery trucks with with their respective capacity limit. At the same time exist "nçustomers with different quantity demand where the supplier must meet the needs of each customer with their delivery trucks through routes that begin and end in the depot. What is sought it optimize total distance travel, so it is required to find the shortest route for each delivery truck.

With the above we note that the vehicle routing problem it is to find the optimal route or i is according to the Desired performance criteria, where the difficulty is NP-completo. Given its wide applicability to logistical problems, it generates variants with different requirements to the until the point that is still matter investigation. This is a problem of combinatorial optimization which it is a branch of the in metematics optimization and computer's science. In side the last, it is the field of artificial intelligence which can be used to solve problems that are complex.

For the routing problem, exist different algorithms that given apriximate solutions, some of them are: Ant Colony, Genetic Algoritms, Tabu Search, between others. It is true that the problem has already solutions with different procedures, shall apply a novel

algorithm called Extremal Optimisation to study their behavior in this type of problem.

Extremal Optimisation is a tecnic fairly recent as it was proposed by Boettcher and Percus in 1990. Exist research work that have produced good results to complex problems (Boettcher and Percus, 1999). Leaning against the above, implementing that model the routing problem with capacity limits with the novel it metaheuristic be held. Once the implementation is completed, computational experiments will be conducted using a set of tests which get results. These results we compare and analyze with values found in other research solution allowing us to draw conclusions regarding the behavior of Extremal Optimisation.

Agradecimientos

En primer lugar quiero agradecer a Dios, fuente de vida, sabiduría y ciencia. Por darme la oportunidad de finalizar una profesión en donde pude ver la ayuda y fidelidad del único Dios verdadero.

A mi familia; Rosa mi madre, Milton mi padre y Caroline mi hermana. Por brindarme apoyo y confianza incondicional en momentos tristes y alegres de mi vida. A aquellas personas especiales se añade otra, mi esposa Blanca Bascur. Quien también su apoyo y confianza fué incondicional al momento de programar y redactar mi tésis.

A mi profesor guía Dr. Pedro Gómez Meneses, quien pacientemente atendió y resolvió todas mis dudas de forma clara, presencial o electrónicamente. También por darme el ánimo y aliento después de exponer en anteproyecto frente a profesores y compañeros de carrera.

*Al profesor Roberto Asín, quien nos brindó financiemiento del proyecto Fondecyt N°*11121220*:"Distributed sat and maxsat for combinatorial optimization" para esta investigación y facilitarnos servidores de la universidad para realizar días de ejecución de programas implementados.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Presentación del problema

El problema que se abordará en esta investigación es el *Problema de ruteo de vehículos con límites de capacidad (CVRP, del inglés Capacitated Vehicle Routing Problem)*. Al cual se buscará una solución aproximada con una metaheurística llamada *Extremal Optimisation (EO)*. Brevemente, CVRP es un problema de optimización combinatoria de dificultad NP-completo que trata de k vehículos con capacidad limitada que satisfacen los requerimientos de m clientes a través de rutas que empiezan y terminan en un único depósito.

En este capítulo se verá la justificación, delimitación, objetivo general, objetivos específico y metodologías para cada objetivo especifico. Lo anterior, culminará en la solución aproximada del problema en investigación para finalmente concluir respecto a esta novedosa metaheurística.

1.2. Justificación del problema

La justificación del problema CVRP para esta investigación, es la siguiente. Existe un conjunto de variantes de problemas de ruteo de vehículos (VRP, del inglés Vehicle Routing Problem), de los cuales CVRP es el más sencillo, pero aún así complejo, dada su complejidad computacional NP-completo. Se quiere implementar el problema y analizar el comportamiento de EO comparándolo con otras metaheurística ya imple-

mentado para este problema. Con los resultados obtenidos se podrá concluir las ventajas y desventajas de este metaheurística.

La justificación de la elección de la metodología EO radica en que es emergente y se ha probado en otros problemas y se quiere ver su comportamiento en nuevo problema diferente a los ya estudiados como: vendedor viajero, bi-partitioning problem, entre otros donde a mostrando ser competitivo en sus resultados. A lo anterior se añade que, EO es un método simple de implementar y no requiere de estructura compleja que produce un tiempo de cálculo mayor.

1.3. Delimitación del problema

Las delimitaciones para esta investigación son las siguientes:

- El tiempo, debido a que solo tendremos un semestre para realizar la investigación.
- El set de pruebas se obtendrá del autor Dorronsoro Díaz (2006) en sitio web http://www.bernabe.dorronsoro.es/vrp/.
- Recursos computacionales, contamos con servidores de Universidad Católica de la Santísima Concepción (UCSC), marca DELL Poweredge R815 de 4 procesadores AMD Opteron(tm) Processor 6278, corriendo a 2400 Mhz. Y cuenta con 192 Gbytes en memoria RAM.
- Compilador gcc (Ubuntu/Linaro 4.8.1-10ubuntu9).
- La cantidad de test serán de 30 por problema.

1.4. Objetivo general

El objetivo general de esta investigación es Aplicar Extremal Optimisation al problema de ruteo de vehículos con límites de capacidad.

1.5. Objetivos específicos

En esta investigación se tienen los siguientes objetivos especificos:

- Estudiar el problema de ruteo de vehículos con límites de capacidad y Extremal Optimisation.
- Modelar el problema a través de matemática y algorítmicamente.
- Implementar EO al problema propuesto.
- Aplicar EO a los benchmark seleccionados.
- Analizar los resultados obtenidos.

1.6. Metodología

La metodología que se usará en cada objetivo específico se describe de la siguiente manera:

Estudiar CVRP y EO.
Para realizar esta investigación se requiere de un estudio del problema y metaheurística en cuestión y así lograr entender el área de estudio. Buscaremos información en investigaciones publicadas proporcionadas por Google Académico, las bases de datos de UCSC y otras fuentes de información (Google, entre otros).

Modelar CVRP.
Para modelar CVRP se realizará siguiendo los dos puntos siguientes:

- Primero: Modelamiento matemático según: programación matemática y lineal. Programación matemática consiste en maximizar o minimizar una función real con respectivas restricciones. A su vez, programación lineal es una rama de programación matemática en donde se ocupan variables y función lineal.
- El segundo punto es el modelamiento algorítmico y para ello nos basaremos en la descripción de EO mediante su pseudocódigo y se modelará el ecosistema del problema con la utilización de estructura de datos de arreglos.

.

Implementar EO a CVRP.

En esta investigación se creará un programa que solucione aproximadamente CVRP, esto quiere decir que se desarrollará la creación de un programa en dos etapas. La primera, es implementar EO básico, el cual trata de explorar en el espacio de búsqueda la solución. Y la segunda es agregar a EO básico diferentes heurísticas de búsqueda local para así tener exploración y explotación en el espacio de búsqueda. En cada etapa se documentará la implementación, dando lugar al uso de un modelo de desarrollo de software prototipo incremental. Para beneficiar al tiempo de ejecución de esta metaheurística se optará por uno de los lenguajes compilados, estructurada el cual será C.

Aplicar EO a los benchmark.

Una vez que se tenga la implementación de EO en sus dos etapas, serán aplicados a un benchmarck que será seleccionado en uno de los capítulos mas adelante. Cada set de prueba será efectuado 30 veces y sus resultados se tabularán posteriormente para analizarlos.

Analizar los resultados.

Cuando ya se tengan los resultados tabulados obtenidos al aplicar EO a los diferentes benchmark, se analizarán los datos. Este análisis será mediante el cálculo del $\%gap$. Así se compararán en cuanto a la calidad del resulado. El $\%gap$ es la distancia o diferencia excesiva que existe entre elementos relacionados entre sí.

1.7. Esquema de la tesis

La investigación se documentará en 6 capítulos, siendo la Introducción el primero de ellos. En este capítulo se realiza la presentación del problema abordado, la justificación de la elección del mismo, sus respectivas delimitaciones, los objetivos general y específicos y la metodología que se usará para cada objetivo específico.

El Capítulo 2 corresponde al marco teórico, en donde se definirán conceptos importantes que se deben tener en cuenta para entender la investigación, se definirá qué son los problemas de optimización combinatoria, problema de empaquetamiento en compar-

timientos, problema del vendedor viajero y problema de ruteo de vehículos con límites de capacidad. También se definirá la metaheurística que se ocupará en esta investigación, Extremal Optimisation y sus diferencias con Algoritmos Genéticos. Un punto importante es que CVRP conjuga dos problemas NP-completo en sí, el problema de empaquetamiento (BPP, del inglés Bin Packing Problem) y el problema del vendedor viajero (TSP, del inglés Travel Salesman Problem).

El Capítulo 3 que corresponde al estado del arte, se presentarán aplicaciones e investigaciones de EO que tienen relación con el tema de esta investigación respecto a problemas de ruteo y también se describirán brevemente distintas técnicas que solucionan CVRP.

En el Capítulo 4 de implementación, se realizará la modelación del problema CVRP para aplicar EO, ya que esta trabaja con un vector solución. Se presentará la fórmula para calcular el fitness de cada individuo y también el algoritmo de búsqueda local que se utilizará con el objetivo de mejorar las soluciones. Además, se procederá a la aplicación de EO sobre CVRP y explicar su respectivo procedimiento.

El Capítulo 5 de Experimentos computacionales, se presentará una lista de benchmark con el fin de elegir uno de ellos el cual será ejecutado 30 veces para presentar los resultados en tablas y gráficos de caja y vigotes que culminarán en análisis posterior de los resultados.

Finalmente, se tiene el Capítulo 6 el cual aborda las conclusiones, contribuciones y trabajo futuro de este proyecto de título.

Capítulo 2

Marco teórico

2.1. Problemas de optimización combinatoria

La optimización combinatoria es una rama de la optimización en matemáticas y en ciencias de la computación, relacionada a la investigación de operaciones, teoría de algoritmos y teoría de la complejidad computacional. Su dominio es sobre problemas de optimización, en donde se busca minimizar o maximizar una función objetivo sujeta a restricciones que serán cubiertas por la solución encontrada (Garcia Calves, 2002).

Hoy en día existen diferentes problemas en donde se requiere una solución exacta, es decir, una única solución factible para un problema en específico. Pero en ocasiones, existen diferentes soluciones, en los cuales es necesario tener un criterio para discriminar entre ellas y llegar a un valor. A su vez existen clases de problemas según su complejidad, P (polinomial) y NP-Completos (los complejos de resolver).

Una definición simple de cada uno de ellos sería la siguiente:

- P: problemas que se pueden resolver en tiempo polinómico.
- NP-Completo: problemas que se pueden comprobar en tiempo polinómico.

Los problemas de clase P en teoría son más sencillos de resolver, ejemplo de estos son problemas lineales, en estos problemas tanto la función objetivo como sus respectivas restricciones, están expresadas de forma lineal. Así, estos problemas en particular

son solucionados utilizando el método simplex, técnica muy usada en investigación de operaciones. El método simplex entrega como resultado el valor óptimo al problema tratado.

Los problemas NP-Completos no pueden ser solucionados por métodos lineales o exactos, dada su complejidad (tiempo de ejecución exponencial). Pero si podemos llegar a una solución aproximada al mismo. Los métodos que resultados aproximados son llamados métodos heurísticos y metaheurísticos. Para mayor información sobre la teoría de complejidad de problemas P y NP completos, leer a Flores Cabezas (2014).

En investigación de operaciones, los resultados obtenidos utilizando heurísticas, tienen un significado en cuanto al nivel de confianza de la solución obtenida o de que alcanza un alto grado de optimalidad y/o factibilidad, en otras palabras, cuan confiable es o no una solución. Las metaheurísticas son estrategias inteligentes para mejorar procedimientos heurísticos muy generales con un elevado rendimiento alcanzando un alto grado de confiabilidad. Es este último método que se aplicará a un problema NP-Completo (Moreno Perez, 2004).

2.2. Problema de empaquetamiento en compartimientos BPP

2.2.1. Definición

El problema de empaquetamiento en compartimientos, es un problema de dificultad NP-Completo. Consiste en el embalaje de un conjunto de objetos en varios paquetes o cajas tal que el peso o volumen total no exceda el valor máximo permitido por las cajas y se ocupe la menor cantidad de cajas posible (Daza et al., 2009). Definimos un problema BPP de la siguiente manera (Dorronsoro Díaz, 2006):

1. Se tiene un conjunto finito de elementos de los cuales cada uno tiene su respectivo peso.

2. Existen restricciones de precedencia entre los elementos de tal manera que se genera un costo el cual podría ser infinito cuando el artículo j sigue al elemento i.

3. Se define un subconjunto ordenado de elementos, el cual es un grupo ordenado de ellos tales que:

   - El peso total del grupo de elementos ordenados no excede la capacidad del paquete o caja.
   - No exista costo infinito entre elementos adyacentes en el subconjunto.

El objetivo es crear una solución factible con el número mínimo de grupos ordenados, es decir, minimizar el número de paquetes o cajas (Bin). Si dos soluciones son iguales o tienen el mismo número de grupos ordenados, se escoge el que tenga costo mínimo.

Una entrada (input) típica para el BPP y su salida (output) correspondiente se ilustra en la Figura 2.1.

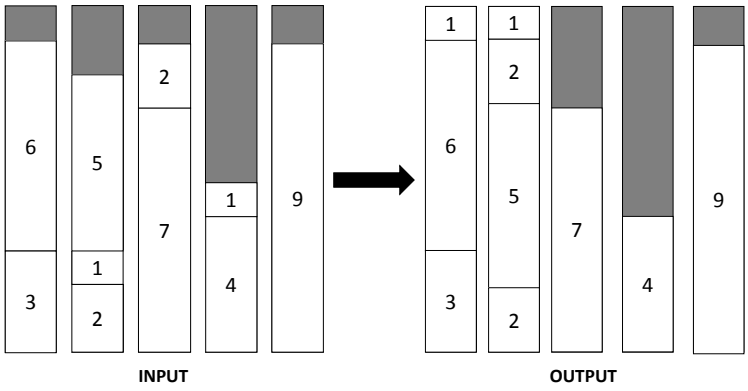

Figura 2.1: Entrada y salida de BPP, inspirado en (Coffman et al., 2016).

2.3. Problema del vendedor viajero TSP

2.3.1. Definición

El problema del vendedor viajero es un problema de complejidad computacional NP-Completo este constituye la situación general y de partida para formular otros problemas combinatorios más complejos, uno de ellos es, el ruteo de vehículos (Daza et al., 2009). El TSP consite en que un vendedor quiere encontrar, a partir de su ciudad natal,

la ruta mas corta que visite a todas las ciudades de los clientes para volver a su ciudad de origen, visitanto exactamente una vez a cada ciudad y en donde no hay demanda asociada a los clientes y tampoco restricciones temporales, existe un costo (distancia) de viajar de un nodo i a un nodo j, y de un nodo j a un nodo i. Si el TSP es simétrico el costo de viajar de un nodo i a otro j es el mismo del nodo j a nodo i, si es asimétrico el costo es distinto.

En la Figura 2.2 se puede apreciar que los clientes c_1, c_2, c_3 y c_4 estan conectados a través de rutas con sus respectivos costos de viaje; las lineas rojas representan la ruta solución que visita una vez a los clientes y a su vez los recorre a todos. El costo de esta ruta es 27, que es la ruta mas corta debido a que la distancia es la mínima.

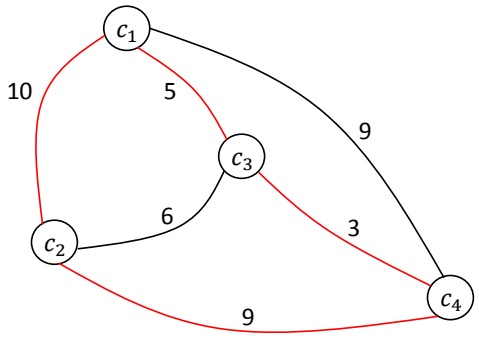

Figura 2.2: Una instancia de TSP, inspirado en (Lorena Stockdale, 2011).

2.4. Problema de ruteo de vehículos con límites de capacidad CVRP

2.4.1. Definición

Los problemas de ruteo han sido materia de investigación por más de 50 años dada su complejidad NP-Completo y su alta aplicabilidad a problemas de logística, transporte, entre otros. El problema de ruteo de vehículos con límites de capacidad (CVRP),

en términos sencillos, consta de satisfacer las determinadas demandas de clientes por medio de una flota de vehículos con capacidad limitada a través de un conjunto de rutas que empiezan y terminan en un único almacén en donde se busca representar el menor costo posible, este costo pueden ser: tiempo de transporte o ruta más corta (Orrego Cardozo, 2013). La Figura 2.3 ilustra un ejemplo de ruteo a partir de un único depósito.

CVRP puede verse como la unión de dos problemas definidos: TSP definido en la Sección 2.3.1 y BPP definido en la Sección 2.2.1. Esto debido a que en CVRP el objetivo es minimizar la cantidad de rutas, lo que se puede ver como minimizar la cantidad de cajas contenedoras BPP. Y además, encontrar la ruta mas corta para distintas rutas, que en TSP es buscar una ruta corta que visite a todos los clientes.

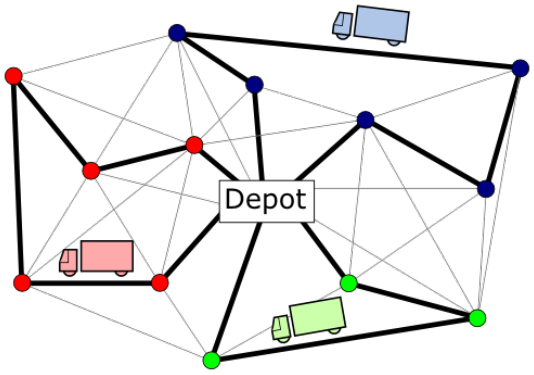

Figura 2.3: Ruteo de vehículos con un único depósito (Calvaresi, 2012).

Para CVRP, existen diversos métodos que dan soluciones aproximadas los cuales serán mencionados mas adelante. Para esta investigación se implementará un novedoso método metaheurístico que será explicado en las siguientes secciones.

2.4.2. Modelamiento matemático

La resolución de CVRP por métodos exáctos involucra modelarlo como un problema de programación lineal entera. En la literatura existen varios modelos matemáticos que describen a este problema, sin embargo, el modelo a describir proviene de (Méndez et al., 2010).

A continuación se describe la formulación para CVRP mediante variables binarias doblemente subindicadas x_{ij} que tomará el valor de 1 si el arco (i, j) es visitado y 0 en caso contrario.

Sea $G = (V, A)$ un grafo completamente conectado con cierta cantidad de vertices denotado por $V = \{1, ..., n\}$ y la cantidad de arcos que conecta a todos los vertices, A. Cada vértice corresponde a un cliente en determinada posición en un plano geográfico en donde el vértice 0 corresponde al depósito del problema. El costo asociado al recorrido entre los clientes (i, j), se obtiene de una matriz de costo c, en donde el costo de recorrer los clientes es $c_{i,j}$. Además, cada cliente tiene asociado una demanda q_i, $q \in \mathbb{R} y q > 0$.

CVRP entonces formulado como un problema de programación entera puede ser descrito de la siguiente forma:

$$mín \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.1}$$

Sujeto a:

$$\sum_{i \in V} x_{ij} = 1, \forall j \in V \backslash \{0\} \tag{2.2}$$

$$\sum_{j \in V} x_{ij} = 1, \forall i \in V \backslash \{0\} \tag{2.3}$$

$$\sum_{i \in V} x_{i0} = k \tag{2.4}$$

$$\sum_{j \in V} x_{0j} = k \tag{2.5}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S), \forall S \subset V \backslash \{0\}, S \neq \varnothing \tag{2.6}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V \tag{2.7}$$

Donde cada vehículo pertenece sólo a una única ruta k y cada ruta puede tener varios vehículos. La variable $r(S)$ corresponde al número mínimo de rutas necesarias para satisfacer a todos los clientes incluidos en S.

La Fórmula 2.1 es la función objetivo a optimizar sujeta a las restricciones anterior-

mente escritas. La función objetivo busca minimizar la suma de las distancias total de cada ruta. Las restricciones 2.2 y 2.3 indica la activación del arco (i, j) lo que determina el recorrido entre los vértices (i, j). Mientras que las restricciones 2.4 y 2.5 indican que k es la cantidad de vehículos utilizados en la solución y que todos los que parten en el depósito terminen en el. Finalmente la restricción 2.6 actúa como restricción de eliminación de sub-tours y a su vez que la demanda total de los clientes no supere la capacidad del vehículo que los visitará.

2.4.3. Tipos de problema de ruteo de vehículos

Como se dijo anteriormente, este problema tiene aplicaciones en muy diferentes campos por lo que surgen distintas restricciones que se deben tomar en cuenta, las cuales van desde el depósito hasta tener un rango de tiempo de recepción de pedido por parte del cliente. Estas restricciones del problema de ruteo de vehículos generan una definición de variantes de 8 casos típicos de problemas, en la Figura 2.4 podemos ver una ilustración de ellos para luego ser brevemente descritos.

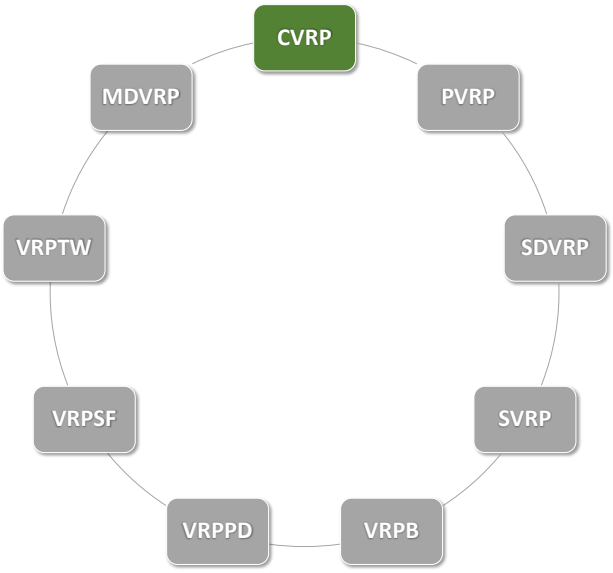

Figura 2.4: Elaboración propia a partir de (Orrego Cardozo, 2013) y (Dorronsoro Díaz, 2006).

- CVRP (Capacitated VRP): Es el problema más general de VRP que trata de una flota de vehículos con límite de capacidad que debe satisfacer la demanda de

una cantidad determinada de clientes por medio de la ruta más corta que inicia y termina en un único depósito (Dorronsoro Díaz, 2006) (Orrego Cardozo, 2013).

- MDVRP (Multiple Depot VRP): Una empresa puede tener varios depósitos, cada uno tiene su flota independiente, de los cuales debe cumplir con la demanda de cada cliente (Dorronsoro Díaz, 2006) (Orrego Cardozo, 2013), es un CVRP con la diferencia que tiene más de un único depósito.

- PVRP (Periodic VRP): Problema en el cual existe un período para la planificación de reparto de M días. Período en el cual se debe cumplir con los clientes (Dorronsoro Díaz, 2006) (Orrego Cardozo, 2013).

- SDVRP (Split Delivery VRP): Split delivery significa entrega dividida, en VRP, los clientes pueden ser servidos por distintos vehículos si reduce los costos generales. Esta variante de VRP es importante si los pedidos de los clientes son tan grandes como la capacidad del vehículo (Dorronsoro Díaz, 2006).

- SVRP (Stochastic VRP): Variante estocástica en la cual uno o varios componentes del problema son aleatorios, por ejemplo, cantidades aleatorias de clientes, demandas y tiempo en que un cliente es servido (Dorronsoro Díaz, 2006).

- VRPPD (VRP Pickup and Delivery): Es un VRP en donde cabe la posibilidad de que los clientes devuelvan algunos pedidos. Se necesita tener en cuenta que el pedido devuelto debe caber en el vehículo. Esta restricción hace que el problema sea más complejo dado que la planificación se hace más difícil (Dorronsoro Díaz, 2006).

- VRPB (RP with Backhauls): Se trata del mismo VRPPD, pero incluye la restricción de terminar todas las entregas antes de iniciar las diversas devoluciones de pedido de cada cliente (Orrego Cardozo, 2013).

- VRPSF (VRP with Satellite Facilities): “Un aspecto importante VRP que ha sido pasado por alto en gran medida es el uso de los servicios de satélite (que permita la comunicación entre el vehículo y el depósito) para reponer vehículos durante una ruta. Cuando sea posible, la reposición por satélite permite a los conductores a que sigan haciendo las entregas hasta el cierre de su turno sin necesidad de regresar a la estación central. Esta situación se presenta principalmente en la

distribución de combustibles y ciertos artículos al por menor"(Orrego Cardozo, 2013).

- VRPTW (VRP with Time Windows): Variante de VRP en la cual se le añade la restricción de una ventana de tiempo, es decir, el cliente está dispuesto a recibir el pedido durante el período de tiempo predeterminado (Orrego Cardozo, 2013).

2.5. Extremal optimisation

En esta parte se abordará EO; su inspiración, descripción, por qué surgió y dada su inspiración se presentarán las diferencias que existe con algoritmos evolutivos, ya que estos operan de forma distinta.

2.5.1. Inspiración

Extremal Optimisation es una metaheurística inspirada en la naturaleza, en la auto-organización crítica del campo de la física estadística de Bak-Sneppen y el modelo de co-evolución.

La auto-organización crítica trata de explicar la manifestación de fenómenos complejos como terremotos, formación de montones de arena, entre otros. El modelo de la co-evolución se trata de un cambio evolutivo recíproco que acontece en especies interactuantes, esta interacción produce que las especies se adapten al entorno (coadaptación) y a su vez la relación entre especies es muy cercana (coespeciación). En este tipo de modelo se requiere de: evolución de cada rasgo de una especie debida a presiones selectivas de otra especie (especificidad), rasgos de especies evolucionan conjuntamente (reciprocidad) y que los rasgos evolucionan al mismo tiempo (simultaneidad) (Gómez, 2003).

2.5.2. Descripción

Extremal Optimisation está inspirado en la auto-organización crítica y la coevolución de especies. Este método fue propuesto por Boetthcer and Percus en 1990. EO surgió porque las técnicas de matemática e inteligencia artificial convencional tienen dificultades ya que los espacios de búsqueda son grandes y complejos y, a su vez, existen

limitaciones de los sistemas computacionales actuales donde se realizan los métodos de resolución (Boettcher and Percus, 1999).

Las características de este método son: sólo requiere de un parámetro en el algoritmo llamado τ y utiliza estructuras algorítimicas sencillas, ya que solo trabaja con una solución que va evolucionando de acuerdo a la función de fitness que evalúa la contribución de una especie o variable en la solución global. Esto genera un uso mínimo en la cantidad de memoria al momento de ejecutar el método. Su funcionamiento trata de lo siguiente (Gómez Meneses, 2012):

1. Cada componente de la solución representa a una especie, ver Figura 2.5.
2. Se asigna un fitness a cada especie.
3. P_i es la probabilidad de que la i^{th} especie sea escogida.

$$P_i = i^{-\tau} \qquad \forall \quad i \qquad 1 \leq i \leq n \tag{2.8}$$

4. Se rankean las especies de la peor a la mejor.
5. Se selecciona una especie utilizando el método de selección la ruleta (RWS, del inglés Roulette Wheel Selection) usando la probabilidad P_i.
6. Se incorpora una nueva especie aleatoriamente en el ecosistema.
7. Se elimina la especie que degrada al ecosistema.
8. Se recalcula el fitness correspondiente a cada especie.

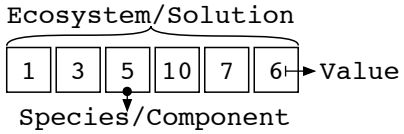

Figura 2.5: Vector solución (Gómez Meneses, 2012).

En la Figura 2.5 podemos ver que EO representa el ecositema a través de un vector solución, en donde cada componente contiene un valor referente al problema a aplicar. Dado el funcionamiento estocástico de EO, realizará la exploración del espacio de búsqueda que se tenga, saltando de un lugar a otro guiado por la función de fitness, en busca de la mejor solución a partir una solución inicial como lo ilustra la Figura 2.6.

Algorithm 1 Pseudocódigo general de EO (Gómez Meneses, 2012)

Generate an initial random solution $X = (x_1, x_2, ..., x_n)$;
Set $X_{best} = X$;
Generate the probabilities array P according to Equation 2.8;
for a present number of iterations **do**
  Evaluate and rank fitness λ_i for each x_i from worst to best, $1 \leq i \leq n$;
  Select component j based on the probability of its rank P_i using RWS;
  x_j = Generate a random appropriate value that is not equal to x_j;
  $Eva(X)$ = Evaluate the new solution;
  if $Eva(X) < Eva(X_{best})$ **then**
    $X_{best} = X$;
  end if
end for
return X_{best} y Eva(X_{best});

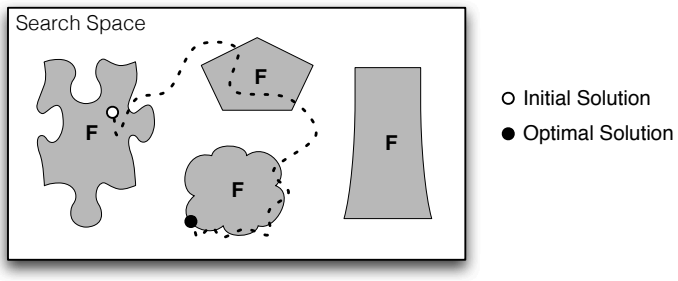

Figura 2.6: Ejemplo de espacio de búsqueda de EO (Gómez Meneses, 2012).

Los puntos ya mencionados son importantes para obtener una implementación de EO ya que cada componente de la solución es una especie, de las cuales se evaluará su fitness para ser rankeados del peor al mejor. Una vez ordenados, se seleccionará una especie mediante el método de la ruleta a partir de una probabilidad P_i para incorporar una nueva especie de forma randómica al ecosistema. Una vez realizado el proceso

anterior, la nueva solución se evalua y compara con la mejor encontrada, si la nueva es mejor que la encontrada entonces se actualiza la mejor solución. Finalmente se retorna la solución y evaluación de la misma. Este proceso se refleja en el Algoritmo 1.

2.5.3. Diferencias con Algoritmos genéticos

El hecho de que este método está inspirado en la coevolución y auto-organización crítica hace que EO se diferencie de las demás metaheurísticas evolutivas. Por ejemplo, si comparamos EO con Algoritmos Genéticos se tienen 4 diferencias que a continuación se describen:

- EO selecciona al peor componente del sistema y lo elimina debido a que tiene la mayor probabilidad de ser seleccionado, es decir, elige un elemento con el peor fitness por lo que este elemento no sobrevive a la siguiente generación. Finalmente este método elimina los elementos que degraban el sistema en lugar de seleccionar los mejores elementos que sobreviven a la siguiente generación como es en algoritmos genéticos.

- EO Requiere de muy pocos parámetros y operadores genéticos, a diferencia de algoritmos genéticos que requieren definir un conjunto de parámetros y hacen uso de operadores genéticos (cruzamiento, mutación, entre otros) que aumentan su complejidad.

- EO calcula el fitness por cada elemento en base a su contribución en la función objetivo de la solución, en cambio algoritmos genéticos evaluan el fitness de todos los elementos de la solución.

- EO trabaja con una única solución, a diferencia de algoritmos genéticos que trabaja con una población de soluciones.

Capítulo 3

Estado del arte

3.1. Extremal optimisation

En este análisis de la investigación se presentan trabajos de investigación de EO que están relacionados con el tema de este proyecto. Se mencionarán de forma breve. Cabe destacar que EO ha sido aplicado en algunos problemas NP-completos de optimización combinatoria aunque no tan extensivamente como otras metaheurísticas tales como Recocido Simulado, Búsqueda Tabú y Colonia de Hormigas (Gómez Meneses, 2012). EO ha sido aplicado a problemas multiobjetivos y de un solo objetivo, para nuestro interés, solo abordaremos problemas de un solo objetivo.

3.1.1. Aplicaciones

Extremal Optimisation fue propuesto por Boettcher y Percus en 1990, método novedoso para la resolución de problemas NP-Completos con pocos operadores y parámetros, por lo tanto, el algoritmo es relativamente simple. Para demostrar su eficacia, se puso a prueba en una serie de problemas combinatorios (Boettcher and Percus, 1999) tales como el vendedor viajero, graph partitioning, 3-colouring and maximum satisfiability, entre otros.

Luego de poner a prueba este algoritmo, se compararon sus resultados con otros algoritmos metaheurísticos como por ejemplo Algoritmos Genéticos, Recocido Simulado y Monte Carlo basado en paseos aleatorios, en donde Extremal Optimisation resultó

ser igual o competitivamente superior, dependiendo del problema.

Alena Shmygelska (Shmygelska, 2007) implementó y evaluó Extremal Optimisation al problema de plegado de proteínas. Esto debido a que la mayoría de los métodos computacionales pierden de vista lugares de búsqueda en las interacciones y permanecen insatisfechas, lo cual dio paso a implementar Extremal Optimisation que asegura visitar las partes inexploradas prometedoras. Finalmente demostró que EO se compara favorablemente en el desempeño contrastándolo con el método Monte Carlo en alcanzar el estado nativo para proteínas G.

Min-Rong Chen, Xia Li y Xi Zhang Lu (Chen et al., 2010) propusieron un algoritmo híbrido llamado PSO-EO de optimización de Enjambre de Partículas (PSO) y Extremal Optimisation. Dada la prematura convergencia de PSO, implementaron EO para lograr prevenir esta pronta convergencia y se logre una mayor exploración del campo de búsqueda. Donde los resultados de la simulación demostraron que PSO-EO es muy adecuado para problemas multimodales/unimodales.

Guo-Qiang Zeng (Zeng et al., 2014) desarrollaron un método mejorado de Extremal Optimisation basado en la población (IRPEO, del inglés Improved Real-coded Population-based Extremal Optimization) para problemas de optimización sin restricciones continuas. En donde la operación principal es incluir la generación de la población inicial de forma aleatoria, la evaluación fitness individual y poblacional, la selección de los malos elementos de acuerdo con la distribución de probabilidad de ley de potencia, generación de nueva población basada en la mutación aleatoria uniforme, y la actualización de la población mediante la aceptación de la nueva población incondicionalmente. Finalmente demostraron que IRPEO es competitivo o incluso mejor que varias versiones de Algoritmos Genéticos. Por otro lado, IRPEO también mostró superioridad a otros algoritmos evolutivos como EO canónico basado en la población, optimización por enjambre de partículas (PSO), y el híbrido PSO-EO.

Actualmente, se han desarrollado algoritmos híbridos que incluyen a EO como método de búsqueda (Randall et al., 2015), dando buenos resultados que prometen investigaciones futuras. Randall et al. (2015), examinaron tres técnicas inspiradas en la

naturaleza, Enjambre de Particulas, Extremal Optimisation y Búsqueda Tabú para aplicarlo a un problema de diseño de perfiles aerodinámicos, estudio que reveló que los tres algoritmos tenían patrones de búsqueda distintivos y favorecieron las regiones de exploración. Finalmente concluyeron que la hibridación de estos algoritmos parece prometedor y será explorado en futuras investigaciones de estos y otros algoritmos.

3.1.2. Aplicado a ruteo

En esta investigación abordaremos un problema de ruteo, debido a esto, en esta sección se presentarán las aplicaciones de EO a problemas de ruteo y sus conclusiones en cada investigación.

Boettcher y Percus (Boettcher and Percus, 1999) aplicaron Extremal Optimisation al problema del vendedor viajero y se compararon los resultados con dos técnicas, Algoritmos Exactos y Algoritmos Genéticos para distintas cantidades de ciudades y dos formas de distancia, euclidiana y aleatoria. En la distancia euclidiana se calcula usando la siguinte fórmula, $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, y la distancia aleatoria simplemente son distancias aleatorias que hacen al cálculo no euclidiano. Para el primero puede corresponder a un vendedor viajero tratando de minimizar costos de conducción y el segundo a un vendedor viajero tratando de minimizar los gastos en una serie de vuelos de aerolíneas, cuyos precios ciertamente no pueden ser usados en el cálculo euclidiano. Los resultados se compararon con las técnicas ya mencionadas, y Extremal Optimisation resultó ser competitivo en sus resultados cuando la distancia es aleatoria, a diferencia de la distancia euclidiana que mostró una diferencia en sus resultados a medida que la cantidad de ciudades aumentaba.

También aplicaron EO a graph partitioning problem (Boettcher and Percus, 1999), (Boettcher and Percus, 2000) , en donde se contrastó con distintas metahurísticas como: Algoritmos Genéticos y Recocido Simulado. Problema el cual consta de un conjunto de puntos N (N es par) y bordes que conectan ciertos pares de puntos, con la finalidad de encontrar una manera de dividir los puntos en dos subconjuntos iguales, cada uno de tamaño $N/2$, con un numero mínimo de bordes de corte a través de la partición. Finalmente concluyeron que EO pareciera ser bastante exitoso en una gran variedad

de puntos, donde también se vió el comportamiento promedio de EO de forma variada, encontrando el mejor resultado al menos 2 veces en las 30 instancias aplicadas. Los resultados obtenidos en otras instancias variaron en 1 % con respecto a la mejor solución.

Marcus Randall, Tim Hendtlass y Andrew Lewis (Randall et al., 2009), realizaron un análisis de los movimientos seleccionados por EO en donde desarrollaron la inclusión de técnicas para restringir el proceso y así permitir que la busqueda sea en espacios viables y factibles, proporcionar una restauración genérica de viabilidad parcial para conducir la solución al espacio factible y el desarrollo de un modelo basado en la población de la metaheurística que elimina de forma adaptativa e introduce nuevos miembros. El problema que abordaron fue el problema de tipo de asignación, en donde demostraron que EO no converge en soluciones esto debido a su naturaleza estocástica. Sin embargo, el desarrollo de EO en su forma canónica en muchas ocaciones da lugar a soluciones pobres demostrándolo con Bin Paking Problem. Con el fin de elevar el rendimiento EO requiere de apoyo adicional. En esta investigación por Marcus Randall et. all. realizaron la propuesta de algoritmos generalizados en donde se puede manejar las restricciones, parcialmente restaurar la factibilidad de una solución y crear/mantener una población de soluciones. Estos combinados con busqueda local, han demostrado claramente que EO es capaz de producir soluciones de buena calidad.

Yu-Wang Chena, Yao-Jia Zhu, Gen-Ke Yang y Yong-Zai Lu (Chen et al., 2011), presentaron un EO mejorado llamado Improved Extremal Optimization (IEO) para resolver el problema del vendedor viajero asimétrico (ATSP). El algoritmo procede a través de dos pasos principales: la dinámica de EO y la optimización cooperativa, método de optimización para problemas complejos (Huang, 2007). Este enfoque brinda como resultados la capacidad confiable de exploración y explotación del espacio de solución, en un conjunto de simulación en instancias de ATSP. Proporcionando, además, un rendimiento superior tanto en la eficacia y eficiencia computacional de complejos problemas de optimización combinatoria.

3.2. Problema de ruteo de vehículos con límites de capacidad

3.2.1. Técnicas para resolver problema

Dada la gran aplicabilidad de problemas de ruteo a problemas reales ha producido variantes de VRP descritas anteriormente. Variantes con distintas restricciones que lo hacen más difíciles de resolver, dentro de estas tenemos, vehículos con distintas capacidades, más de un depósito, entre otros. A su vez, existen distintas técnicas que resuelven este tipo de problemas. Algunas técnicas de solución son: técnicas exactas, técnicas heurísticas y técnicas metaheurísticas, cada una de ellas será descrita brevemente y se mencionarán algunos algoritmos ocupados.(Orrego Cardozo, 2013).

3.2.1.1. Técnicas exactas

Las técnicas exactas tienen la particularidad de que siempre arrojan una solución óptima, es decir, da la solución a un problema de forma exacta. Esta técnica no es adecuada cuando se requieren soluciones rápidas sobre problemas muy grandes en cantidad de variables.(Orrego Cardozo, 2013). Algunas técnicas son:

- Ramificación y acotamiento (Branch and Bound, hasta 100 nodos, Fisher, 1994).
- Ramificación y corte (Branch and Cut, Augerat et al.,1998).
- Programación dinámica (Balinski y Quandt, 1964).
- Programación lineal entera (Balinski y Quandt, 1964).

3.2.1.2. Técnicas heurísticas

Una heurística es un algoritmo que encuentra soluciones suficientemente buenas en tiempos de computación cortos a un problema en particular sin importar si la solución es óptima. Los algoritmos heurísticos son eficientes ya que ejecutan procedimientos razonablemente rápidos.(Brownlee, 2011)(Dorronsoro Díaz, 2006). Algunas heurísticas son:

- Método del ahorro (Clarke and Wright (1964)).

- Asignar primero y rutear después.
- Rutear primero y asignar después.
- Matching Based.

3.2.1.3. Técnicas metaheurísticas

Las metaherurísticas pueden considerarse un marco algorítmico general que puede ser aplicado a diferentes problemas de optimización realizando pocas modificaciones para adaptarlas a un problema específico, a diferencia de las heurísticas que sí son para un problema en particular. Las soluciones que encuentre una metaheurística serán una aproximación, por lo que no se sabrá si una solución es óptima. (Brownlee, 2011) (Dorronsoro Díaz, 2006). Algunas metaheurísticas son:

- Algoritmos Genéticos.
- Colonia de Hormigas.
- Búsqueda Tabú.
- Recocido Simulado.

Capítulo 4

Implementación

En este capítulo se mostrará el modelamiendo de la solución, la fórmula aplicada para el cálculo de fitness a cada especie de la solución, la búsqueda local que se ocupará para mejorar las soluciones y finalmente el pseudocódigo de EO aplicado a CVRP que describe la estructura del código. Para esta etapa de la investigación veremos dos versiones de EO. Su modelamiento y funcionamiento es el mismo, pero el cálculo de fitness se obtiene a partir de dos fórmulas distintas.

4.1. Modelamiento del problema

La forma de representar un ecosistema en EO es a través de un vector. En otras investigaciones de CVRP se trata la solución mediante n rutas y sus respectivos clientes (Orrego Cardozo, 2013), sólo el orden de visita de clientes asociados a rutas (Salah Niazy and Badr, 2012), entre otros. Lo anterior nos inspiró a diseñar un modelo de un vector que contiene los clientes y otro que indica a qué ruta pertenece dicho cliente, la Figura 4.1 ilustra cómo es el modelo, qué contiene y cómo es.

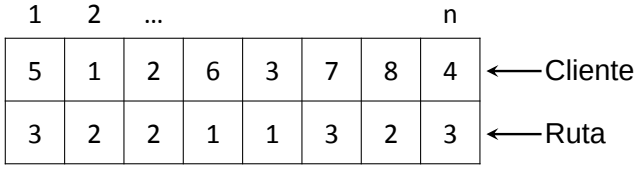

Figura 4.1: Representación de vector solución de CVRP.

El modelo propuesto ilustrado en Figura 4.1 es usado en las dos versiones que se desarrollaron de EO. Para entenderlo, se debe leer de izquierda a derecha y se interpreta de la siguiente forma: la ruta 1 visita a clientes 6 y 3, ruta 2 a los clientes 1, 2 y 8 y ruta 3 a los clientes 5, 7 y 4.

CVRP es un problema de minimización del costo de transporte en rutas, el costo de un cliente a otro se mide en distancia. Por lo anterior, una forma de evaluar el vector solución representado en Figura 4.1 sería realizando una suma de las distancias entre los clientes de cada ruta ilustrado en Fórmula 4.1.

Lo que realiza la primera parte de la Fórmula 4.1 es lo siguiente; para la ruta k sumará las distancias entre sus clientes sin contar la distancia entre el depósito y el primer cliente de la respectiva ruta. De la misma forma, no cuenta la distancia entre el último cliente de la ruta con el depósito. Sin embargo suma todas las distancias entre clientes de las respectivas k rutas. La segunda parte de esta fórmula, si considera la distancia entre el depósito y el primer cliente de la k ruta correspondiente en donde 0 denota el depósito y $x_{primero}$ el primer cliente de la ruta. La tercera parte realiza la suma entre el último cliente (x_{ultimo}) y el depósito (0). Finalmente el objetivo de esta fórmula es minimizar la distancia.

$$min = \sum_{k=1}^{ruta} \sum_{i \in k} Dist_{(x_i)(x_{i+1})} + \sum_{k=1}^{ruta} Dist_{(0)(x_{primero})} + \sum_{k=1}^{ruta} Dist_{(x_{ultimo})(0)} \tag{4.1}$$

Donde:

$ruta$ = es el número total de rutas de la solución,

k = es la ruta que se está evaluando,

i = es cliente que se está evaluando,
$Dist_{ij}$ = es la matriz de distancia que hay entre el cliente i al cliente j,
x_i = es el cliente perteneciente a la ruta k correpondiente.
$x_{primero}$ = es el primer cliente de la ruta k,
x_{ultimo} = es el último cliente de la ruta k.

4.2. Fitness

Para saber que tan apto o que tan bueno es un nodo o cliente, se ideó una formulación para evaluarlos. Esto permite saber la contribución de cada variable x_i del vector solución. Las ideas son distintas entre EO1 y EO2 que corresponden a Fórmula 4.2 y Fórmula 4.5 respectivamente. Cabe destacar que la Fórmula 4.2 es inspirada a partir de (Gómez Meneses, 2012), y la Fórmula 4.5 es creación en conjunto entre el profesor guía de tesis y alumno respectivos.

$$\lambda(x_i) = \begin{cases} -(C - \sum_{j \in ruta(x_i)} d_j) - \frac{D_i}{D_{max}+1} & , \quad \forall i \\ -(C - \sum_{j \in ruta(x_i)} d_j) - \frac{d_i}{d_{max}+1} & , \quad \forall i \in I \end{cases} \tag{4.2}$$

$$D_i = \sum_{j=1}^{n} Dist_{ij} \tag{4.3}$$

$$D_{max} = max\{D_i | i = 1, ..., n\} \tag{4.4}$$

Donde:
C = es la Capacidad del vehículo,
$\sum d_j$ = es la suma de todas las demandas en la ruta asignada al cliente x_i,
d_i = es la demanda del cliente x_i,
d_{max} = es la demanda máxima de todos los clientes,
n = es la cantidad de clientes,

$Dist_{ij}$ = es la distancia que hay entre el cliente i al cliente j,
D_i = es la suma de las distancias del cliente i con todos los clientes,
I = es el conjunto de rutas infactibles,
D_{max} = es el máximo valor D_i.

Para analizar la factibilidad de la solución considerada para ambas versiones EO1 y EO2, se realizará de la siguiente forma. Si todas las rutas son factibles, es decir, no se excede la capacidad de cada vehículo, entonces la solución es factible. Si, por lo menos una ruta excede la capacidad del vehículo, entonces se considerará la solución como infactible.

La versión EO1 ocupa las Fórmulas 4.2, 4.3 y 4.4 según la factibilidad de la solución. Cuando una solución es factible, se evaluará el fitness de todos los clientes según la primera parte de la Fórmula 4.2 y se rankearán de mayor a menor. En la evaluación se tomarán en cuenta dos criterios. La primera es la diferencia entre la capacidad del vehículo y la demanda de la ruta a la cual pertenece el nodo. La segunda es el porcentaje de aporte de distancia del nodo con respecto a la distancia máxima entre todos los clientes. Mientras más negativo es el valor, más degrada al ecosistema esto porque será el cliente que tenga mayor porcentaje de aporte en distancia y que pertenece a una ruta en donde existe mayor espacio de capacidad disponible.

Si por el contrario la solución es infactible, entonces se evaluará el fitness de todos los clientes que pertenecen a rutas infactibles según la segunda parte de la Fórmula 4.2 y se rankearán de menor a mayor. Aquí también se tomarán en cuenta dos criterios para la evaluación. Primero, la diferencia entre la sobrecapacidad del vehículo y la demanda de la ruta a la cual pertenece el nodo. Segundo, el porcentaje de aporte de demanda que tiene el nodo. Mientras mas positivo es el valor, mas degrada al ecosistema esto porque será el cliente que tenga mayor porcentaje de aporte en demanda y que pertenece a la ruta más sobrecargada.

$$\lambda(x_i) = \begin{cases} Dist_{(anterior)(x_i)} + Dist_{(x_i)(posterior)} & , \quad \forall i \\ -(d_i + \frac{D_i}{D_{max}+1}) & , \quad \forall i \in I \end{cases} \tag{4.5}$$

$$D_i = \sum_{j=1}^{n} Dist_{ij} \tag{4.6}$$

$$D_{max} = max\{D_i | i = 1, ..., n\} \tag{4.7}$$

Donde:

d_i = es la demanda del cliente x_i,
n = es la cantidad de clientes,
$Dist_{ij}$ = es la distancia que hay entre el cliente i al cliente j,
D_i = es la suma de las distancias del cliente i con todos los clientes,
I = es el conjunto de rutas infactibles,
D_{max} = es el máximo valor D_i.

La versión EO2 ocupa las Fórmulas 4.5, 4.6 y 4.7. Esta función también trabaja con dos evaluaciones distintas de acuerdo a la factibilidad de la solución. Cuando una solución es factible, se evaluará el fitness de todos los clientes según la primera parte de la Fórmula 4.5 y se rankearán de mayor a menor. En la evaluación se realiza la suma de las distancias del cliente anterior con el cliente en evaluación, y luego con el cliente posterior. Mientras mas positivo es el valor, mas degrada al ecosistema esto porque la distancia entre el cliente con su antesesor y posterior son grandes, es decir, se encuentran lejos uno de otro.

Si por el contrario la solución es infactible, entonces se evaluará el fitness de todos los clientes que pertenecen a rutas infactibles según la segunda parte de la Fórmula 4.5 y se rankearán de menor a mayor. En la evaluación se realizará la suma de la demanda

del respectivo cliente y el porcentaje de aporte de distancia. Todo esto multiplicado por -1. Cada cliente tiene su respectiva demanda, pero existe la posibilidad que exista la misma demanda para distintos clientes, para diferenciar uno de otro se sumó el porcentaje de aporte de distancia. Mientras mas negativo es el valor, mas degrada al ecosistema esto porque la infactibilidad la producen los clientes que contienen la mayor demanda.

4.3. Búsqueda Local

La idea de aplicar una búsqueda local es mejorar las soluciones encontradas por EO en su forma canónica y así estar cada vez mas cerca o igualar las mejores soluciones encontradas hasta el momento de cada tipo de problema. Lo anterior es realizado mediante el proceso de explorar repetitivamente una vecindad de una solución en busca de una mejor solución. Cuando no logra mejorar la solución encontrada se dice que se llega a un óptimo local (Resende and Velarde, 2003). El vecindario se define como el grupo de soluciones obtenidas a partir de pequeños movimientos en la solución que se quiere mejorar.

La búsqueda local que se ocupará en esta investigación es *2opt* y *1 intercambio*, la razón del uso de esta es que ya sido usado en problemas de tipo ruteo de vehículos y vendedor viajero dando buenos resultados (Jaramillo Posada, 2012).

El proceso de *2opt* ilustrado en Figura 4.2 realiza intercambio de clientes entre arcos de una misma ruta. En el ejemplo de la Figura 4.2 se seleccionan de forma aleatoria los arcos (A,B) y (C,D) para realizar una nueva conexión intercambiando las secuencias (A,D) y (C,B) produciendo una ruta distinta. Como este proceso es realizado en una misma ruta y solo se realizan intercambios, la factibilidad de la ruta permanece igual.

Por otra parte, tenemos el mecanismo de *1 intercambio* ilustrado en la Figura 4.3. Esta parte de la búsqueda local realiza combinaciones en la solución mediante intercambios con el nodo siguiente hasta el número de nodos que se tengan. Así, en la Figura 4.3 se puede apreciar una combinación del nodo B con el nodo C en el vector solución.

Como en el vector solución se tienen todos los clientes y rutas, un cliente puede pasar de una ruta a otra para así encontrar una mejor solución.

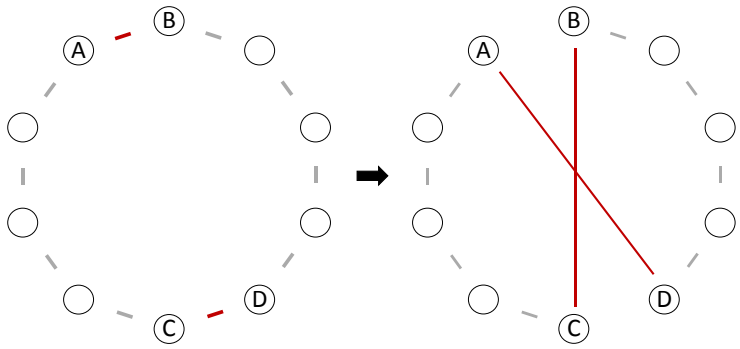

Figura 4.2: Procedimiento 2opt inspirado en (Alba and Dorronsoro, 2008).

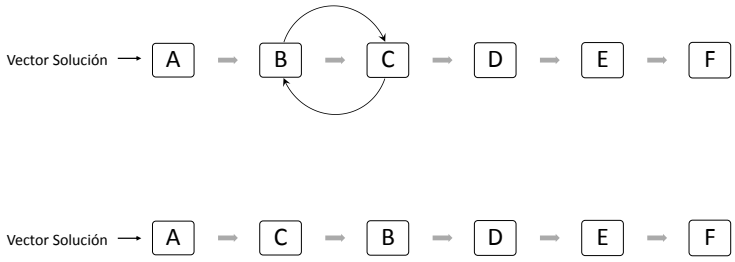

Figura 4.3: Procedimiento 1 intercambio inspirado en (Alba and Dorronsoro, 2008).

Para ambos procedimientos, *2opt* y *1 intercambio*, se necesita un límite de iteraciones dado que una búsquda local encuentra un óptimo local. En base a lo anterior se decidió ocupar un límite de 20 iteraciones propuesto por la literatura (Alba and Dorronsoro, 2009), (Alba and Dorronsoro, 2008), a su vez el Algoritmo 4 fue inspirado en los autores mencionados. A continuación se presentará el pseudocódigo del algoritmo

de búsqueda local implementado y las funciones que contiene: *2opt* ilustrado en Algoritmo 2 y *1 intercambio* ilustrado en el Algoritmo 3.

El Algoritmo 2 tiene dos parámetros de entrada: la solución actual y la ruta. Se guarda la solución en una variable auxiliar para luego recuperar y contabilizar los clientes que están en la ruta respectiva. Si la cantidad de clientes es mayor a 2, entonces se elije un cliente de forma randómica distinto al número de clientes, esto porque se realiza el intercambio del cliente que sigue, si se elije el último cliente su sucesor es el depósito y si se intercambia el depósito se creará una nueva ruta.

Una vez realizado el proceso de la linea 8 del Algoritmo 2, se elije un nuevo cliente de forma randómica distinto al número de clientes y al cliente seleccionado en el proceso anterior, esto porque no tiene sentido realizar un intercambio del mismo cliente.

Luego de tener los dos clientes seleccionados, se procede al respectivo intercambio, para finalizar en la actualización de la variable auxiliar que será retornada por el Algoritmo 2 y utilizada posteriormente por Algoritmo 4.

Algorithm 2 Pseudocódigo de 2opt

Require: Solución S, ruta J

1: $S\_aux = S$;
2: **for** L número de clientes **do**
3: **if** S_aux_{1L} es igual a J **then**
4: Recuperar clientes de ruta J;
5: **end if**
6: **end for**
7: **if** cantidad de clientes recuperados de ruta $J > 2$ **then**
8: Elejir un cliente randómicamente, dentro del conjunto de valores posibles;
9: Elejir otro cliente randómicamente, dentro del conjunto de valores posibles;
10: Realizar intercambio de clientes correspondientes;
11: **end if**
12: Actualizar S_aux con el intercambio anterior;
13: **return** S_aux;

El Algoritmo 3 tiene 3 parámetros de entrada: cliente L, cliente K y la solución actual. La solución es almacenada en una variable auxiliar para realizar cambios sobre ella. Este proceso consta en el intercambio del cliente L con el cliente K para finalmen-

te actualizar la variable auxiliar que será retornada para su posterior uso en Algoritmo 4.

Algorithm 3 Pseudocódigo de 1Int

Require: cliente L, cliente K, Solución S

1: $S\_aux = S$;
2: Intercambiar cliente L con cliente K;
3: Actualizar S_aux con el intercambio correspondiente;
4: **return** S_aux;

Finalmente se tiene el Algorimo 4 completo, que se hace uso de los Algoritmos 2 y 3 para generar la búsqueda local de nombre *2o1i*. Las versiones de EO que terminen en *2o1i* indican que estas contienen este método de búsqueda local.

El proceso de la búsqueda local *2o1i*, será explicado de la siguiente manera. Se necesita de la solución inicial como parámetro de entrada, la cual será almacenada en dos variables. Una para el proceso de *2opt* y la otra para el proceso de *1 intercambio*. Variables que significarán la mejor solución encontrada hasta el momento de cada proceso para su posterior análisis.

Primero, está el proceso de *2opt*, en donde, para un número determinado de iteraciones, se realiza lo siguiente: se ordena la solución por ruta, primero la ruta 1, 2, 3 y así sucesivamente sin perder el orden de los clientes. Para la ruta J se realiza un proceso *2opt* descrito en el Algoritmo 2 para luego analizar su factibilidad y si esta es positiva, es decir factible, comparará la evaluación de la nueva solución con la evaluación de la mejor solución de *2opt*. Si la comparación resulta verdadera, la solución encontrada será guardada como la mejor solución de *2opt*. Este proceso lo realizará por cada ruta, y por cada iteración antes mencionada.

Segundo, está el proceso de *1 intercambio* en donde para un número determinado de iteraciones, se realiza lo siguiente: por cada cliente L se realizará el intercambio por cliente K. La variable K, siempre será el cliente sucesor del cliente L. Para ambos, se procede al intercambio y luego se analiza la factibilidad de la nueva solución obtenida, si es factible, se compara la evaluación de la nueva solución con la evaluación de la mejor solución para *1 intercambio*. Si la nueva solución es mejor, entonces es guardada

como la mejor solución de *1 intercambio*. Este proceso también se realizará por cada iteración determinada y por cada cliente.

Al finalizar los procesos *2opt* y *1 intercambio* en el Algoritmo 4, se tienen 3 soluciones a analizar. Primero, se busca la mejor solución entre los procesos *2opt* y *1 intercambio*, para luego comparar el resultado anterior con la solución inicial y así encontrar la mejor solución que finalmente será retornada como la solución encontrada por la búsqueda local.

Algorithm 4 Pseudocódigo de búsqueda local 2o1i para ser usado en dos versiones

Require: Solución S;
1: $Best2\_Opt = S$;
2: $Best1\_Int = S$;
3: $S\_aux1 = S$;
4: $S\_aux2 = S$;
5: //Primero, se realiza el 2opt
6: **for** número máximo I de iteraciones para búsqueda local **do**
7: $ordenar\_solucion\_aux(S\_aux1)$;
8: **for** J número de rutas **do**
9: $S\_aux1 = 2opt(S, J)$;
10: **if** Solución S_aux1 es factible **then**
11: **if** $Evaluar(S\_aux1) < Evaluar(Best2\_Opt)$ **then**
12: $Best2\_Opt \leftarrow S\_aux1$;
13: **end if**
14: **end if**
15: **end for**
16: **end for**
17: //Segundo, se realiza el 1Int
18: **for** número máximo I de iteraciones para búsqueda local **do**
19: **for** L número de clientes **do**
20: **for** $K = L + 1$ número de clientes **do**
21: $S\_aux2 = 1Int(L, K, S)$;
22: **if** Solución S_aux2 es factible **then**
23: **if** $Evaluar(S\_aux2) < Evaluar(Best2\_Int)$ **then**
24: $Best2\_Int \leftarrow S\_aux2$;
25: **end if**
26: **end if**
27: **end for**
28: **end for**
29: **end for**
30: **if** $Evaluar(Best2\_Opt) < Evaluar(Best2\_Int)$ **then**
31: $Best2o1i \leftarrow Best2\_Opt$;
32: **else**
33: $Best2o1i \leftarrow Best2\_Int$;
34: **end if**
35: **if** $Evaluar(Best2o1i) < Evaluar(S)$ **then**
36: $S \leftarrow Best2o1i$;
37: **end if**
38: **return** S;

Como ya hemos visto, CVRP trata de un problema que abarca otros dos, BPP y

TSP. Esto aumenta su complejidad en encontrar una solución. Ralphs et al. (2001) menciona que los enfoques de resolución de CVRP actuales, tratan al problema como un problema de enrutamiento con distintos métodos de resolución, pero se sabe muy poco sobre una resolución desde un enfoque de empaquetamiento. Dicho esto y lo mencionado en sección introductoria (página 4 párrafo 1) nos motivó a realizar otra implementación de una búsqueda local Falkenauer, usado en (Falkenauer, 1996).

La heurística propuesta por Falkenauer (Falkenauer, 1996), es una heurística de agrupación (Cruz-Reyes et al., 2012).Cuyo funcionamiento se explicará con ayuda de la Figura 4.4. Se tienen 6 contenedores de capacidad 10, de los cuales se abren los 2 que tengan menor cantidad de ítem, estos serán items libres. Quedan 4 contenedores para realizar los 3 siguientes pasos: intentar reemplazar 2 items del primer contenedor por 2 items libres, reemplazar 2 items del segundo contenedor por 1 item libre y reemplazar 1 item del tercer contenedor por 1 item libre y el cuarto contenedor queda igual. Finalmente se debe reinsertar los items libres usando la heurística de First Fit Decreasing (FFD) (Falkenauer, 1996) para obtener una nueva solución de empaquetado. Aplicandolo a CVRP, cada contenedor sería un vehículo con su respectiva capacidad y cada item sería un cliente y su demanda.

The solution before local search (the bin capacity is 10):
The bins: | 3 3 3 | 6 2 1 | 5 2 | 4 3 | 7 2 | 5 4 |

Open the two smallest bins:
Remaining: | 3 3 3 | 6 2 1 | 7 2 | 5 4 |
Free items: 5, 4, 3, 2

Try to replace 2 current items by 2 free items, 2 current by 1 free or 1 current by 1 free:
First bin: 3 3 3 → 3 5 2 new free: 4, 3, 3, 3
Second bin: 6 2 1 → 6 4 new free: 3, 3, 3, 2, 1
Third bin: 7 2 → 7 3 new free: 3, 3, 2, 2, 1
Fourth bin: 5 4 stays the same

Reinsert the free items using FFD:
Fourth bin: 5 4 → 5 4 1
Make new bin: 3 3 2 2
Final solution: | 3 5 2 | 6 4 | 7 3 | 5 4 1 | 3 3 2 2 |

Repeat the procedure: no further improvement possible

Figura 4.4: Un ejemplo de el uso de algoritmo de búsqueda local de Falkenauer, (Falkenauer, 1996).

Gómez-Meneses and Randall (2009) ocupa Best Fit Decreasing (BFD) para soluciones iniciales y así tener la cantidad mínima de contenedores, lo cual nos inspiró para usar BFD en vez de FFD al momento de reagrupar los items libres. En BFD los items son ordenados de forma decreciente según su peso y luego intenta poner un item en un contenedor que tenga un espacio libre mínimo.

A continuación se describirá el proceso de búsqueda local de Falkenauer implementado en esta investigación ilustrado en el Algoritmo 6. Se debe destacar que las versiones de EO1_2o_f y EO2_2o_f contienen dos tipos de búsqueda local implementadas, una es de enrutamiento (2 opt) y la otra es de Falkenauer.

Primero, se describirá el proceso de BFD ilustrado en Algoritmo 5 el cual trabaja con los clientes libres obtenidos de las dos rutas que tienen mas espacio libre, para ordenarlos de forma decreciente. Para cada cliente L libre, se calcula una variable $sobra_i$ el cual

se obtiene a partir de la resta de la $capacidad_i$ y la $demanda_L$ de dicho cliente, esto para analizar si los clientes caben dentro del vehículo i. Si $sobra_i$ es mayor o igual a cero se procede a realizar la asignación de dicho cliente a la ruta i y luego actualizar la capacidad de este. En caso contrario, se procede a crear una nueva ruta y asignar el cliente respectivo para luego actualizar la capacidad de la ruta y volver a la ruta anterior con el propósito de llenar completamente su capacidad, dado que los clientes están ordenados de forma decreciente lo que quiere decir que existe la posibilidad de que algún otro cliente con menor demanda entre a dicha ruta. Finalmente se obtienen las 2 rutas antes desarmadas con los clientes libres.

Algorithm 5 Pseudocódigo de BFD

Require: Clientes libres;
1: Ordenar los clientes libres de forma decreciente según sus demandas;
2: **for** Número L de clientes libres **do**
3: $sobra_i = capacidad_i - demanda_L$;
4: **if** $sobra_i >= 0$ **then**
5: Asignar cliente L a ruta i;
6: Actualizar capacidad de ruta i;
7: **else**
8: Crear nueva ruta $i = i + 1$;
9: Asignar cliente L a ruta nueva i;
10: Actualizar capacidad de ruta i;
11: Volver a ruta anterior $i = i - 1$;
12: **end if**
13: **end for**
14: **return** $ruta$;

Ahora tenemos el Algoritmo 6 el cual representa la búsqueda local de Falkenauer. En donde se requiere de la solución S como parámetro de entrada la cual es almacenada en una variable auxiliar S_{aux} para realizar procedimientos sobre esta. Se calcula la demanda total de cada ruta, para luego ordenarlas de menor a mayor con el propósito de rescatar los clientes de la primer y segunda ruta para obtener así los clientes libres y realizar intercambios. Como se rescataron clientes de 2 rutas distintas, estas quedan vacías, por lo tanto se reestructuran rutas nuevas en donde la cantidad de estas se ve reducida en 2 rutas. Para cada ruta se realiza el intercambio de 2 clientes con 2 clientes libres, luego se calcula la holgura de la ruta k, es decir, cuanto espacio libre tiene y si este es mayor a 0, se realiza el intercambio de 2 cliente de la ruta k con 1 cliente libre,

nuevamente se calcula la holgura de la misma ruta y si resulta mayor a 0 se realiza el último intercambio de 1 cliente con 1 cliente libre. Como podemos ver, por cada ruta se intentan realizar los 3 pasos mencionados en párrafos anteriores y después de cada paso se calcula la holgura de dicha ruta para así optimizar líneas de procedimientos. Luego se proceden a crear las nuevas rutas con los clientes que quedaron libres usando el Algoritmo 5 (BFD), una vez terminado, se actualiza la solución auxiliar S_{aux} con todos los cambios realizados para finalmente retornar la nueva solución. A esta nueva solución encontrada, se le aplicará el método de *2opt* representado en el Algoritmo 7, el procedimiento de este es el mismo descrito en el Algoritmo 4 con la diferencia que no contiene la parte de *1 intercambio*.

Algorithm 6 Pseudocódigo de búsqueda local de Falkenauer para ser usado en dos versiones

Require: Solución S;
1: $S_{aux} = S$;
2: Calcular la demanda total de cada ruta de la solución S_{aux};
3: Ordenar las rutas de menor a mayor según la demanda total;
4: Obtener clientes libres de la primera y segunda ruta;
5: Crear nuevas rutas, sin clientes libres;
6: **for** Número k de rutas sin clientes libres; **do**
7: Intercambio de 2 clientes de ruta k por 2 clientes libres;
8: Calcular $holgura_k$ de ruta k;
9: **if** $holgura_k > 0$ **then**
10: Intercambio de 2 clientes de ruta k por 1 cliente libre;
11: **end if**
12: Calcular $holgura_k$ de ruta k;
13: **if** $holgura_k > 0$ **then**
14: Intercambio de 1 cliente de ruta k por 1 cliente libre;
15: **end if**
16: **end for**
17: Crear rutas nuevas con clientes libres usando Algoritmo 5;
18: Actualizar solución S_{aux} con todos los cambios realizados;
19: **return** S_{aux};

Algorithm 7 Pseudocódigo de búsqueda local 2o_f para ser usado en dos versiones

Require: Solución S;
1: $Best2\_Opt = S$;
2: S_aux1 = solución encontrada por Algoritmo 6;
3: **for** número máximo I de iteraciones para búsqueda local **do**
4: $ordenar\_solucion\_aux(S\_aux)$;
5: **for** J número de rutas **do**
6: $S\_aux1 = 2opt(S\_aux1, J)$;
7: **if** Solución S_aux1 es factible **then**
8: **if** $Evaluar(S\_aux1) < Evaluar(Best2\_Opt)$ **then**
9: $Best2\_Opt \leftarrow S\_aux1$;
10: **end if**
11: **end if**
12: **end for**
13: **end for**
14: **if** $Evaluar(Best2\_Opt) < Evaluar(S)$ **then**
15: $S \leftarrow Best2\_Opt$;
16: **end if**
17: **return** S;

4.4. EO aplicado a CVRP

CVRP es un problema catalogado como NP-completo, consta de minimizar el costo de transporte (distancia) y el número de rutas, es decir, encontrar diferentes k rutas que empiezan y terminan en un único depósito para abastecer los requerimientos (demandas) de m clientes a través de vehículos con capacidad limitada, constante y uniforme. La distancia de ir del cliente i al cliente j es la misma que del cliente j al cliente i. Además se debe tener en cuenta la siguiente restricción, la suma de las demandas de cada cliente de su respectiva ruta, no debe ser mayor a la capacidad del vehículo.

Muchas implementaciones para resolver CVRP utilizan métodos complejos, algunas de ellas son: Colonia de Hormigas (Dorronsoro Díaz, 2006), Algoritmos Genéticos (Méndez et al., 2010), métodos de resolución exactos (Salah Niazy and Badr, 2012), entre otros. EO se caracteriza por recorrer el espacio de búsqueda para encontrar una buena solución, moviendose entre soluciones factibles e infactibles (Gómez-Meneses and Randall, 2008). En esta sección se presentará el pseudocódigo de las dos versiones EO1 y EO2, Algoritmo 8 y Algoritmo 9 respectivamente y el proceso que se implementó para

este problema de ruteo.

Para ambas implementaciones, se tuvo que considerar lo siguiente: probabilidad de permutación entre clientes y cambio de ruta. Esto porque la solución inicial aleatoria contiene cierta cantidad de clientes a visitar para cada ruta, si sólo hubieran permutaciones esta cantidad no varía, es siempre constante ya que solo existirían permutaciones. Dado a lo anterior se agregó un nuevo proceso que es el cambio de ruta. Para ejemplificar veamos la Figura 4.5 y Figura 4.6

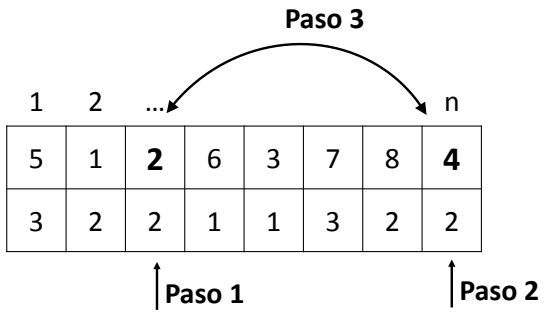

Figura 4.5: Procedimiento cuando ocurre permutación.

Analicemos la permutación de clientes ilustrado en Figura 4.5, EO rankea los clientes en cuestión, todos los clientes si la solución es factible o todos los que están en rutas infactibles. De ese rankig se selecciona uno mediante el método de la ruleta que corresponde al Paso 1. Una vez seleccionado un cliente, EO dice que se debe asignar un nuevo valor aleatorio concerniente al Paso 2. El valor aleatorio no puede ser cualquier número, es decir, existe un rango valores que para nuestro problema es $[1, ..., n]$ son todos los clientes del problema. Una vez encontrado este valor, se realiza el intercambio de componente de la solución mostrado en Paso 3.

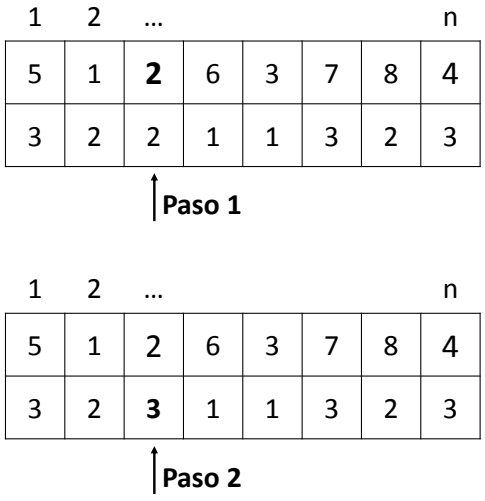

Figura 4.6: Procedimiento cuando ocurre cambio de ruta.

Para el cambio de ruta ilustrado en Figura 4.6 se realiza el Paso 1 que corresponde al ranking de los clientes y selección de uno de ellos, recordar que los clientes en cuestión dependerá de si la solución es factible o infactible. Nuevamente EO dice que se debe asignar un valor nuevo aleatorio, pero para este caso el rango es $[1, ..., k]$ en donde son todas las rutas del problema. Una vez encontrado este nuevo valor, se abre lugar al Paso 2, el cual realiza la asignación de la nueva ruta.

A continuación se describirá el proceso del pseudocódigo de EO1 y no EO2. Esto porque ambas versiones realizan lo mismo, la única diferencia es la Fórmula de evaluación para obtener el fitness ya mencionadas en Fórmula 4.2 y Fórmula 4.5.

El proceso para el Algoritmo 8 es el siguiente. Se calcula la cantidad de rutas mínima sumando todas las demandas de los clientes y a esta suma se divide por la capacidad de los vehículos para luego redondear en una unidad mas, independiente del decimal obtenido, es decir, por ejemplo si la suma total de las demandas da un resultado de 410 y la capacidad de los vehículos es 100 entonces $\frac{410}{100} = 4,1$ la aproximación será 5, lo que quiere decir que se tendrán 5 rutas. Enseguida se inicia una solución S aleatoria, si esta solución es factible, entonces la mejor solución encontrada S_{best} es igual a la solución

factible hasta el momento. EO calcula y usa un vector de probabilidades según lo descrito en la Sección 2.5.2, por lo que se generará este vector. Se realizarán 2.000.000 y 100.000 iteraciones para las versiones canónicas y versiones con búsqueda local respectivamente (en Sección 5.3 se explicará la cantidad de iteraciones). La diferencia de la cantidad de iteraciones entre las versiones radica en que la búsqueda local es exaustiva lo que aumenta el tiempo de ejecución. En cada iteración se consulta por la factibilidad de la solución S, si es factible entonces se realiza el correspondiente cálculo de fitness de acuerdo a la primera parte de la Fórmula 4.2, si es infactible entonces se calcula de acuerdo a la segunda parte de la Fórmula 4.2. Lo anterior para rankear los clientes según corresponda, si se calculó el fitness para solucion factible entonces se rankea de mayor a menor, si fuese infactible se rankea de menor a mayor. Luego se selecciona un cliente a través del método de la ruleta para en seguida arrojar un número aleatorio decimal que determinará que proceso realizar, permutación o cambio de ruta. Si esta nueva solución S es factible, se evaluará para compararla con S_{best} y si es menor, entonces S_{best} se actualiza. Luego se retorna la mejor solución encontrada S_{best}, la cual indicará las rutas y el orden de visita de sus respectivos clientes para finalmente mostrar la distancia de recorrido total de aquella solución, Evaluar (S_{best}).

Algorithm 8 Pseudocódigo de modelo de EO1 para CVRP

```
Inicializar una solucion S aleatoria;
if Solucion S es factible then
    S_best ← S;
end if
Generar vector de probabilidades P_i según τ;
for número de iteraciones do
    if Solucion S es factible then
        Evaluar fitness λ_i de acuerdo a Fórmula 4.2, primera parte;
    else
        Evaluar fitness λ_i de acuerdo a Fórmula 4.2, segunda parte;
    end if
    Rankear los items de menor a mayor o mayor a menor fitness según factibilidad;
    Seleccionar un cliente usando el método de selección de la ruleta y probabilidad
    P_rand;
    Permutar cliente con otro o cambiar de ruta según probabilidad aleatoria;
    if Solucion S es factible then
        if Evaluar(S) < Evaluar(S_best) then
            S_best ← S;
        end if
    end if
end for
return S_best y Evaluar(S_best);
```

Para el Algoritmo 9, como se mencionó anteriormente, se realiza el mismo procedimieno descrito en parrafos anteriores, con al diferencia que la fórmula de evaluación de fitness es distinta.

Algorithm 9 Pseudocódigo de modelo de EO2 para CVRP

```
if Solucion S es factible then
    Evaluar fitness λ_i de acuerdo a Fórmula 4.5, primera parte;
else
    Evaluar fitness λ_i de acuerdo a Fórmula 4.5, segunda parte;
end if
```

La implementación de EO1 y EO2 con búsqueda local, serán llamadas EO1_2o1i y EO2_2o1i respectivamente, en donde lo único diferente a Algoritmos 8 y 9 es lo siguiente: al momento de analizar la factibilidad de la nueva solución generada por EO, esta es usada en la búsqueda local para retornar su respectiva solución. El Algoritmo 10 es la versión de EO1 con búsqueda local (EO1_2o1i), en donde se aprecia la búsqueda local

en linea 16. Para la versión EO2_2o1i es el mismo procedimiento que realiza Algoritmo 10, con la diferencia que la Fórmula para calcular el fitness es la ilustrada en Algoritmo 9.

Algorithm 10 Pseudocódigo de modelo de EO1_2o1i para CVRP

```
1: Inicializar una solucion S aleatoria;
2: if Solucion S es factible then
3:    S_best ← S;
4: end if
5: Generar vector de probabilidades P_i según τ;
6: for número de iteraciones do
7:    if Solucion S es factible then
8:       Evaluar fitness λ_i de acuerdo a Fórmula 4.2, primera parte;
9:    else
10:      Evaluar fitness λ_i de acuerdo a Fórmula 4.2, segunda parte;
11:   end if
12:   Rankear los items de menor a mayor o mayor a menor fitness según factibilidad;
13:   Seleccionar un cliente usando el método de selección de la ruleta y probabilidad
      P_rand;
14:   Permutar cliente con otro o cambiar de ruta según probabilidad aleatoria;
15:   if Solucion S es factible then
16:      S = busqueda_local(S);
17:      if Evaluar(S) < Evaluar(S_best) then
18:         S_best ← S;
19:      end if
20:   end if
21: end for
22: return S_best y Evaluar(S_best);
```

El pseudocódigo de versiones de EO con búsqueda local *2opt* y Falkenauer no se documentarán. Esto porque es el mismo procedimiento de Algormitmo 10 con la diferencia que la solución encontrada por la búsqueda local de Falkenauer es ocupada en el 2 opt descrito en Sección 4.3.

Capítulo 5

Experimentos computacionales

5.1. Introducción

En esta sección de la investigación se realizarán experimentos computacionales a los benchmark seleccionados, generando resultados para su posterior análisis estadístico. Para cada una de las instancias o problemas del benchmarck seleccionado, se aplicarán los algoritmos propuestos ejecutándolos 30 veces con valores de semilla randómicos distintos. Estas instancias serán obtenidas del sitio web de Bernabé Dorronsoro Díaz (Dorronsoro Díaz, 2006). El análisis estadístico a estas pruebas implicará calcular el valor mínimo, mediana y máximo de las 30 ejecuciones, compararlos con los mejores resultados encontrados de cada instancia de benchmark resueltos y concluir respecto a Extremal Optimisation aplicado a problema de ruteo. Una lista para elegir instancias con sus soluciones son las que salen en el sitio web antes mencionado. Algunas de ellos son:

- Augerat et al.
- Breedam.
- Christofides and Elion.
- Fisher.

Se debe recordar que: para realizar los experimentos computacionales, se ocuparán los servidores de UCSC mencionados como una delimitación en la Sección 1.3.

Una vez ejecutadas las versiones sobre las instancias y obtenidos todos los resultados, estos se medirán en %gap respecto a la mejor solución encontrada hasta el momento de cada instancia. Esto para evaluar el desempeño de las versiones implementadas. %gap Se define de la siguiente forma $\%\text{gap} = \frac{b-a}{b}$ donde b es el mejor valor encontrado hasta el momento y a es el valor encontrado de cada instancia por la versión respectiva. Esta investigación abarca un problema de minimización, por lo tanto se ocupará $\%\text{gap} = \frac{a-b}{b}$ esto porque el númerador daría un número negativo si se calculase $b - a$. A medida que mas nos acerquemos a un $\%\text{gap} = 0$ estaremos mas cerca de la mejor solución, donde $\%\text{gap} = 0$ significa que se logró encontrar la mejor solución conocida hasta el momento.

5.2. Set de pruebas (Benchmark)

Las instancias de problema a ocupar será el set A del benchmark Augerat (Dorronsoro Díaz, 2006) que contiene 27 instancias con su respectiva mejor solución encontrada hasta el momento por un método de resolución exácto Branch-and-Cut-and-Price (Fukasawa et al., 2006) y por la metaheurística de Búsqueda Tabú (Rochat and Taillard, 1995). Se seleccionó el set A porque tiene más intancias para resolver que el set B y P, lo que permitirá un análisis más sólido al tener más datos.

Para entender de mejor forma se explicará una instancia, por ejemplo A-n32-k5 donde n32 es el número de nodos en un plano cartesiano de los cuales uno es el depósito y k5 es el número de rutas ocupadas en esa instancia. Finalmente esta instancia se puede interpretar de la siguiente forma: se tiene un depósito y 31 clientes con sus respectivas demandas a satisfacer a través de 5 vehículos con capacidad limitada, es decir, se realiza el ruteo de clientes a través de cinco rutas.

La distancia entre los nodos se calculó mediante la distancia euclidiana obtenida a partir de un nodo origen (x_1, y_1) y uno destino (x_2, y_2) mediante la fórmula $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ donde la distancia de ir a un cliente i a un cliente j es el mismo del j al i.

Cada instancia tiene un tightness, es decir, un cierto grado de dificultad medido

en un número decimal entre 0 y 1. Mientras mas cercano a 0 sea el valor de tightness menos complejo será dicha instacia, y lo contrario si es cercano a 1. La fórmula que se utilizó para indicar este valor fue $\frac{demanda}{capacidad}$, donde la *demanda* es la suma de todas las demandas de los clientes y la *capacidad* representa la suma de todas las capacidades de los vehículos (Ralphs, 2003).

Para decidir con que τ vamos a trabajar, analizamos las versiones EO1 y EO2 para distintos valores de τ en el rango $[1, 2]$ con incremento de 0.1 y semilla 1 a la instancia A-n32-k5 (mejor valor encontrado 784) para 400.000 iteraciones. En la Figura 5.1 se puede apreciar los valores obtenidos para distintos valores de τ reflejados en un gráfico, donde se puede destacar lo siguiente; para EO1 $\tau = 1.6$ y para EO2 $\tau = 1.2$ se obtienen mejores resultados.

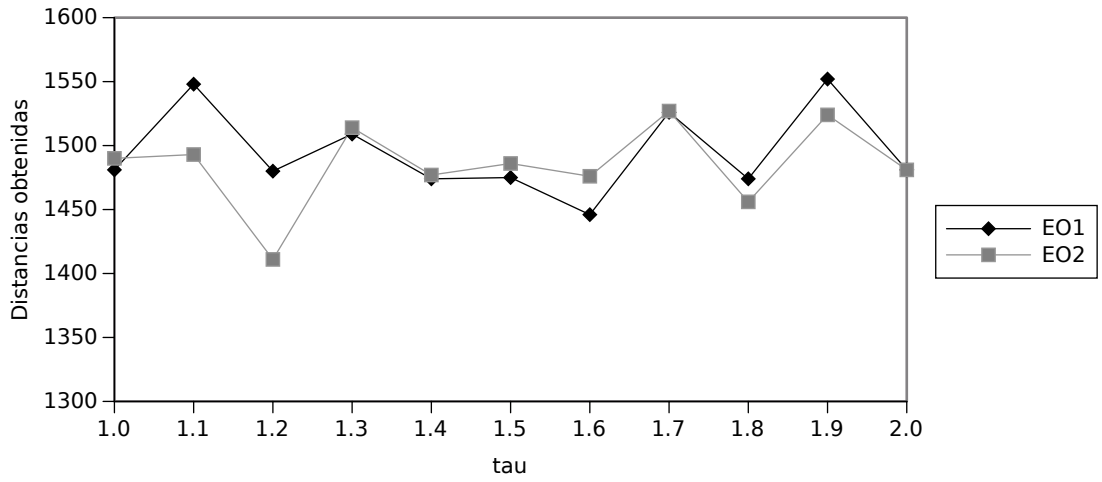

Figura 5.1: Distintos valores de τ arrojan distintos resultados, para la instancia A-n32-k5 y semilla 1.

Según (Boettcher and Percus, 2001), se recomienda para τ un valor de 1.4 en problemas de asignación. Para comprobar que efectivamente es así, decidimos comprobar tres valores de τ los cuales son: 1.2, 1.4 y 1.6 para las versiones EO1 y EO2 con la misma instancia ya usada pero ahora con 30 semillas distintas y 500.000 iteraciones, como lo muestra la Tabla 5.1. Para la tabla anterior se calculó el valor máximo, mínimo, la mediana, desviación estandar y la varianza reflejados en la Tabla 5.2 para los distintos

valores de τ propuestos.

EO1			
Semilla	τ =**1.2**	τ =**1.4**	τ =**1.6**
1	1480	1474	1446
2	1481	1478	1499
3	1529	1410	1475
4	1530	1522	1479
5	1451	1514	1494
6	1471	1467	1424
7	1511	1493	1453
8	1492	1496	1449
9	1403	1456	1505
10	1490	1458	1513
11	1428	1444	1474
12	1482	1479	1511
13	1521	1482	1478
14	1433	1518	1484
15	1449	1451	1491
16	1480	1447	1508
17	1489	1507	1491
18	1524	1445	1485
19	1449	1445	1448
20	1464	1468	1510
21	1510	**1380**	1506
22	1500	1447	1523
23	1515	1491	1517
24	1503	1481	1513
25	1465	1477	1404
26	1409	1493	1485
27	1446	1492	1464
28	1467	1490	1534
29	1490	1491	1416
30	1488	1445	1481

EO2			
Semilla	τ =**1.2**	τ =**1.4**	τ =**1.6**
1	1238	1226	1211
2	1248	1204	1192
3	1250	1223	1200
4	1266	1214	1252
5	1270	1264	1261
6	1270	1222	1223
7	1230	1185	1230
8	1269	1223	1210
9	1317	1220	1146
10	1199	1145	1162
11	1230	1184	1217
12	1220	1199	1260
13	1260	1209	1228
14	1227	1205	1222
15	1241	1193	1247
16	1235	1205	1247
17	1178	1194	1186
18	1276	1183	1237
19	1270	1213	1247
20	1285	1201	1218
21	1263	1175	1196
22	1276	1226	1209
23	1246	1171	1256
24	1267	1194	1213
25	1243	**1123**	1232
26	1193	1227	1250
27	1263	1200	1249
28	1314	1208	1253
29	1259	1179	1201
30	1243	1187	1172

Tabla 5.1: Tabla izquierda corresponde a resultados de versión EO1 y tabla derecha a EO2

Análisis de EO1			
max	1530	1522	1534
medi	1481	1477,5	1485
min	1403	1380	1404
desv	33,98	31,21	32,27
var	1154,64	974,06	1041,35

Análisis de EO2			
max	1317	1264	1261
medi	1254,5	1202,5	1222,5
min	1178	1123	1146
desv	30,82	26,66	29,94
var	949,87	710,76	896,4

Tabla 5.2: Tabla izquierda corresponde a versión EO1 y tabla derecha a EO2

Con esta información concluimos lo siguiente: a pesar de encontrar un τ mejor para

cada versión, los datos nos muestran que la recomendación de (Boettcher and Percus, 2001) es efectiva para este problema, por lo tanto de aquí en adelante usaremos $\tau = 1.4$.

5.3. Resultados

En esta sección se mostrarán los resultados obtenidos para las distintas versiones implementadas. Recordar que se ocupó los servidores proporsionados por UCSC para esta etapa de nuestra investigación, por lo tanto los resultados fueron obtenidos mediante el uso de las distintas versiones a través de estos servidores.

Se sometieron a prueba las siguientes versiones de EO: EO1, EO2, EO1_2o1i, EO2_2o1i, EO1_2o_f y EO2_2o_f. A pesar de que estas versiones no pudieron encontrar las mejores soluciones para cada una de las 27 instancias, algunas de ellas estuvieron cerca, tales como EO1_2o1i y EO2_2o1i.

Para las versiones EO1 y EO2, las pruebas se trataron de la siguinte manera. Cada instancia se ejecutó 30 veces y cada ejecución estuvo compuesta por 2.000.000 iteraciones, entre los 30 resultados obtenidos se buscó el valor mínimo, máximo, mediana y se calculó el %gap de estos tres. Lo anterior se aplicó a ambas versiones mencionadas.

Por otro lado, las versiones con búsqueda local EO1_2o1i y EO2_2o1i, las pruebas se trataron de la siguinte manera. Cada instancia se ejecutó 30 veces y cada ejecución estuvo compuesta por 100.000 iteraciones de EO. A estas iteraciones hay que sumarle la cantidad de iteraciones de la búsqueda local por lo que realizaremos el siguiente calculo de interaciones. A cada iteración de las 100.000 se le suman 20 iteraciones de *2opt* y 20 iteraciones de *1 intercambio*, arrojando un total de $100.000 * (20 + 20) = 4.000.000$ iteraciones. Este es el valor máximo de iteraciones que puede tener el algoritmo en 100.000 iteraciones de EO, debido a que solo se aplica búsqueda local a soluciones factibles. Una vez obtenidos los resultados se realiza el mismo proceso de buscar los valores mínimos, máximos, mediana y calcular el %gap de estos valores. Proceso aplicado a ambas versiones mencionadas.

También se realizaron pruebas para las versiones búsqueda local EO1_2o_f y EO2_2o_f

sobre las cuales se ejecutó en cada instancia 30 veces y cada ejecución estuvo compuesta por 100.000 iteraciones de EO. A estas iteraciones hay que sumarle la cantidad de iteraciones de la búsqueda local. Aquí también se debe sumar la cantidad de iteraciones de la búsqueda local realizando el siguiente cálculo. A cada iteración de las 100.000 se le suman 20 iteraciones de *2opt* y *falkenauer* esto porque la solución de falkenauer se le aplica el mecanismo de *2opt*. La suma a cada iteración arroja un total de $100.000 * 20 = 2.000.000$, valor máximo de iteraciones que puede tener EO en 100.000 iteraciones. Finalmente se realiza el mismo proceso de buscar los valores mínimos, máximos, mediana y calcular el %gap de los los 30 resultados. Este proceso se aplica a ambas versiones mencionadas.

Los resultados de las 6 versiones se encuentran tabulados en %gap en la Tabla 5.3 donde se puede apreciar qué tan distante se encuentran de las mejores soluciones encontradas hasta el momento para cada una de las instancias.

			EO1			EO2			EO1_2o1i			EO2_2o1i			EO1_2o_f			EO2_2o_f		
Instancia	**Valor ***	**Tightness**	**min**	**med**	**max**	**min**	**med**	**max**	**min**	**med**	**max**	**min**	**med**	**max**	**min**	**med**	**max**	**min**	**med**	**max**
A-n32-k5	784	0.82	0,73	0,842	0,895	0,438	0,527	0,573	0,189	0,237	0,268	0,12	0,184	0,226	0,524	0,581	0,67	0,341	0,396	0,452
A-n33-k5	661	0.89	0,731	0,833	0,911	0,455	0,611	0,655	0,169	0,305	0,342	0,191	0,252	0,297	0,517	0,614	0,657	0,29	0,468	0,536
A-n33-k6	742	0.90	0,544	0,683	0,749	0,501	0,623	0,678	0,123	0,354	0,453	0,034	0,385	0,461	0,306	0,52	0,597	0,301	0,499	0,559
A-n34-k5	778	0.92	0,634	0,755	0,821	0,532	0,629	0,688	0,231	0,373	0,416	0,198	0,307	0,388	0,414	0,582	0,638	0,361	0,469	0,542
A-n36-k5	799	0.88	0,757	0,834	0,907	0,484	0,595	0,64	0,282	0,322	0,372	0,159	0,26	0,305	0,519	0,573	0,635	0,317	0,429	0,468
A-n37-k5	669	0.81	0,951	1,047	1,112	0,595	0,677	0,72	0,235	0,283	0,318	0,188	0,245	0,278	0,552	0,718	0,773	0,43	0,496	0,558
A-n37-k6	949	0.95	0,656	0,746	0,825	0,491	0,703	0,798	0,104	0,423	0,626	0,125	0,46	0,602	0,347	0,522	0,702	0,247	0,547	0,697
A-n38-k5	730	0.96	0,993	1,084	1,147	0,86	1,039	1,129	0,415	0,614	0,822	0,308	0,653	0,879	0,6	0,812	0,96	0,448	0,795	0,905
A-n39-k5	822	0.95	0,83	0,932	0,995	0,7	0,866	0,922	0,236	0,523	0,648	0,388	0,554	0,612	0,478	0,711	0,813	0,439	0,663	0,707
A-n39-k6	831	0.88	0,89	1,018	1,088	0,646	0,821	0,899	0,392	0,46	0,528	0,247	0,496	0,544	0,609	0,764	0,836	0,408	0,653	0,769
A-n44-k7	937	0.95	0,877	1,006	1,069	0,811	0,927	1,009	0,339	0,581	0,778	0,271	0,66	0,75	0,61	0,76	0,862	0,586	0,73	0,819
A-n45-k6	944	0.99	1,128	1,363	1,501	1,043	1,243	1,536	0,45	0,779	1,148	0,475	0,886	1,211	0,811	0,947	1,213	0,696	0,882	1,364
A-n45-k7	1146	0.91	0,67	0,739	0,793	0,526	0,602	0,655	0,349	0,414	0,447	0,309	0,385	0,413	0,472	0,553	0,615	0,421	0,488	0,538
A-n46-k7	914	0.86	1,02	1,081	1,148	0,708	0,813	0,864	0,408	0,493	0,524	0,312	0,431	0,464	0,778	0,832	0,867	0,582	0,645	0,729
A-n48-k7	1073	0.89	0,958	1,008	1,068	0,663	0,809	0,844	0,466	0,552	0,593	0,406	0,47	0,51	0,727	0,772	0,823	0,526	0,63	0,713
A-n53-k7	1010	0.95	1,189	1,352	1,443	1,059	1,3	1,39	0,519	0,868	1,102	0,495	0,9	1,088	0,887	1,016	1,226	0,666	0,979	1,189
A-n54-k7	1167	0.96	1,036	1,171	1,223	0,967	1,154	1,219	0,494	0,792	0,984	0,56	0,87	1,022	0,729	0,888	1,057	0,59	0,892	1,043
A-n55-k9	1073	0.93	1,185	1,311	1,371	1,02	1,273	1,341	0,396	0,877	1,126	0,35	0,958	1,141	0,799	1,025	1,232	0,746	0,998	1,212
A-n60-k9	1354	0.92	0,998	1,126	1,216	0,933	1,055	1,127	0,492	0,801	0,919	0,487	0,8	0,895	0,651	0,898	1,013	0,634	0,874	0,958
A-n61-k9	10	0.98	1,121	1,395	1,671	0,976	1,241	1,476	0,506	0,79	1,593	0,444	0,658	1,148	0,838	1,022	1,457	0,724	0,908	1,192
A-n62-k8	1288	0.92	1,23	1,342	1,38	1,079	1,153	1,229	0,689	0,866	0,952	0,719	0,801	0,858	0,968	1,036	1,093	0,865	0,933	1,008
A-n63-k9	1616	0.97	1,002	1,101	1,204	0,91	1,07	1,251	0,456	0,731	0,868	0,584	0,769	1,036	0,569	0,817	0,961	0,584	0,805	0,912
A-n63-k10	1314	0.93	1,148	1,213	1,264	0,959	1,185	1,254	0,531	0,872	1,049	0,473	0,884	1,099	0,786	0,952	1,113	0,721	0,92	1,129
A-n64-k9	1401	0.94	1,045	1,129	1,196	0,883	1,115	1,201	0,512	0,842	0,994	0,476	0,845	1,003	0,707	0,906	1,062	0,63	0,845	1,025
A-n65-k9	1174	0.97	1,468	1,691	1,866	1,333	1,645	1,947	0,852	1,184	1,626	0,706	1,266	1,551	1,083	1,317	1,592	0,971	1,316	1,605
A-n69-k9	1159	0.94	1,555	1,662	1,745	1,343	1,616	1,69	0,971	1,204	1,38	0,887	1,249	1,387	1,198	1,333	1,489	1,022	1,302	1,436
A-n80-k10	1763	0.94	1,245	1,336	1,387	0,976	1,29	1,368	0,782	1,016	1,15	0,717	1,003	1,14	0,923	1,077	1,193	0,86	1,06	1,143
Promedio			0,985	1,104	1,185	0,811	0,985	1,078	0,429	0,65	0,816	**0,394**	**0,653**	**0,789**	0,682	0,835	0,968	0,571	0,764	0,897

Tabla 5.3: Resultados obtenidos medidos en %GAP para el set A de benchmark Augerat . *, soluciones encontrada por Búsqueda Tabú y Branch and Cut.

5.4. Análisis

Se realizaron experimentos computacionales para distintos valores de τ y semilla 1 con el propósito de saber cual de ellos ocupar, estos se encuentran reflejados en la Figura 5.1 los cuales fueron distintos para cada versión, sin embargo, se utilizó un $\tau = 1.4$ con 30 semillas distintas que según la literatura (Boettcher and Percus, 2000) es el valor de τ donde mejores resultados se obtienen para problemas de asignación. Se comprobó que para este problema $\tau = 1.4$ fue efectivo dando mejores resultados que los valores encontrados para cada versión, la Tabla 5.1 refleja el experimento computacional. Con lo anterior podemos reafirmar que para este valor de τ se obtienen los mejores resultados.

Sobre el comportamiento de los resultados de EO aplicado a CVRP en su forma canónica, están lejos de la solución de cada instancia, lo que nos hace ver que EO necesita de alguna otra heurística para lograr encontrar o estar más cerca de la solución, es decir, EO en su forma canónica relativamente arroja soluciones pobres. Esta afirmación también la hemos encontrado en la literatura donde se menciona que la calidad de las soluciones de EO canónico son pobres y de que requiere ayuda adicional, (Randall et al., 2009). La Tabla 5.3 refleja los resultados de las versiones canónicas EO1 y EO2, en donde se aprecia que los valores de %gap son cercanos a 1 incluso superandolos en algunas instancias.

En la literatura, se encontró que los problemas de CVRP siempre se tratan de solucionar desde una perspectiva de ruteo, pero se sabe muy poco desde la perspectiva de empaquetamiento donde cada ruta sería un bin o caja contenedora y los clientes con sus demandas los elementos con su respectivo peso. Esto nos motivó a realizar la implementación de una búsqueda local de empaquetamiento y ruteo juntos. Las versiones EO1_2o_f y EO2_2o_f contienen a esta perspectiva, donde se desarrolló la heurística de Falkenauer y 2opt. Sus resultados tabulados en la Tabla 5.3 fueron mejores que las versiones canónicas, logrando estar cerca de las soluciones actuales para cada instancia.

Como las versiones de EO canónicas no lograron encontrar las soluciones, se implementó una búsqueda local de ruteo, las versiones EO1_2o1i y EO2_2o1i representan este desarrollo y sus resultados están ilustrados en la Tabla 5.3. Aquí podemos ver que

las soluciones mejoraron notoriamente incluso casi resolver una instancia la cual fue A-n33-k6 de versión EO2_2o1i donde el %gap fue de 0.034. Sin embargo, no se logró encontrar las distintas soluciones de cada instancia. Pero sí se concluye de que la búsqueda local de *2opt* y *1 intercambio* es efectiva para problemas de tipo VRP y de ruteo en general.

Para nuestra implementación de la metaheurística EO, inicialmente se calculó la cantidad de rutas a ocupar de forma automática para cada instancia, por lo que el problema de minimizar las rutas se solucionó de forma sencilla quedando solamente minimizar el ruteo. Por lo tanto, tratar de solucionar CVRP desde una perspectiva de empaquetamiento y ruteo no es buen enfoque debido a que ya se calculó la cantidad de rutas mínima.

Por otro lado, enfrentar al problema sólo con empaquetamiento, no se minimizará la distancia repercutiendo en las soluciones donde el objetivo es reducir la distancia y cantidad de rutas, lo que producirá una disperción aleatoria del ruteo de los clientes, pero sí se ocupará de forma eficiente la capacidad de los vehículos. Sin embargo, podemos concluir de que se puede utilizar algún método de empaquetamiento para tratar a las soluciones infactibles de CVRP y volver al espacio de soluciones factibles de forma rápida.

Cabe destacar, que EO en todas sus versiones no discriminó en la resolución de instancias para distintos valores de tightness, es decir, que estos valores no afectan a esta metaheurística. Reflejo de lo anterior son los resultados para las instancias A-n32-k5 y A-n33-k6 para la versión EO2_2o1i, donde los valores de tightness son 0.82 y 0.90 respectivamente y la solución mínima para cada instancia son de $0,12$ y $0,034$, de los cuales podemos ver que para un tightness mayor se obtuvo una mejor solución. Otro ejemplo que nos permitió comprobar que no discriminó, son las instancias A-n36-k5 y A-n38-k5 con sus valores de tightness de 0.88 y 0.96 respectivamente para la misma versión de EO, de los cuales se obtuvieron mejores resultados para instancia que tiene menor valor de tightness, 0.88.

La cantidad de clientes influye en la calidad de la solución. Para las instancias A-n63-k9 y A-n65-k9 que tienen la misma cantidad de rutas, iguales valores de tightness

y distinto número de clientes, se obtienen mejores resultados para la que tiene menor cantidad de clientes. Por lo tanto a mayor cantidad de clientes el escenario se torna más complejo debido a la variabilidad de combinaciones posibles.

La fórmula para calcular el fitness de cada versión es distinta. Una involucra la capacidad disponible (EO1) o excedida del vehículo y otra la distancia del cliente anterior y su posterior (EO2). Aquí podemos apresiar la importancia de crear una buena fórmula de cálculo de fitness a la hora de implementar alguna metaheurística, reflejo de esto son los resultados de las versiones EO1_2o1i y EO2_2o1i para la instancia A-n33-k6 %gap mínimo de 0.123 y 0.034 respectivamente. Una buena creación de fórmula para calcular fitness debe ser guiada según la función objetivo para un mejor desempeño. También se puede ver que en la Tabla 5.3 las distintas versiones de EO2 arrojan resultados mejores que la versión de EO1 para algunas instancias.

Finalmente, EO analiza un nodo o cliente y calcula su fitness según la factibilidad de la solución. Si la solución es factible se procede a calcular el fitness de todos los clientes, independiente de la versión de EO. Se rankea y elije un cliente por medio del método de selección la ruleta y se cambia por un valor aleatorio. En este proceso de cambio pueden ocurrir dos eventos: primero, permutación del cliente y segundo, cambio de ruta del cliente procesos ilustrados en Figura 4.5 y 4.6. En ambos casos es para mejorar una ruta de la solución a la cual pertenece dicho cliente en evaluación, pero es probable que se perjudique otra ruta distinta a la que se está operando, lo que repercute en que la solución entera sea perjudicada debido a que no se sabe si el cambio o permutación favorese o no a la otra ruta. Es decir, se trata de mejorar una ruta y, a su vez, puede empeorarse otra.

Capítulo 6

Conclusiones

A continuación se describirá un resumen del trabajo creado junto con el cumplimiento de los objetivos, la contribución lograda y, finalmente se trazarán trabajos futuros a seguir basados en esta investigación.

Para tener conocimiento y poder entender CVRP y EO, se investigó sobre estos en distintas fuentes de información reflejadas en bibliografía. Se encontraron aspectos relevantes como: que existen distintas variantes de VRP con sus respectivas restricciones; que la mayoría de estas variantes han utilizado el modelamiento matemático usado en investigación de operaciones; que CVRP está compuesto por dos problemas TSP y BPP; y además, que existen distintas técnicas que resuelven este tipo de problemas. Con respecto a EO, se investigó sobre la base de su inspiración, descripción y sus diferencias con Algoritmos Genéticos y sus aplicaciones. Toda esta información se puede encontrar en los Capítulos 2 y 3. Con todo lo anterior, se logró cumplir con dos objetivos específicos: el de Estudiar Extremal Optimisation y el problema de Ruteo de Vehículos con límites de Capacidad; y también el de Modelar el problema a través de la matemática.

Una vez estudiado EO se dió comienzo a la etapa de implementación de esta metaheurística que corresponde al Capítulo 4. Para esto se procedió a modelar algorítmicamente el vector solución y crear las fórmulas para calcular los fitness correspondientes y así evaluar la aptitud de cada variable, se implementó dos fitness distintos para los cuales se obtuvieron resultados diferentes. Por lo tanto, la formulación de un buen fit-

ness para la evaluación de nodos, ayuda en la obtención de buenas soluciones. EO arrojó soluciones pobres, por lo que se decidió implementar una búsqueda local que mejorara las soluciones, demostrando de forma empírica que EO en su versión canónica necesita de ayuda extra para resolver problemas complejos.

Al realizar esta etapa de implementación se logró cumplir con el objetivo de modelar algorítmicamente e implementar EO al problema propuesto.

Las distintas versiones implementadas se aplicaron al set A del benchmark Augerat. Estas pruebas dieron lugar a experimentos computacionales para la elección de τ el cual fue de 1.4. Para este valor de τ los resultados obtenidos de cada instancia se midieron en % gap y se tabularon. Luego de tener los resultados tabulados, se abrió paso a la etapa de análisis para cada proceso de los experimentos realizados. Esta información se encuentra en el Capítulo 5, lo que nos genera el cumplimiento de los siguientes objetivos: Aplicar EO a los benchmark seleccionados y Analizar los resultados obtenidos.

El desarrollo y cumplimiento de cada objetivo en los distintos capítulos de la presente investigación concluye en que el objetivo general que es Aplicar Extremal Optimisation al problema de ruteo de vehículos con límites de capacidad también se cumple.

Finalmente, la diferencia entre los dos métodos que encontraron las soluciones, Branch-and-Cut y Búsqueda Tabú, comparado con EO, radica en que EO debe encontrar la solución en el espacio de búsqueda total mientras que los otros dos métodos van acotando el espacio de búsqueda logrando así ser más eficientes.

6.1. Contribuciones

En esta investigación, se pueden rescatar las siguientes contribuciones.

- Aplicar por primera vez EO al problema de ruteo de vehículos con límites de capacidad. Este estudio permitirá a futuros investigadores conocer las ventajas y desventajas de EO aplicado a CVRP. Sirviendo de base a nuevas ideas para ser aplicadas en otros escenarios de estudio.
- Se demuestra empíricamente que una búsqueda local eficiente que complementa

a EO para el problema CVRP es 2opt y 1 intercambio.

- Tratar de solucionar este tipo de problema desde un enfoque de empaquetamiento, para nuestro caso CVRP, no es buena práctica. Esto porque solo se está optimizando la capacidad de los vehículos y no la distancia. Sí sería buena práctica implementar una heurística de empaquetamiento, de Falkenauer o cualquier otra, para enfrentar la infactibilidad de la solución y volver al espacio de soluciones factibles de forma rápida y así optimizar iteraciones de la metaheurística en implementación.

6.2. Trabajo futuro

Se detectaron los siguientes posibles trabajos futuros basados en esta investigación los cuales son:

- Relizar iteraciones con valores de τ mayores a 2, para corroborar algún posible mejor comportaminto de EO.
- Probar un modelamiento en vector solución distinto por EO.
- Mejorar soluciones iniciales mediante heurística del vecino mas cercano.
- Probar otras fórmulas para el cálculo del fitness a cada individuo de la solución.
- Implementar un catalizador para las soluciones infactibles.
- Implementar heurística de empaquetamiento a soluciones infactibles.
- Implementar un algormito híbrido de EO con Búsqueda Tabú, y así evitar caer en óptimos locales mediante el uso de una estructura de memoria.
- Implementar algún mecanismo que permita acotar el espacio de búsqueda de EO.
- Implementar una versión de EO basado en poblaciones para ampliar el área de búsqueda.
- Implementar una versión paralela de EO trabajando con distintos valores de τ de forma simultánea.

Bibliografía

Alba, E. and Dorronsoro, B. (2008). A hybrid cellular genetic algorithm for the capacitated vehicle routing problem. In Abraham, A., Grosan, C., and Pedrycz, W., editors, *Engineering Evolutionary Intelligent Systems*, volume 82 of *Studies in Computational Intelligence*, pages 379–422. Springer Berlin Heidelberg.

Alba, E. and Dorronsoro, B. (2009). *Cellular Genetic Algorithms*. Operations Research/Computer Science Interfaces Series. Springer US.

Boettcher, S. and Percus, A. (1999). Extremal optimization: Methods from co-evolution. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 825–832.

Boettcher, S. and Percus, A. (2000). Natures way of optimizing. *Artificial Intelligence*, 119(1):275–286.

Boettcher, S. and Percus, A. (2001). Extremal optimization for graph partitioning. *Physical Review E*, 64:026114.

Brownlee, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. Lulu.com, 1st edition.

Calvaresi, D. (2012). Web application CVRP. Recuperado de http://davidecalvaresi.it/. Fecha de último acceso: 8 de marzo de 2016.

Chen, M.-R., Li, X., Zhang, X., and Lu, Y.-Z. (2010). A novel particle swarm optimizer hybridized with extremal optimization. *Applied Soft Computing*, 10(2):367 – 373.

Chen, Y.-W., Zhu, Y.-J., Yang, G.-K., and Lu, Y.-Z. (2011). Improved extremal optimization for the asymmetric traveling salesman problem. *Physica A: Statistical Mechanics and its Applications*, 390(23):4459–4465.

Coffman, E., Garey, M., Graham, R., and Johnson, D. (2016). Bin packing. Recuperado de http://www.ams.org/samplings/feature-column/fcarc-bins2. Fecha de último acceso: 8 de marzo de 2016.

Cruz-Reyes, L., Quiroz C., M., Alvim, A. C. F., Fraire Huacuja, H. J., Gómez S., C., and Torres-Jiménez, J. (2012). Heurísticas de agrupación híbridas eficientes para el problema de empacado de objetos en contenedores. *Computación y Sistemas*, 16(3):349–360.

Daza, J. M., Montoya, J. R., and Narducci, F. (2009). Resolución del problema de enrutamiento de vehículos con limitaciones de capacidad utilizando un procedimiento metaheurístico de dos fases. *Revista EIA, Escuela de Ingeniería de Antioquia, Medellín (Colombia)*, 6(12):23–38.

Dorronsoro Díaz, B. (2006). The VRP web. Recuperado de http://www.bernabe.dorronsoro.es/vrp/. Fecha de último acceso: 8 de marzo de 2016.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2(1):5–30.

Flores Cabezas, X. A. (2014). Problemas del milenio: P vs np. *Revista de Divulgación, Amarun*, 1(2014):1–15.

Fukasawa, R., Longo, H., Lysgaard, J., Aragao, M. P. d., Reis, M., Uchoa, E., and Werneck, R. F. (2006). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, 106(3):491–511.

Garcia Calves, P. (2002). Problemas de optimización combinatoria. Recuperado de http://www.iiia.csic.es/udt/es/artificialintelligence/problemas-optimizacion-combinatoria. Fecha de último acceso: 8 de marzo de 2016.

Gómez, J. M. (2003). Fundamentos teóricos de ecología evolutiva: Coevolución. Recuperado de http://www.ugr.es/ jmgreyes/coevolucion.html. Fecha de último acceso: 8 de marzo de 2016.

Gómez-Meneses, P. and Randall, M. (2008). Extremal optimisation with a penalty approach for the multidimensional knapsack problem. In Li, X., Kirley, M., Zhang,

M., Green, D. G., Ciesielski, V., Abbass, H. A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K. C., Branke, J., and Shi, Y., editors, *SEAL*, volume 5361 of *Lecture Notes in Computer Science*, pages 229–238. Springer.

Gómez-Meneses, P. and Randall, M. (2009). A hybrid extremal optimisation approach for the bin packing problem. In Korb, K. B., Randall, M., and Hendtlass, T., editors, *Artificial Life: Borrowing from Biology*, volume 5865 of *Lecture Notes in Computer Science*, pages 242–251. Springer Berlin Heidelberg.

Gómez Meneses, P. S. (2012). *Extremal optimisation applied to constrained combinatorial multi-objective optimisation problems*. Phd thesis, School of Information Technology, Bond University.

Huang, X. (2007). Cooperative optimization for energy minimization: a case study of stereo matching. *arXiv preprint cs/0701057*.

Jaramillo Posada, J. R. (2012). Búsqueda tabú para el ruteo de vehículos. *Ingeniería Industrial*, 30(14):22–49.

Lorena Stockdale, M. (2011). El problema del viajante: un algoritmo heurístico y una aplicación. Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales. Departamento de Matemática. Tesis de Licenciatura.

Méndez, A., Simón, S., Palumbo, D., Chiachera, E., and Carnero, M. (2010). Dos enfoques para la solución del problema de ruteo de vehiculos (cvrp): Aplicación a un caso real de recolección de residuos. *Mecánica Computacional*, 29(15):9367–9377.

Moreno Perez, J. A. (2004). Metaheuristicas: Concepto y propiedades. universidad de la laguna, España. Recuperado de http://www.tebadm.ulpgc.es/almacen/seminarios/MH%20Las%20Palmas%202.pdf. Fecha de último acceso: 8 de marzo de 2016.

Orrego Cardozo, J. P. (2013). Solución al problema de vehiculos con capacidad limitada CVRP a través de la heurística de barrido y la implementación del algoritmo genético de chu-beasley. *Universidad Tecnológica de Pereira*. Trabajo de grado para optar al título de Ingeniero Industrial.

Ralphs, T. (2003). Vehicle routing data sets. Recuperado de http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm. Fecha de último acceso: 8 de marzo de 2016.

Ralphs, T., Hartman, J., and Galati, M. (2001). Capacitated vehicle routing and some related problems. Recuperado de http://www.bernabe.dorronsoro.es/vrp/. Fecha de último acceso: 8 de marzo de 2016.

Randall, M., Hendtlass, T., and Lewis, A. (2009). Extremal optimisation for assignment type problems. In Lewis, A., Mostaghim, S., and Randall, M., editors, *Biologically-Inspired Optimisation Methods: Parallel Algorithms, Systems and Applications*, volume 210 of *Studies in Computational Intelligence*, pages 139–164. Springer-Verlag.

Randall, M., Rawlins, T., Lewis, A., and Kipouros, T. (2015). Performance comparison of evolutionary algorithms for airfoil design. *Procedia Computer Science*, 51:2267 – 2276. International Conference On Computational Science, {ICCS} 2015 Computational Science at the Gates of Nature.

Resende, M. and Velarde, G. (2003). Grasp: Procedimientos de búsqueda miopes aleatorizados y adaptativos. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, (15):61–76.

Rochat, Y. and Taillard, E. D. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1(1):147–167.

Salah Niazy, N. and Badr, A. (2012). Complexity of capacitated vehicles routing problem using cellular genetic algorithms. *International Journal of Computer Science and Network Security*, 12(2).

Shmygelska, A. (2007). An extremal optimization search method for the protein folding problem: The go-model example. In Thierens, D., editor, *Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation*, pages 2572–2579.

Zeng, G.-Q., Lu, K.-D., Chen, J., Zhang, Z.-J., Dai, Y.-X., Peng, W.-W., and Zheng, C.-W. (2014). An improved real-coded population-based extremal optimization method for continuous unconstrained optimization problems. *Mathematical Problems in Engineering*, 2014:9.